CACiC 2023
CONGRESO ARGENTINO DE CIENCIAS DE LA COMPUTACIÓN

Reconocimiento de Logotipos de Empresas Tecnológicas mediante Redes Neuronales Artificiales

Ignacio Ibañez[1], Juan Bautista Rodriguez[1], Cristian Desiderio Arana[1], Cinthia Vegega[1,2], María F. Pollo-Cattaneo[1,2]

[1] Universidad Tecnológica Nacional. Facultad Regional La Plata. Argentina
[2] GEMIS. Universidad Tecnológica Nacional. Facultad Regional Buenos Aires. Argentina
ignacio@innovotecnologias.com.ar, cinthia.vegega@gmail.com, flo.pollo@gmail.com

Resumen. En el presente trabajo, se detallan y comparan los resultados obtenidos de la aplicación de dos arquitecturas posibles de Redes Neuronales Artificiales, con el fin de reconocer logotipos de empresas del sector tecnológico, en el marco de un trabajo de cátedra de la asignatura Inteligencia Artificial de la carrera de Ingeniería en Sistemas de Información de la Universidad Tecnológica Nacional, Facultad Regional La Plata.

Palabras Clave: Inteligencia Artificial, Redes Neuronales Artificiales, Redes Convolucionales, Redes Bakpropagation

1. Introducción

Dentro del ámbito de la Universidad Tecnológica Nacional, Facultad Regional La Plata, la asignatura Inteligencia Artificial dictada en el quinto año de la carrera de Ingeniería en Sistemas de Información tiene como uno de sus objetivos que el estudiante logre analizar, formalizar e implementar problemas a través de la implementación de Redes Neuronales Artificiales, aplicando metodologías, técnicas y herramientas pertenecientes a las buenas prácticas profesionales [1]. Es a partir de este objetivo que las docentes de la cátedra, María Florencia Pollo-Cattaneo y Cinthia Vegega (que pertenecen al grupo de investigación GEMIS [2]), proponen a los estudiantes que transitan esta materia, la realización de un proyecto que pueda ser resuelto utilizando dicha tecnología. En este contexto, un grupo de estudiantes propone para su trabajo, la comparación de dos tipos de Redes Neuronales Artificiales con el objetivo de abordar la problemática del reconocimiento de logotipos de empresas del rubro tecnológico ya que la aplicación de software tradicional para este tipo de tarea resultaría laboriosa y poco eficiente [3]. En primera instancia se describen los elementos de trabajo (sección 2), para luego mostrar los resultados obtenidos (sección 3). Finalmente, se presentan las conclusiones y futuras líneas de trabajo (sección 4).

2. Elementos de Trabajo

El objetivo final del trabajo es obtener dos modelos predictivos capaces de reconocer y distinguir el logo de cada una de las siguientes empresas del rubro tecnológico: AMD, Apple, Cisco, IBM y Microsoft. La situación ideal es que sean capaces de reconocer el logo dispuesto de distintas formas y con sus respectivos

cambios a lo largo de los años. Generalmente, la aproximación a este tipo de casos consiste en aplicar Redes Neuronales Convolucionales (CNN) [4] en dos vertientes: Logo Detection (LD), a fin de determinar la posición del logo, y Logo Recognition (LR), que trata de identificar a qué empresa pertenece dicho logo. En este caso, se analiza la última vertiente. Además, se considera conveniente utilizar dos enfoques de trabajo distintos, y luego comparar los resultados obtenidos a través de cada uno de ellos:

- *Red Neuronal Artificial Convolucional realizada en Python con librería externa*
- *Red Neuronal Artificial Feedforward con Backpropagation desarrollada en C#*

Para ello, se utiliza un conjunto de imágenes de prueba y entrenamiento de un repositorio público de GitHub "Logo-2k-plus-Dataset" [5].

