# Prototipo de software basado en aprendizaje profundo para mantenimiento predictivo en la Industria 4.0 - 9no. Congreso Nacional de Ingenieria Informatica y Sistemas de Información...

**Article** · November 2021

CITATIONS
0

READS
67

**2 authors:**

Mauro José Pacchiotti
National University of Technology
**5** PUBLICATIONS **2** CITATIONS
SEE PROFILE

Pablo Paletto
National University of Technology
**1** PUBLICATION **0** CITATIONS
SEE PROFILE

# Prototipo de software basado en aprendizaje profundo para mantenimiento predictivo en la Industria 4.0

**Pacchiotti, Mauro José**
**Paletto, Pablo Andrés**

***Universidad Tecnológica Nacional, Facultad Regional Santa Fe***

**Abstract**
*En la Industria 4.0, la utilización de las TICs posibilita la recolección y procesamiento de gran cantidad de datos acerca del funcionamiento de los diferentes equipamientos industriales, por lo que representa un desafío el aprovechar los conjuntos de datos de distintos sensores para tareas de mantenimiento predictivo de los dispositivos. Este trabajo propone un modelo experimental que simula la captura de datos y clasificación del estado de un dispositivo en tiempo real, utilizando datos sintéticos, datos provistos por la industria y modelos de aprendizaje profundo. Se realizaron dos experimentos con series de datos de diferente origen y características para entrenar los modelos de aprendizaje profundo, logrando una alta precisión en la predicción del estado del dispositivo en función de los valores de los sensores en los instantes previos.*

**Palabras Clave**
Industria 4.0, mantenimiento predictivo, aprendizaje profundo, redes neuronales convolucionales, simulador.

## Introducción

Desde los primeros talleres donde el hombre en grupos y con la utilización de maquinarias, comenzó a producir bienes, ocurrieron fallas en los equipamientos que necesitaron de la reparación de estos para poder continuar con la producción. Inicialmente los mismos operarios que trabajaban en los distintos procesos eran los que reparaban sus propias máquinas y herramientas, luego, las empresas comenzaron a separar las actividades de los trabajadores en dos grupos, actividades de producción y actividades de mantenimiento. En 1930 Henry Ford incorpora en su empresa el concepto de mantenimiento, inclusive, creando el puesto de director de Mantenimiento para quien estaba a cargo de gestionar estas tareas.

La llegada de la segunda guerra mundial y un incremento de la jornada laboral con líneas en que la producción no se detenía, repercutieron en un mayor deterioro de los equipamientos industriales donde las paradas de producción por fallas de las máquinas comenzaron a producir grandes pérdidas. Comienzan a realizarse tareas de prevención en el mantenimiento para reemplazar las partes que sufren desgaste en los equipamientos antes que estos fallen y de esta forma reducir los tiempos de parada de producción.

La buena aplicación de mantenimiento en la industria tiene muchas ventajas: mejora la calidad de los productos, reduce los tiempos de producción, disminuye los riesgos de accidente de trabajo, reduce el tiempo perdido en paradas imprevistas que implican pérdida de materiales, aminora los fallos irreparables en los equipamientos y mejora la previsión de los tiempos de producción favoreciendo la elaboración de presupuestos, entre otras. [1]

El mantenimiento preventivo, puede dar muy buenos resultados, aunque teniendo en cuenta que los proveedores de equipamientos industriales hacen hincapié en la confiabilidad de sus equipos, esto muchas veces impacta generando altos costos de mantenimiento.

La probabilidad de falla de componentes en los equipamientos es alta al principio y al final de su vida útil, debido a fallas en la instalación o en la fabricación del repuesto y al desgaste sufrido por la utilización de

