Reúso de un modelo de Aprendizaje Profundo para reconocimiento de dígitos manuscritos

Mauro José Pacchiotti[1], Luciana Ballejos[1] y Mariel Ale[1]

[1] Centro de I+D de Ingeniería en Sistemas de Información
UTN-FRSF
{mpacchiotti, lballejos, male}@frsf.utn.edu.ar

Resumen. Las técnicas de Aprendizaje Automático (AA) han avanzado significativamente en la solución de diversos problemas, lo que ha llevado a una amplia difusión en su uso y desarrollo. Actualmente existen distintos modelos que han alcanzado un alto nivel de desempeño, lo que plantea la duda de qué hacer cuando nos enfrentamos a un problema para el cual ya existe un modelo muy eficiente. Desde hace tiempo esta situación ha impulsado la investigación y el desarrollo de diferentes técnicas para reutilizar estos modelos, en lugar de emprender el diseño, implementación y entrenamiento de uno nuevo, con todo el esfuerzo que ello conlleva. En este trabajo se presenta un problema de clasificación y se propone la reutilización de una red neuronal convolucional con el objetivo de reconocer números manuscritos. Asimismo, se ha evaluado el desempeño del modelo reutilizado.

Palabras clave: Transferencia de Aprendizaje, Reúso de modelos, Aprendizaje Automático, Reconocimiento de dígitos numéricos.

1 Introducción

Con el avance de las técnicas de AA en algunos dominios, se han creado conjuntos de datos públicos que han motivado la búsqueda de modelos que mejoran cada vez más diversas tareas. A medida que algunos modelos han alcanzado altos niveles de rendimiento, se han aplicado distintas técnicas de transferencia de aprendizaje o reutilización con el fin de utilizar los modelos entrenados y adaptarlos a conjuntos de datos distintos a los que fueron utilizados en su entrenamiento [1].

La reutilización suele ocurrir más a menudo con modelos que solucionan problemas genéricos sobre dominios más comúnmente utilizados. En estos casos se pueden reutilizar modelos preentrenados que se ajustan al volver a entrenar las últimas capas con un nuevo conjunto de datos o, incluso, utilizar el modelo tal cual está, sin realizar ningún ajuste [2]. En este trabajo se propone utilizar un modelo bien conocido para solucionar un problema de clasificación de imágenes. Este modelo ha sido entrenado con un conjunto de datos también popular, pero se aplicará a imágenes que nunca ha procesado previamente.

A partir de la necesidad de agilizar el registro de los datos de telegramas electorales, se requirió de la utilización de un modelo de AA para reconocer las cifras escritas manualmente en los telegramas. La estrategia propuesta entonces fue reutilizar un modelo que tenga un buen desempeño en este tipo de tarea y un dataset de dígitos

manuscritos con la cantidad suficiente de muestras para entrenar este modelo. Esta propuesta tiene como objetivo reducir los costos y tiempos que implicarían, principalmente, las actividades de generación de un dataset etiquetado, con el balance y la cantidad suficiente de muestras.

Para la generación de los conjuntos de datos sobre los que se iba a reutilizar el modelo se recurrió a telegramas de años anteriores que estaban cargados -manualmente- en una base de datos con números de una, dos y tres cifras. Esto implicó que, para generar los conjuntos de imágenes a clasificar, fue necesario extraer los números de las celdas del telegrama y separar luego los dígitos de estos números para conformar el dataset de dígitos del 0 al 9. Finalmente, dado que el conjunto de telegramas se encontraba registrado en una base de datos correspondiente a los resultados de dicha elección, se utilizaron esos valores a modo de etiquetas de los datos para poder evaluar luego la precisión de la solución desarrollada.

En resumen, el trabajo tiene como objetivo evaluar la capacidad de un modelo de aprendizaje profundo entrenado con un dataset reutilizado para clasificar números manuscritos desconocidos, utilizando un conjunto de datos generado a partir de documentos que contienen números de diferentes cifras. Para lograr esto, se implementó un clasificador y un proceso de preprocesamiento de imágenes para preparar los datos de entrada del modelo.

En este sentido, los aportes de esta propuesta son:

1. Un dataset etiquetado con imágenes de números manuscritos de múltiples dígitos [4].
2. Un dataset etiquetado con imágenes de dígitos numéricos manuscritos del 0 al 9 [5].
3. La evaluación del prototipo utilizado aplicando reúso del modelo y el dataset de entrenamiento.

El resto del trabajo está organizado de la siguiente forma: la Sección 2 discute el marco teórico y propuestas existentes para la reutilización de modelos AA. La Sección 3 expone la preparación de los datos, modelos y la implementación del prototipo. La Sección 4 detalla las pruebas realizadas y resultados obtenidos, y la Sección 5 resume las conclusiones y trabajos futuros.