El primer enfoque consiste en la utilización de una Red Neuronal Artificial (RNA) del tipo convolucional. Se trata de una variante de las redes perceptrón multicapa (o multi-perceptrón), y se las considera ideales para aquellas tareas que involucren "visión artificial" como, por ejemplo, la clasificación y segmentación de imágenes, dado que "imitan" el comportamiento de las neuronas que componen la corteza visual primaria en un cerebro humano. Se utiliza Python como lenguaje de programación, en conjunto con las librerías *TensorFlow* [6] y *Keras* [7]. Ambas de código abierto, ampliamente utilizadas y difundidas en el campo del aprendizaje automático y de la construcción de redes neuronales. La topología se conforma por 4 capas. La primera se denomina *Flatten* o capa de aplanamiento. Se utiliza para convertir los datos de entrada (las imágenes) en un vector unidimensional. La segunda es *Dense (1/2)*. Está totalmente conectada y se compone de 128 neuronas. Su función de activación es ReLu (unidad lineal rectificada). La tercera es *Dropout***.** Si bien se considera una capa, el "dropout" o "capa de abandono" es, en realidad, una técnica de regularización utilizada para reducir el sobreajuste en la red neuronal. Particularmente para esta red, se estableció en un valor de 0,2 de las entradas a cero de forma aleatoria en cada paso de actualización. La última es *Dense (2/2)* que es la capa de salida, también totalmente conectada y compuesta únicamente de 5 neuronas (correspondientes a cada una de las posibles clasificaciones / empresas). La función de activación es Softmax, y permite estimar la probabilidad de que la entrada pertenezca a cada clase [8]. Por último, cabe mencionar que se selecciona como optimizador al *algoritmo de Adam*, basado en gradientes, y *Categorical Cross Entropy* como función de pérdida.

El segundo enfoque, sin vinculación con librerías de terceros específicas, se logra al ejecutar un experimento en el entorno de desarrollo de software Microsoft .NET [9, 10] conformado por una solución compuesta de 3 proyectos individuales. El primero, es una aplicación de escritorio capaz de consultar imágenes en la web, para luego descargarlas, redimensionarlas y ubicarlas en una estructura de directorios adecuada para el entrenamiento y prueba de la RNA El segundo, es un proyecto de tipo *biblioteca* denominado RNA que contiene las clases que constituyen la red neuronal en sí. Puede utilizarse desde diferentes tipos de proyectos y plataformas presentando una RNA con topología de 3 capas (entrada, oculta y salida). La RNA es del tipo *feed-forward* (todas sus conexiones son hacia adelante, no hay conexiones hacia atrás o recursivas) con aprendizaje mediante el algoritmo *backpropagation*. Este algoritmo es del tipo "corrección del error por propagación hacia atrás" (del inglés *backpropagating error correction*). Se implementa mediante optimización iterativa de primer orden en lo que

se conoce como “descenso del gradiente”. Permite encontrar mínimos locales de una función diferenciable (las de activación de la RNA). La interpretación del gradiente es fundamental: indica la dirección de máximo crecimiento de la función. Por lo tanto, si se buscan mínimos locales, se debe tomar la misma dirección del gradiente en el punto analizado, pero en sentido contrario. La cantidad de neuronas de entrada está íntimamente relacionada con la cantidad de pixeles que constituyen la imagen. Así, cuando se trabaja con imágenes de 200 pixeles, se generan 40 mil neuronas de entrada (200 * 200). La capa de salida tiene una neurona por cada resultado posible. Al evaluar 5 logos de empresas, se cuenta con 5 neuronas de salida. Al solicitarle a la RNA que infiera el nombre de un logo presentado, las 5 neuronas resultan en un valor comprendido entre 0 y 1. Un valor de 1 indica que esa neurona presenta “certeza”. El último, es otro proyecto de ventanas de escritorio que permite entrenar, probar y hacer uso de la RNA para la identificación de logos. Este último proyecto se denomina App y referencia al proyecto RNA antes mencionado para hacer uso de la red neuronal. Presenta una interfaz gráfica (figura 1) capaz de establecer la función de activación (Sigmoide, Tangente hiperbólica, ReLU o Leaky ReLU), la cantidad de neuronas en la capa oculta, las iteraciones (*epochs*) y la tasa de aprendizaje inicial. Por último, presenta un control que permite activar la recreación de la RNA en cada entrenamiento. Para monitorear el funcionamiento de la misma se presenta el elemento (la empresa) de la cual se están procesando sus logos y debajo del nombre de la empresa, se visualiza la imagen identificada. Sobre la derecha, la tasa de aprendizaje y la exactitud. Cuanto menor sea el valor de la tasa de aprendizaje, más lento se realizará ese proceso. En el caso de la exactitud, es óptima si vale 1. A fin de lograr rigurosidad en el lenguaje, se aclara que se utiliza genéricamente el término logo para referirse a logotipos, imagotipos (texto e imagen), isotipos o isologos.