este. Por lo tanto, las tareas de mantenimiento innecesarias aumentan la tasa de fallas cuando un elemento defectuoso se instala o cuando ocurre un error humano.
Finalmente, el mantenimiento preventivo se basa erróneamente en la idea de que la probabilidad de ocurrencia de las fallas operativas aumenta exponencialmente en un momento determinado, entonces los componentes se reemplazan o reparan antes de que ese momento ocurra. Esta suposición no es cierta en muchos casos; hay varios patrones en los que la probabilidad de falla nunca aumenta, sino que es constante en el tiempo. Entonces un componente podría fallar en cualquier momento. Ejemplos de este fenómeno son los patrones de falla de componentes eléctricos y electrónicos, donde las tareas de sustitución en períodos de tiempo planificados no implican una prueba de fiabilidad. [2]
La Industria 4.0**,** también llamada cuarta revolución industrial, es un concepto utilizado en los últimos años para describir a la industria funcionando con la colaboración de distintas nuevas tecnologías que digitalizan, interconectan y optimizan procesos de manera nunca vista. Con la llegada de Internet de las Cosas (IoT) y las comunicaciones 5G se facilita la interconexión de equipamientos sin importar la distancia entre ellos, mientras el almacenamiento en la nube permite el resguardo y alta disponibilidad de grandes cantidades datos.
Estas grandes cantidades de datos que empiezan a estar disponibles en las plantas industriales motivan la adopción de técnicas de Machine Learning para, mediante el análisis de estos datos, abordar requisitos y necesidades industriales. Para este trabajo en particular, el foco principal se pone en la predicción, es decir la capacidad de estimar y anticipar eventos sobre los activos industriales o predecir el estado del activo en un tiempo estimado.
En las próximas secciones se detalla el desarrollo del proyecto, el prototipo de software propuesto, dos conjuntos de datos de distintas fuentes y características, los modelos seleccionados para estos conjuntos de datos y las pruebas realizadas con ambos, finalizando con las conclusiones y los trabajos futuros.

**El prototipo de software**

Este trabajo propone el desarrollo de un prototipo de herramienta de software que pretende simular el proceso de recepción de datos en tiempo real de un determinado dispositivo y predecir, a partir de estos datos y los recibidos anteriormente, el estado de un determinado componente. Para el desarrollo de este prototipo se plantearon los siguientes requerimientos.
1) Un proceso que simule la emisión de lecturas con una cierta frecuencia que se ajustará de acuerdo con los parámetros reales de los equipos estudiados.
2) Un segundo proceso que capture las lecturas emitidas por el proceso simulador y a partir de estas lecturas pueda predecir y emitir un mensaje con el estado predicho del componente o equipamiento monitoreado en un cierto lapso de tiempo futuro.
3) El proceso predictor debe tener la capacidad de leer distintos modelos entrenados para distintos tipos de componentes y series de tiempo.
4) Debido a que se trata de dos procesos asíncronos, por un lado, el emisor de lecturas y por el otro en predictor de estado, es necesario la existencia de un repositorio al que ambos procesos tengan acceso.
A partir del objetivo y requerimientos antes planteados se propuso un modelo, como se puede observar en la figura 1, los procesos A y B son independientes, por lo que se deben implementar mediante programación concurrente en dos hilos distintos de ejecución.
El repositorio que comunica ambos procesos es una cola (FIFO) para mantener el orden de las lecturas en caso de demoras en el Proceso B que causen acumulación de datos en el repositorio. El repositorio debe tener implementado un mecanismo de exclusión mutua (MUTEX) para evitar