2 Marco Teórico y Trabajos Relacionados

En los últimos años se han realizado muchos estudios sobre la reutilización de modelos de aprendizaje profundo en diferentes dominios. Entre ellos, Kumar, Gupta y otros [6] desarrollan un modelo capaz de clasificar pacientes como positivos para COVID-19, neumonía, tuberculosis o sanos, a partir de imágenes de rayos X y reutilizando modelos preentrenados. Chemmakha y Lazaar [7] implementan un modelo convolucional al que aplican transferencia de aprendizaje con dos modelos preentrenados, MobileNetV2 y ResNet-50, para la clasificación de malware a partir de imágenes en miniatura obtenidas con la proyección de los primeros 1024 bytes del archivo ejecutable.

Más relacionados al objetivo de este trabajo, también existen estudios que utilizan el modelo de transferencia de aprendizaje en reconocimiento de dígitos escritos a mano. Así, Finjan, Rasheed y otros [8] utilizan transferencia de aprendizaje con el modelo preentrenado Resnet-34 sobre redes convolucionales para reconocimiento de dígitos en árabe. Aneja y Aneja [9] aplican transferencia de aprendizaje a una red convolucional

profunda a partir de los modelos preentrenados AlexNet, DenseNet-121, DenseNet-201, Vgg-11, Vgg-16, Vgg-19 e Inception-V3 para reconocer dígitos de alfabetos Devanagari y presentan un análisis comparativo de los resultados obtenidos con los distintos modelos preentrenados. También Zhang [10] compara los resultados obtenidos con el uso del entrenamiento convencional y la transferencia de aprendizaje para el reconocimiento de dígitos numéricos escritos a mano entre cinco conjuntos de datos numéricos: tibetano, árabe, bengalí, devanagari y telugu utilizando dos modelos: un perceptrón multicapa y una red convolucional.

Particularmente, en este trabajo, se reutiliza un modelo convolucional entrenado con el dataset MNIST para el reconocimiento de dígitos numéricos.

2.1 Transfer Learning

La transferencia de aprendizaje (Transfer Learning) entre distintos modelos de aprendizaje automático surgió de la idea de emular la capacidad de los humanos de transferir conocimiento entre tareas similares. Esta técnica permite reutilizar el aprendizaje adquirido en una tarea para mejorar el desempeño en otra tarea relacionada [11].

Si bien el Transfer Learning fue introducido por Stevo Bozinovski y Ante Fulgosi [12][13] en 1976, actualmente, este concepto promete ser el próximo impulsor del éxito comercial del aprendizaje automático según menciona Andrew Ng [14].

Pan y Yang [15] proponen una definición de Transfer Learning a partir de los conceptos de Dominio y Tarea.

Un dominio D está definido por un espacio de características χ y una probabilidad marginal $P(X)$ donde $X = \{x_1, x_2, \dots, x_n\} \in \chi$. Entonces χ es todo el espacio de posibles características, x_i es la i-esima característica contenida en algunas muestras de los datos de entrenamiento y X es una muestra de aprendizaje particular con un subconjunto de características de χ. Dos Dominios son diferentes si son diferentes sus espacios de características o sus distribuciones de probabilidad marginal.

Dado un dominio $D = \{\chi, P(X)\}$, una tarea comprende todo el conjunto de etiquetas Y y un modelo o función predictiva $f(.)$ que predice una etiqueta y dada una instancia de x. Entonces, una tarea queda definida como $T = \{y, f(.)\}$ y puede ser aprendida de los datos de entrenamiento que consisten en un conjunto de pares $\{x_i, y_i\}$ donde $x_i \in X$ e $y_i \in Y$ (ver Fig. 1).

A partir de un dominio D_s y una tarea T_s, un dominio objetivo D_T y una tarea de aprendizaje T_T , Transfer Learning pretende ayudar a mejorar el aprendizaje de la función predictiva objetivo $f(.)$ en D_T usando el conocimiento en D_s y T_s, donde $D_s \neq D_T$ o $T_s \neq T_T$.

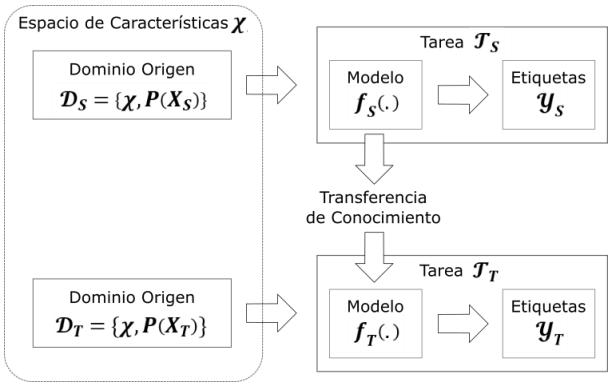

Fig. 1. Ejemplo de Transfer Learning donde los Dominios comparten el Espacio de Características, pero tienen distintas distribuciones marginales.