Fig. 1. Interfaz Proyecto App.

3. Resultados

Referido a la **RNA desarrollada en Python**, se observa que el error general obtenido durante la fase de entrenamiento oscila en cada corrida del código. Sin embargo, tras varias ejecuciones, el entrenamiento resulta en 84,36% de precisión para 35 épocas. Si bien no se trata de una muestra realmente significativa para obtener conclusiones definitivas, preliminarmente se observa que la precisión obtenida (80%), es similar a la que arrojó la librería *Keras* durante la etapa de entrenamiento. Para obtener una aproximación más rigurosa, con un volumen mayor de corridas, a través de un sencillo algoritmo se toma el 25,6% de las imágenes disponibles en los conjuntos, para validar la predicción de la red entrenada (22 de AMD, 18 de Apple, 14 de Cisco,

22 de IBM y 24 de Microsoft, totalizando 100 imágenes). En este caso, los resultados obtenidos son los siguientes: 91 validaciones correctas y 9 incorrectas. El error es del 9%, aún menor que el error general calculado durante el entrenamiento.

Referido a la **RNA desarrollada en C#**, se alcanza una exactitud máxima de 83% mediante la función de activación Sigmoide, 50 neuronas en la capa oculta y una tasa de aprendizaje de 0,0001. Este valor se logró con aproximadamente 1.500 *epochs*. Este enfoque presenta varias ventajas y desventajas que se enuncian a continuación, entendiéndose, tácitamente, que aquellas que resulten una ventaja de una de las opciones será una desventaja de la otra.

Ventajas

- El control sobre los algoritmos en ejecución es total.
- No hay costos potenciales de, por ejemplo, licencias que los desarrolladores de la librería puedan establecer a futuro.
- El ciclo de vida del *software* no depende de terceros.
- Puede registrarse la propiedad intelectual (puede haber restricciones, pero ciertamente menos que con una librería externa).
- La seguridad es íntegramente controlada por la organización o particular propietario. Es importante destacar aquí, que ante una solución de código abierto como es TensorFlow, podríamos analizar el código en busca de brechas de seguridad. Luego de pasar exitosamente el análisis podríamos compilarla en infraestructura propia para lograr un alto grado de seguridad.
- Desde el punto de vista conceptual, resulta más provechosa dado que expone la estructura interna y sus pilares conceptuales fundamentales mucho más explícitamente. Esto puede facilitar el ajuste de los parámetros para mejores los resultados.
- Debe respaldarse en bibliografía teórica que puede involucrar conceptos de otras áreas de conocimiento. En contrapartida, las librerías externas con cierto grado de maduración ponen a disposición manuales de uso que suelen ser de rápido aprovechamiento.