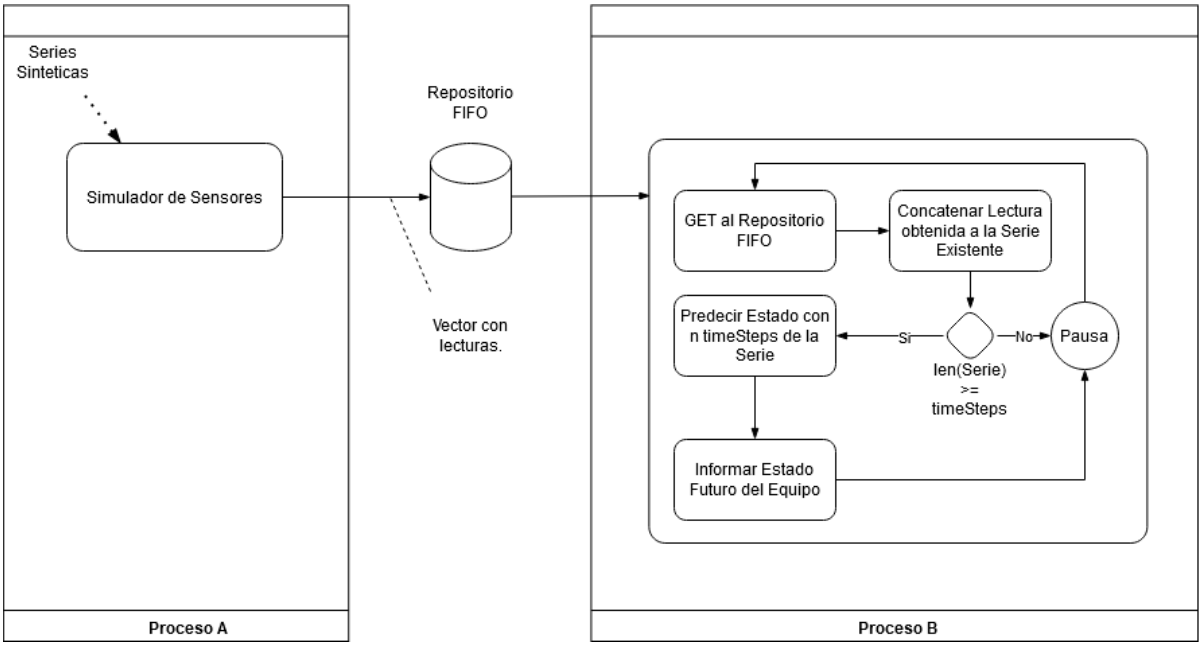


*Figura 1: Modelo propuesto del prototipo.*

conflictos en caso de acceso simultáneo de ambos procesos al mismo.

Resultando muy importante cumplir con el requerimiento que especifica un prototipo capaz de leer distintos tipos de modelos optamos por dar la capacidad al mismo de, antes de la simulación propiamente dicha, leer la estructura del modelo implementado, desde un archivo JSON, y los parámetros entrenados para el mismo desde un archivo H5(Hierarchical Data Format).

En función de las distintas estrategias propuestas para desarrollar el prototipo y la necesidad de utilizar modelos de aprendizaje profundo, se decide utilizar Python[1] como lenguaje para la implementación, en un entorno local a fin de tener en correcto control de los distintos procesos concurrentes, el repositorio y mecanismos de exclusión mutua.

En la Figura 2 se presenta el diagrama de flujo del prototipo implementado. En primer lugar, se leen las series de datos que alimentan el proceso simulador. Luego, se leen los datos necesarios para el proceso predictor (modelo, parámetros entrenados y parámetros de normalización). Una vez obtenidos todos los datos necesarios para ambos procesos, se inician operando de forma totalmente asíncrona. Mientras el proceso simulador se encarga de depositar en el repositorio de intercambio FIFO un vector de lecturas por cada paso de tiempo, el proceso predictor está constantemente chequeando el repositorio. Si este encuentra un nuevo vector de lecturas, lo concatena a los ya recibidos, toma la cantidad necesaria de últimas lecturas (si ya dispone de esa cantidad) y realiza la predicción.

Para finalizar el proceso predictor, entrega el resultado de la predicción que puede ser en una gráfica, en texto por pantalla y/o guardada en un archivo con el historial de predicciones.

**Conjuntos de datos**

Por tratarse de un proyecto basado en datos, inicialmente se dedicaron todos los esfuerzos a la búsqueda de conjuntos de datos que hayan sido utilizados y validados en investigaciones anteriores y cuenten con el etiquetado necesario. Luego de avanzado el proyecto en este sentido, se consiguió un dataset anonimizado provisto por la industria con el que también se realizaron pruebas con el prototipo.