2.2 Reencuadre

La posibilidad de reutilizar modelos y conjuntos de datos depende en gran medida del dominio al que éstos pertenecen. Además, los modelos más versátiles pueden ser reutilizados con mayor frecuencia, lo que ofrece ventajas significativas, tales como el ahorro en recursos de cómputo necesarios para el entrenamiento del modelo y la posibilidad de reutilizar conjuntos de datos. Esto evita la necesidad de recolectar o etiquetar nuevos datos. Estas tareas son, a menudo, costosas en términos de tiempo y recursos, lo que hace que el reúso sea una opción atractiva.

Esta versatilidad tiene relación con el tipo de problema que soluciona el modelo en cuestión y el dominio al que pertenece el dataset. Hernández-Orallo, Martínez-Usó y otros [16] proponen Reframing [16], un enfoque sistemático donde explican el proceso de preparar un modelo versátil que puede adaptarse a diferentes contextos permitiendo su reutilización. Los autores clasifican la forma en que se puede reutilizar el modelo en tres tipos: a) *reentrenamiento del modelo*, donde se reutiliza el modelo y pueden reutilizarse algunos parámetros, pero se reentrena con un nuevo conjunto de datos; b) *revisión del modelo*, que trata cambios en partes de éste según un nuevo contexto y c) *reencuadre*, que muestra cómo puede reformularse un problema desde un contexto de entrenamiento para que se adapte a un contexto de producción. Luego, se clasifican los tipos de reencuadre que pueden implementarse, dependiendo de si es necesario reformular las entradas, las salidas o la estructura del modelo a ser reutilizado.

La Figura 2 muestra el proceso de reúso con reencuadre de la entrada en el contexto de producción. Esto quiere decir que, de alguna forma, se reutiliza el modelo sin reformularlo, entrenado con un dataset distinto al que se utilizará en contexto de producción. En este tipo de reencuadre es necesario reformar las entradas en producción adaptando la estructura de datos que se utiliza para representar las entradas, haciendo que sea similar a la utilizada durante la fase de entrenamiento.

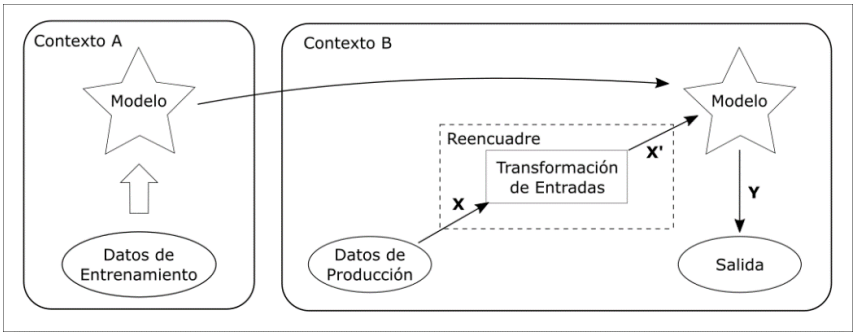

Fig. 2. Reencuadre de entrada aplicando transformación de características en contexto de producción [16].

3 Solución propuesta

En este proyecto se utiliza una red neuronal convolucional con un buen rendimiento conocido en la tarea requerida [17] y ésta se entrena reutilizando un dataset de dígitos numéricos provenientes de la literatura [3]. Con este modelo entrenado en un determinado contexto de aprendizaje, se pretende realizar pruebas en un contexto de producción. Si bien ambos contextos tienen como entradas imágenes con dígitos numéricos, estas imágenes difieren en estructura, dimensión y proporciones. Esto deriva en la necesidad de adaptar las entradas en contexto de producción al formato que se utilizó para las entradas en contexto de entrenamiento. Se puede concluir que, si bien se transfiere aprendizaje adquirido con un conjunto de datos para ser aplicado en otro contexto y con otros datos, es importante citar que el modelo no se modifica ni ajusta en la transferencia. El principal esfuerzo está en el reencuadre de las entradas del dataset de producción que tendrán que adaptarse al formato de las entradas del dataset de entrenamiento. En la Figura 3 puede observarse el diagrama de flujo de la estrategia aplicada, que tiene como principal ventaja que no es necesario confeccionar un dataset etiquetado para el entrenamiento, lo que implica un ahorro significativo de esfuerzo.