Desventajas

- Se requieren integrantes en el equipo de desarrollo con un nivel de experiencia medio o alto en el marco de trabajo seleccionado (.NET en este caso). Por el contrario, el uso de librería externas suele favorecer a los programadores inexpertos (desde uno o dos años de experiencia en la actividad).
- La curva de aprendizaje resulta en mayor tiempo insumido para ser capaces de aportar mejoras y disponer de un prototipo funcional con resultados aceptables.
- Desarrollar una solución propia conlleva cierta falta de certeza al momento de escalar en requerimientos. Las librerías de terceros desarrolladas por grandes comunidades (como TensorFlow de Google) brindan un respaldo fundamental en este aspecto y se nutren de aportes constantes de su comunidad. Además, no es

posible variar con facilidad la topología, sus funciones de activación y demás parámetros.

Finalmente, se concluye que ambos enfoques propuestos ofrecen resultados acordes, cada uno con sus fortalezas y debilidades.

4. Conclusiones

El objetivo de este trabajo consiste en reconocer logotipos de empresas del sector tecnológico utilizando para ello dos enfoques: una Red Neuronal Artificial Convolucional realizada en Python con librería externa y una Red Neuronal Artificial Feedforward con Backpropagation desarrollada en C#. Los resultados obtenidos con ambos enfoques fueron exitosos para el objetivo planteado inicialmente. La red neuronal convolucional desarrollada en Python presenta un rendimiento aceptable al retornar más del 80% de aciertos. Por otro lado, la red desarrollada en C# también alcanza una exactitud de más del 80%. Como futuras líneas de trabajo, dado que el dataset propuesto para las pruebas y entrenamiento es relativamente pequeño, se podría ampliar ajustando, además, algunos parámetros para obtener un error menor en cada una de las redes utilizadas. Asimismo, este trabajo se presentará como ejemplo de aplicación el cuatrimestre que viene, dejándolo disponible para que los estudiantes que cursan la asignatura de Inteligencia Artificial puedan utilizarlo como apoyo para sus respectivos trabajos de cátedra.

Referencias

1. Universidad Tecnológica Nacional, Facultad Regional La Plata (2023). Programa Analítico de la asignatura Inteligencia Artificial. Disponible en: https://www.frlp.utn.edu.ar/sites/default/files/inteligenci_artificial_1150.pdf Último Acceso: 29/07/2023.
2. Grupo GEMIS (2023). Historia Grupo GEMIS. Disponible en http://grupogemis.com.ar Último Acceso: 29/07/2023.
3. Nielsen, M. A. (2018). Neural Networks and Deep Learning. Neuralnetworksanddeeplearning.com; Determination Press. Disponible en: http://neuralnetworksanddeeplearning.com/chap1.html Último Acceso: 29/07/2023.
4. Raúl Castilla Bravo. Universidad Europea de Madrid. Reconocimiento de logotipos de marcas mediante Redes Neuronales de Convolución (CNN). https://titula.universidadeuropea.com/bitstream/handle/20.500.12880/767/CastillaBravoRaul.pdf Último Acceso: 29/07/2023.
5. A Large-Scale Logo Dataset for Scalable Logo Classification. GitHub. https://github.com/msn199959/Logo-2k-plus-Dataset Último Acceso: 29/07/2023.
6. API Documentation de TensorFlow Core v2.1.0. (n.d.). www.tensorflow.org/api_docs Último Acceso: 29/07/2023.
7. Team, K. (n.d.). Keras documentation: Keras API reference. Keras.io. https://keras.io/api
8. Neural networks. (n.d.). Ml4a.github.io. https://ml4a.github.io/ml4a/neural_networks Último Acceso: 29/07/2023.
9. Gary, Miller. A 3-Layer Feed-Forward Neural Net in C#, with Graphical Display. https://www.codeproject.com/Articles/5272136/A-3-Layer-Feed-Forward-Neural-Net-InCsharp-with-G Último Acceso: 29/07/2023.

10. García Martínez, Servente & Pasquini (2007) Sistemas Inteligentes. Editorial Nueva Librería.