A continuación, se detallan la obtención y preparación del conjunto de datos de la

[1] *https://www.python.org/*

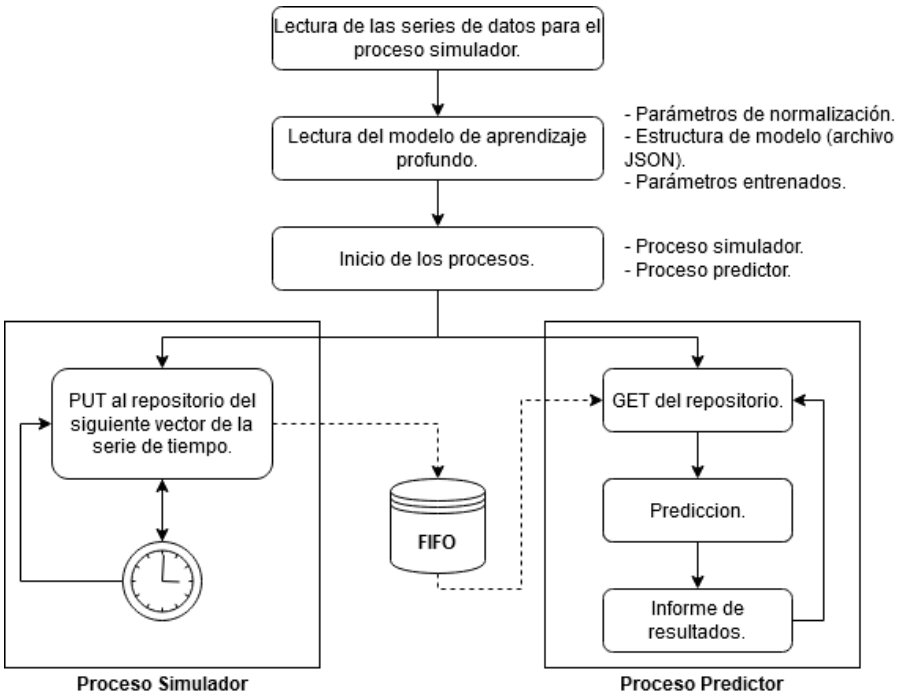


*Figura 2: Diagrama de flujo del prototipo.*

literatura y luego del provisto por la industria.

**Conjuntos de datos de la literatura**

La búsqueda de conjuntos de datos para el proyecto se realizó principalmente en dos repositorios de la Universidad de California[2][3] y en la plataforma Kaggle[4].

Encontramos tres conjuntos de datos que se ajustaban a las necesidades del proyecto, dos de ellos con series de tiempo sobre mediciones de ruido de un motor y una clasificación binaria acerca del estado de este y un tercer dataset con series de tiempo que provienen de distintos sensores ubicados en una plataforma de pruebas hidráulicas y las etiquetas con los estados de cuatro diferentes componentes luego de transcurrido el tiempo de las mediciones reportadas por los sensores.[3]

Luego de la lectura de la documentación y fuentes de los conjuntos de datos encontrados en los diferentes repositorios decidimos comenzar a realizar pruebas con el dataset de Monitoreo de Plataforma Hidráulica de Pruebas [3] antes descripto, debido a que tiene mayor cantidad de muestras, de atributos y componentes con condición etiquetada.

Estos datos, atributos y objetivos monitoreados nos permitieron realizar una mayor cantidad de pruebas exploratorias sobre el conjunto de datos con el fin de encontrar subconjuntos de atributos y etiquetas para analizarlos y probarlos en un modelo de aprendizaje profundo.

El sistema repite ciclos de carga constante (durante 60 segundos), en cada ciclo realiza mediciones con distintas frecuencias y al finalizar este registra la condición de cuatro componentes.

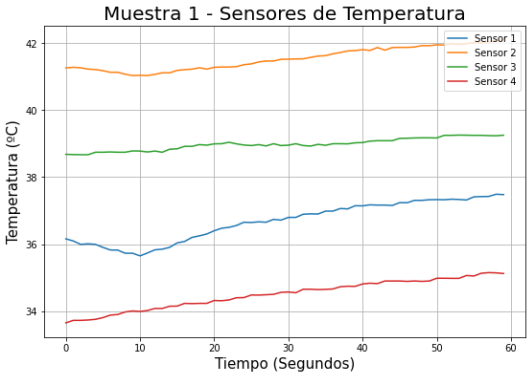


*Figura 3: Gráfica de una serie de tiempo*

[2] *http://www.timeseriesclassification.com*
[3] *https://archive.ics.uci.edu/ml/index.php*
[4] *https://www.kaggle.com/*