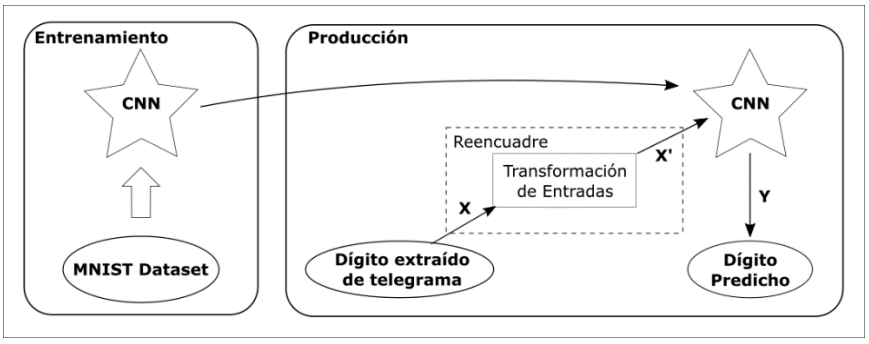

Fig. 3. Reencuadre de entrada aplicado en nuestra propuesta.

Para la generación del conjunto de datos sobre los que se iba a aplicar el modelo reutilizado se utilizó un conjunto de telegramas electorales de años anteriores, completados con cifras manuscritas y digitalizados. La estructura fija que se repite en estos documentos facilitó la extracción de las imágenes de números, aplicando máscaras de corte de forma semiautomática (Fig. 4).

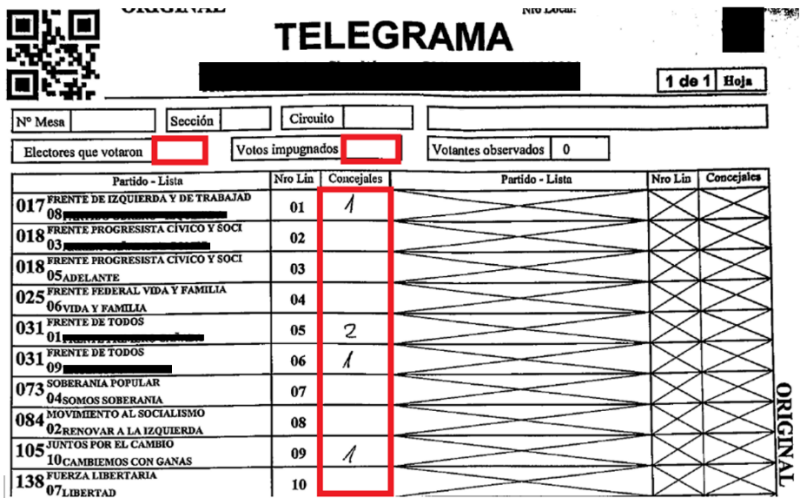

Fig. 4. Recorte de telegrama donde se aprecian celdas que se completan manualmente y repiten su disposición en todos los documentos, permitiendo automatizar la extracción.

3.1 Generación del dataset de números de uno a tres dígitos

Las celdas extraídas de los documentos electorales fueron revisadas y se seleccionó manualmente un lote con el fin de no incluir las que posean marcas no deseadas, las que no contenían números o las que estaban muy desplazadas debido al corte automático de la misma. También se excluyó las que poseían tachones o manchas que impedían obtener la imagen del número. Finalmente, se contó con un conjunto de 10000 imágenes de números desde el 0 al 303, este rango de valores es particular del dominio, ya que se trata de cifras que representan conteos de votos. Para estas imágenes se pudieron extraer las etiquetas de la base de datos donde se cargaron oportunamente en forma manual los telegramas. Este primer conjunto de datos se utilizaría en las pruebas de imágenes con números de una a tres cifras (Fig. 5).

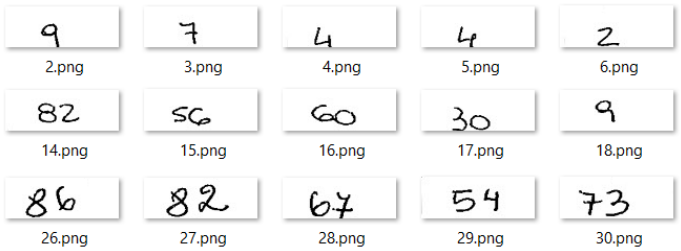

Fig. 5. Ejemplos de imágenes incluidas en el dataset de números de múltiples dígitos.

3.2 Generación del dataset de dígitos

Luego, con la aplicación de herramientas de procesamiento de imágenes provistas por la librería Open CV [18], se extrajeron los dígitos individuales de los números. Éstos fueron revisados para eliminar marcas incorrectas, dígitos pegados y manchas que no representaban un dígito válido. A continuación, se reencuadraron las imágenes de dígitos para que tengan el mismo formato -en cuanto a dimensión y color- que el dataset MNIST [3] utilizado para entrenar originalmente el modelo, como se detalla en la Sección 3.4. Por último, el etiquetado de los dígitos se extrajo de las etiquetas existentes de los números cuyos dígitos fueron extraídos (Fig. 6).