El conjunto de datos tiene en total 2205 muestras, por cada muestra los 17 sensores generan series de datos a distinta frecuencia en la entrega de lecturas.
Se realizó un análisis exploratorio de datos sobre el dataset, graficando las distintas series de tiempo (Figura 3) y calculando el balance para las distintas etiquetas de salida.
A partir del análisis exploratorio de datos y teniendo en cuenta, que tipos de sensores se correlacionan con los distintos componentes, se seleccionaron cuatro subconjuntos de datos (Tabla 1) para realizar pruebas con el modelo de aprendizaje profundo.

| **Sensores** | **Etiqueta** | **Denominación** |
|---|---|---|
| Flujo (2) | Fuga de bomba | FS1-2_Leaking |
| Presión (6) | Fuga de bomba | PS1-6_Leaking |
| Presión (6) | Estado de válvula | PS1-6_Valve |
| Temperatura (4) | Estado de cooler | TS1-4_Cooler |

*Tabla 1: Subconjuntos de datos seleccionados.*

## Modelo para el dataset de la literatura

La selección del modelo a utilizar depende de diferentes factores: la naturaleza de los datos, la existencia de conjuntos de datos etiquetados, la cantidad de datos disponibles, entre otros. Las fuentes académicas de datos utilizadas en el trabajo constan de sensores dispuestos en diferentes partes de un equipo que reportan una medición con cierta periodicidad estable y conocida, por lo tanto, estas fuentes tienen la forma de series de tiempo. Esta característica hace necesaria la existencia de etiquetas con el estado del componente o equipo que se está censando para poder entrenar el modelo de aprendizaje profundo. Dentro de los modelos que permiten trabajar con Series de Tiempo existen dos grandes grupos: Redes Neuronales Recurrentes y Redes Neuronales Convolucionales [4].
Para la realización de este trabajo se optó por modelos convolucionales teniendo en cuenta que el objetivo del modelo de aprendizaje profundo sería reconocer diferentes características en las señales que permitan clasificar un estado del equipamiento o componente en el futuro. Se utilizó inicialmente un modelo convolucional 1D simple (Tabla 2), la estructura de este se extrajo y ajustó de la documentación del framework [5] seleccionado para implementar el modelo.

| |
|---|
| Capa convolucional 1D con 100 filtros de tamaño 6 y función de activación Relu. |
| Capa convolucional 1D con 100 filtros de tamaño 6 y función de activación Relu. |
| Capa MaxPooling 1D con tamaño de ventana 3. |
| Capa convolucional 1D con 100 filtros de tamaño 6 y función de activación Relu. |
| Capa convolucional 1D con 100 filtros de tamaño 6 y función de activación Relu. |
| Capa GlobalAveragePooling1D. |
| Capa Dropout con una tasa de 0,5. |
| Capa de salida, completamente conectada, con tres unidades de salida, una por cada clase y función de activación Softmax. |

*Tabla 2: Capas del modelo utilizado.*

Para el entrenamiento del modelo se utilizó entropía cruzada [6] como función de pérdida y Adam [7] como algoritmo de