El resultado de esta etapa fue un conjunto de 13970 imágenes de dígitos y sus correspondientes etiquetas.

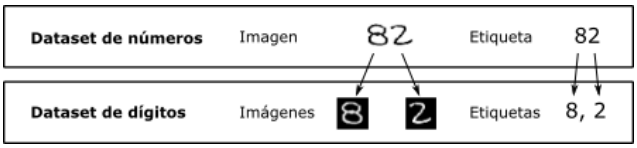

Fig. 6. Generación automática del dataset de dígitos.

3.3 Modelo y dataset reutilizado

Para contar con el modelo a reutilizar, se implementó en Python [19] una red convolucional que proviene de la documentación oficial de la librería Keras [17], en esta se describe la exactitud lograda por el modelo con el conjunto de datos de dígitos MNIST cercana al 99% y se publica la definición del mismo que consiste en múltiples capas convolucionales y de activación para finalizar en una capa de salida con 10 unidades, una por dígito a reconocer, con una función de activación Softmax [20]. En la Figura 7 se puede observar la estructura del modelo seleccionado que posee 34826 parámetros entrenables.

```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 26, 26, 32)        320
 max_pooling2d (MaxPooling 2D)  (None, 13, 13, 32)      0
 conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
 max_pooling2d_1 (MaxPooling 2D) (None, 5, 5, 64)       0
 flatten (Flatten)            (None, 1600)              0
 dropout (Dropout)            (None, 1600)              0
 dense (Dense)                (None, 10)                16010
=================================================================
Total params: 34,826
Trainable params: 34,826
Non-trainable params: 0
_________________________________________________________________
```

Fig. 7. Estructura del modelo seleccionado.

El dataset MNIST [3] (Fig. 8) empleado para el entrenamiento original del modelo comprende 70000 imágenes etiquetadas de dígitos numéricos, que se dividen en dos particiones, una de 60000 ejemplos para entrenamiento y otra de 10000 ejemplos para test.

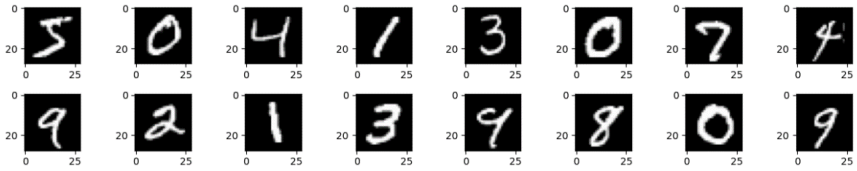

Fig. 8. Ejemplos de dígitos incluidos en el dataset MNIST.

El modelo descripto entrenado se evaluó con la partición de prueba del dataset MNIST. Las métricas de evaluación arrojaron muy buenos resultados, tal como informaba la fuente de la que se obtuvo [17]. La Tabla 1 muestra estos resultados, donde todos los valores se obtienen calculando el promedio de la medida sobre las diez clases. Se guardó el modelo con los parámetros entrenados para su evaluación con el dataset generado a partir de los telegramas en las etapas descriptas en las secciones 3.1 y 3.2.

Tabla 1. Resultados del modelo con la partición de prueba del conjunto de datos MNIST.

Métrica	Resultado
Precisión	0,99
Recall	0,99
F1-Score	0,99

3.4 Reencuadre de las entradas

Como se ha mencionado, el esfuerzo en la propuesta de reúso se centra en el reencuadre de las entradas en la etapa de producción, para que éstas se adecuen en forma y preprocesamiento a las utilizadas durante el entrenamiento.

El objetivo final del encuadre es que las imágenes de dígitos a clasificar deben ser matrices de 28 por 28 pixeles en un solo color. Esta transformación se logró en cuatro etapas, haciendo uso de la librería OpenCV [18]. Al inicio se transforma la imagen del dígito capturado a un solo color, luego se controlaron las medidas de la imagen y se rellenó en la dimensión que sea necesario para que sea una imagen cuadrada. Al obtener la imagen cuadrada, se le adicionó como relleno un margen de similares dimensiones al del dataset MNIST y, por último, a la imagen en un color, cuadrada y con los márgenes correspondientes se la redimensionó para que tenga 28 x 28 píxeles y se invirtieron los colores. Un resumen de este proceso de reencuadre puede verse en la Figura 9.

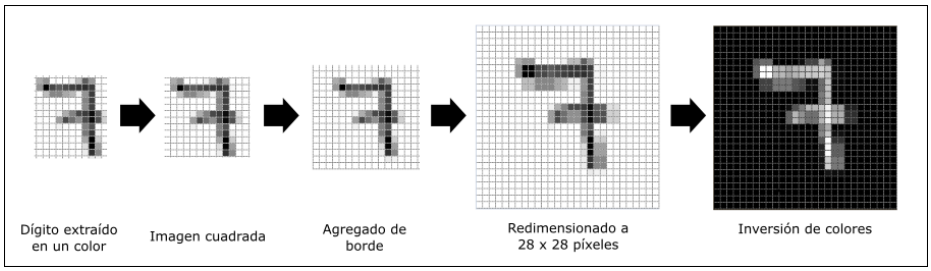

Fig. 9. Etapas del proceso de Reencuadre de las entradas.

4 Pruebas y evaluación del prototipo

4.1 Evaluación para múltiples dígitos

Para evaluar el modelo entrenado con números de una a tres cifras se utilizó el conjunto de datos obtenido en la etapa descripta en la sección 3.1. Como se citó anteriormente, este conjunto de datos posee 10000 imágenes de números desde el 0 al 303. La distribución dispar -que puede observarse en la Figura 10- es propia del dominio, notándose que son mucho más comunes las celdas con valores bajos en este tipo de formularios.

Para la evaluación del modelo con este conjunto de datos se implementó un prototipo que extrae los dígitos individuales que forman el número, los clasifica y luego, con los valores reconocidos de los dígitos, conforma el número completo. Para la evaluación de esta estrategia, este valor conformado a partir de los dígitos reconocidos se compara con la etiqueta correspondiente.

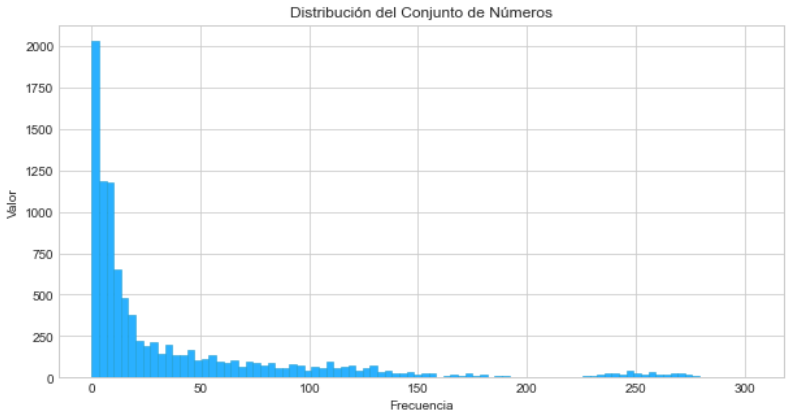

Fig. 10. Distribución del Conjunto de Números.

Debido a la gran diferencia en la cantidad de muestras de los diferentes números y a que de algunos no se poseían muestras, se realizó la evaluación del modelo sobre este conjunto de datos utilizando el porcentaje de números correctamente reconocidos como medida. Como resultado de las pruebas se obtuvo que el 69.18% de los números de

múltiples cifras fueron reconocidos correctamente. No se utilizaron otras métricas para las pruebas con números de varios dígitos, debido a la inexistencia de muestras de algunos valores y el marcado desbalance del conjunto de datos.

4.2 Evaluación para un dígito

El conjunto de datos de dígitos conseguido con el proceso descripto en la sección 3.2 consta de 13970 imágenes de dígitos del 0 al 9 con la distribución mostrada en la Figura 11, que deja ver un desbalance marcado en la cantidad de muestras para el dígito “1”. Teniendo en cuenta que el conjunto de datos no se utilizaría para entrenamiento, sino sólo para evaluar el desempeño del modelo reutilizado, se decidió continuar con todos los datos y evaluar el modelo con métricas que describan el comportamiento.

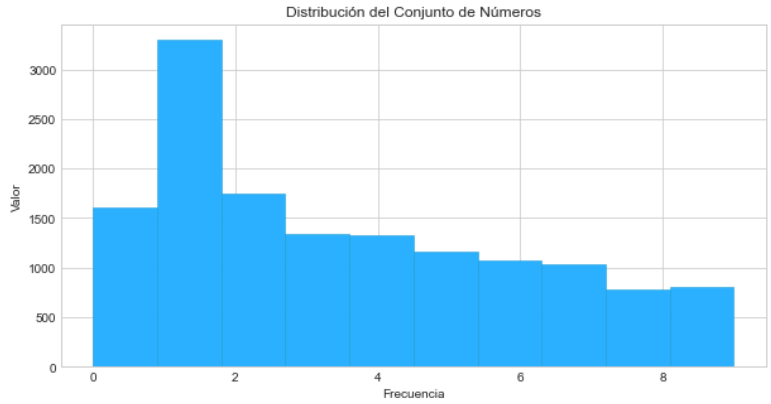

Fig. 11. Distribución del conjunto de números.

Se evaluó el modelo reutilizado con el conjunto de dígitos midiendo el porcentaje de aciertos, que arrojó una exactitud del 84.25%.

También con este conjunto de datos se evaluó el modelo utilizando otras métricas. En la Figura 12 se puede observar la matriz de confusión con los resultados del modelo vs. los valores etiquetados.

En la matriz de confusión se pueden apreciar buenos resultados de acuerdo con los valores más altos de aciertos situados en la diagonal principal, aunque también podemos reconocer una alta proporción de fallos en la clasificación del dígito “1”, el cual fue clasificado erróneamente como dígito “4” y dígito “7”. También del dígito “7”, que fue clasificado erróneamente como dígito “2”.

Debido al ya citado desequilibrio en las clases del conjunto de datos, se calcularon otras métricas para medir correctamente el desempeño del modelo. Estos resultados pueden observarse en la Tabla 2, donde todos los valores se obtienen calculando el promedio de la medida sobre las diez clases.

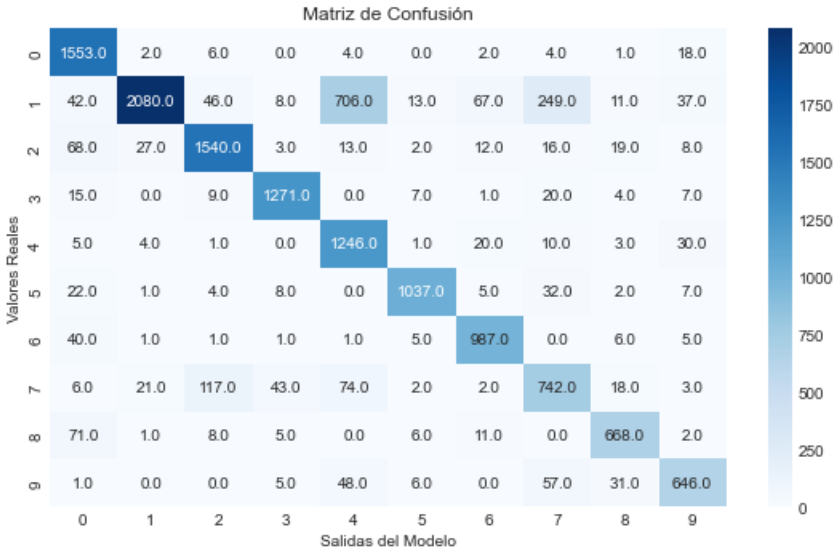

Fig. 12. Matriz de confusión con los resultados de las pruebas.

Tabla 2. Resultados del modelo con el conjunto de datos de prueba.

Métrica	Resultado
Precisión	0,848
Recall	0,868
F1-Score	0,850

4.3 Resultados

El desempeño del modelo fue evaluado por separado para los resultados obtenidos en ambas pruebas, con uno o varios dígitos. Si se observa la diferencia en la cantidad de aciertos entre ambas propuestas, y teniendo en cuenta que los dígitos son los mismos (ya que el conjunto de dígitos individuales fue extraído del conjunto de números de varios dígitos), se puede interpretar que en algunos casos el prototipo que extrae los dígitos comete errores que se reflejan en esa diferencia, constituyendo éste uno de los puntos a mejorar. Por otro lado, es errónea una clasificación de un número de varias cifras si una de ellas está mal clasificada, aunque las demás sean correctas. Entonces, la forma más adecuada de medir el desempeño del modelo reusado es con los resultados del conjunto de dígitos individuales, ya que en esa situación se elimina la posibilidad de una mala extracción, o de que predicciones incorrectas afecten otras correctas.
Si se observa el resultado de aciertos para dígitos individuales con un 84.25% y se compara con el mejor valor obtenido con la partición de Prueba del dataset MNIST, se ve que es inferior. Como se cita ut supra, la performance se ve muy afectada por algunos dígitos en los que el modelo tiene muchas predicciones incorrectas. Este comportamiento puede deberse a la diferencia en la forma de escribir algunos de estos dígitos en distintas regiones. Para este trabajo en particular existe una diferencia de dominio, entre el conjunto de datos MNIST con dígitos de origen americano y el

conjunto de datos de prueba originado a partir de documentos de nuestra región, que parece impactar en estos errores cometidos por el modelo.

5 Conclusiones y trabajos futuros

De los resultados de las pruebas pueden derivarse distintas conclusiones. Las pruebas con números de varias cifras son insuficientes teniendo en cuenta la mala distribución del conjunto de datos, ya que de algunos números no se poseen muestras y que, por razones antes descriptas, estos llegan sólo hasta el 303, aunque se haya alcanzado una exactitud del 69.18% de clasificaciones correctas, puede incrementarse el dataset como trabajo futuro para realizar nuevas pruebas. Luego, en las pruebas con el dataset de dígitos, queda como trabajo futuro analizar estrategias para solucionar algunos casos que presentaron más errores en las predicciones, como el “1”, el cual fue clasificado erróneamente como dígito “4” y dígito “7” o el “7”, que fue clasificado erróneamente como dígito “2”.

Además, queda también como trabajo futuro realizar pruebas incrementando el dataset de entrenamiento original MNIST con el nuevo dataset de dígitos aportado por este trabajo, o bien, utilizar alguna técnica de ajuste fino con el nuevo dataset sobre el modelo preentrenado. Al reentrenar el modelo con el aporte de estas nuevas muestras, se podrá evaluar posteriormente si existe alguna mejora en la precisión con un conjunto seleccionado de datos de prueba.

Con respecto a los datasets aportados por este trabajo, puede incrementarse la cantidad de muestras del dataset de números para mejorar la distribución y completitud de éste. También es necesario revisar manualmente el etiquetado del dataset de dígitos para detectar y corregir posibles errores derivados del proceso automático que lo generó.

6 Referencias

1. Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. J Big Data 3, 9 (2016). https://doi.org/10.1186/s40537-016-0043-6
2. Zhou, ZH. Learnware: on the future of machine learning. Front. Comput. Sci. 10, 589–590 (2016). https://doi.org/10.1007/s11704-016-6906-3
3. LeCun, Y., Cortes, C. and Burges, C.J.C. (1998) The MNIST Database of Handwritten Digits. New York, USA. http://yann.lecun.com/exdb/mnist/
4. Pacchiotti, M. J. (2023), “Handwritten Numbers (HN)”, Mendeley Data, V1, doi: 10.17632/b9v9x5pk96.1
5. Pacchiotti, M. J. (2023), “Handwritten Numeric Digits (HND)”, Mendeley Data, V1, doi: 10.17632/jpp6vbn9zs.1
6. Kumar, N., Gupta, M., Gupta, D. et al. Novel deep transfer learning model for COVID-19 patient detection using X-ray chest images. J Ambient Intell Human Comput 14, 469–478 (2023). https://doi.org/10.1007/s12652-021-03306-6
7. Habibi, O., Chemmakha, M. & Lazaar, M. Performance Evaluation of CNN and Pre-trained Models for Malware Classification. Arab J Sci Eng (2023). https://doi.org/10.1007/s13369-023-07608-z
8. Hasan, Rasool & Rasheed, Ali & Hashim, Ahmed & Murtdha, Mustafa. (2021). Arabic handwritten digits recognition based on convolutional neural networks with resnet-34

model. Indonesian Journal of Electrical Engineering and Computer Science. http://doi.org/10.11591/ijeecs.v21.i1.pp174-178
9. N. Aneja and S. Aneja, "Transfer Learning using CNN for Handwritten Devanagari Character Recognition," 2019 1st International Conference on Advances in Information Technology (ICAIT), Chikmagalur, India, 2019, pp. 293-296, doi: 10.1109/ICAIT47043.2019.8987286.
10. Le Zhang. 2020. A Transfer Learning Approach for Handwritten Numeral Digit Recognition. In Proceedings of the 3rd International Conference on Software Engineering and Information Management (ICSIM '20). Association for Computing Machinery, New York, NY, USA, 140–145. https://doi.org/10.1145/3378936.3378970
11. R. Caruana, D. L. Silver, J. Baxter, T. M. Mitchell, L. Y. Pratt,and S. Thrun, Learning to learn: knowledge consolidationand transfer in inductive systems, (1995).
12. Stevo. Bozinovski and Ante Fulgosi (1976). "The influence of pattern similarity and transfer learning upon the training of a base perceptron B2." (original in Croatian) Proceedings of Symposium Informatica 3-121-5, Bled.
13. Stevo Bozinovski (2020) "Reminder of the first paper on transfer learning in neural networks, 1976". Informatica 44: 291–302.
14. A. Y. Ng, “Nuts and bolts of building applications usingdeep learning,” NIPS 2016, 2016. [Online]. Available:https://www.youtube.com/watch?v=wjqaz6m42wU
15. S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEETransactions on Knowledge and Data Engineering, vol. 22,no. 10, pp. 1345–1359, Oct 2010.
16. Hernández-Orallo, J., Usó, A.M., Prudêncio, R.B., Kull, M., Flach, P.A., Ahmed, C.F., & Lachiche, N. (2016). Reframing in context: A systematic approach for model reuse in machine learning. AI Commun., 29, 551-566.
17. https://keras.io/examples/vision/mnist_convnet/
18. https://opencv.org/
19. https://www.python.org/
20. Charniak, E. (2019). Introduction to deep learning. Cambridge, Massachusetts; London, England: The MIT Press.