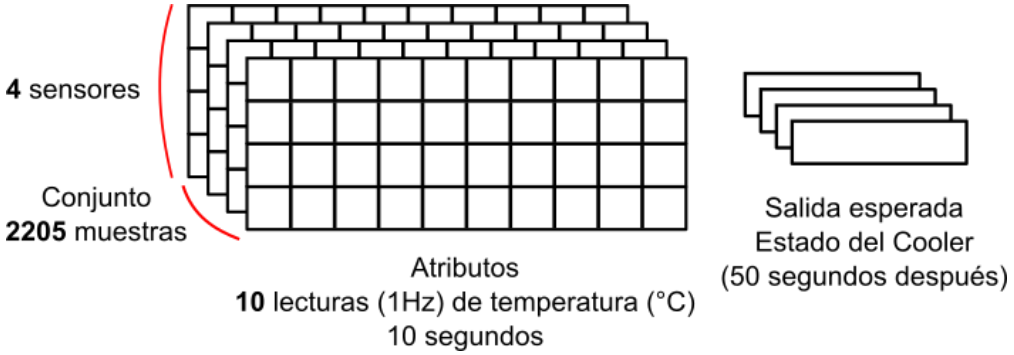


*Figura 4: Estado final de la serie de tiempo.*

optimización. Los resultados del entrenamiento con los conjuntos de datos definidos son los siguientes: Para el conjunto FS1-2_Leaking el modelo logró una precisión del 72,51%. Para el conjunto PS1-6_Leaking se obtuvo una precisión del 53,32%. Con el conjunto PS1-6_Valve el resultado obtenido fue de una precisión del 100%. Y por último con el conjunto TS1-4_Cooler, el modelo arrojó una precisión del 99,4%. De estos resultados podemos concluir que las series de tiempo obtenidas a partir de los sensores de flujo y presión no explican el estado de la bomba y su fuga. Mientras que las series de tiempo de los sensores de presión tienen correlación con el estado de la válvula y las de los sensores de temperatura tienen correlación con el estado del enfriador. Por los requisitos para realizar las pruebas y validación del prototipo junto a los resultados obtenidos se decidió continuar trabajando con el conjunto TS1-4_Cooler.

Sobre el conjunto de datos finalmente seleccionado se realizó una reducción de la serie original a fin de obtener una ventana de tiempo entre la predicción y el estado predicho del componente (Figura 4), quedando series de 10 pasos de tiempo y la etiqueta de salida con el estado del componente monitoreado 50 segundos después con codificación One-hot. Estas modificaciones repercutieron en la estructura del modelo (Tabla 3), donde se

| |
|---|
| Capa convolucional 1D con 100 filtros de tamaño 6 y función de activación Relu. |
| Capa convolucional 1D con 100 filtros de tamaño 3 y función de activación Relu. |
| Capa GlobalAveragePooling1D. |
| Capa Dropout con una tasa de 0,5. |
| Capa de salida, completamente conectada, con tres unidades de salida, una por cada clase y función de activación Softmax. |

*Tabla 3: Capas del modelo final.*

mantuvieron las funciones de perdida y optimización del modelo anterior. Luego de los cambios realizados se alcanzó una precisión del 99,3%. Si bien la precisión alcanzada es satisfactoria, se continuó con el análisis del modelo para optimizar su estructura y así evitar sobredimensionarlo. Para esto es necesario ajustar de alguna manera los hiperparámetros del mismo. En nuestro caso se utilizaron los conceptos de "Optimalidad de Pareto" un método de optimización multiobjetivo [8]. Los objetivos de optimización fueron el tiempo que demora el modelo en realizar una predicción y la precisión alcanzada durante el proceso de entrenamiento. Luego de realizado el procedimiento se obtuvieron los hiperparámetros para el modelo final (Tabla 4).

| | |
|---|---|
| Tamaño 1er. ventana convolucional | 6 |
| Filtros de la 1er. capa convolucional | 60 |
| Tamaño 2da. ventana convolucional | 4 |
| Filtros de la 2da. capa convolucional | 60 |
| Tasa de la capa Dropout | 0.6 |

*Tabla 4: Valores de los hiperparámetros luego de la optimización.*

**Conjunto de datos de la industria**

El dataset provisto por la industria consiste en una serie de 912 elementos, cada uno de los cuales contiene la lectura entregada por 119 sensores de flujo cada media hora y una columna denominada "TARGET" que es la etiqueta de salida.

Como datos de la etiqueta solo se aclara que esta identifica un estado crítico cuando su valor decrece significativamente. A fin de reflejar un estado binario que indique si se presenta una falla o no, se agregó al conjunto de datos una columna denominada "LABEL(T+1)". Ésta contiene un valor 0 si el valor de la columna "TARGET" es mayor a 3.00 en el paso de tiempo siguiente (media hora después). Caso contrario, el valor de "LABEL(T+1)" es 1, indicando un valor bajo de la variable TARGET, significando una posible falla. Cabe aclarar que el valor 3 como umbral para el cambio de estado de la variable "LABEL(T+1)" se seleccionó, analizando visualmente el comportamiento de los datos.

En la figura 5 se puede observar el comportamiento de la variable TARGET y la etiqueta LABEL(T+1), encontrando en esta última un desbalance importante en la

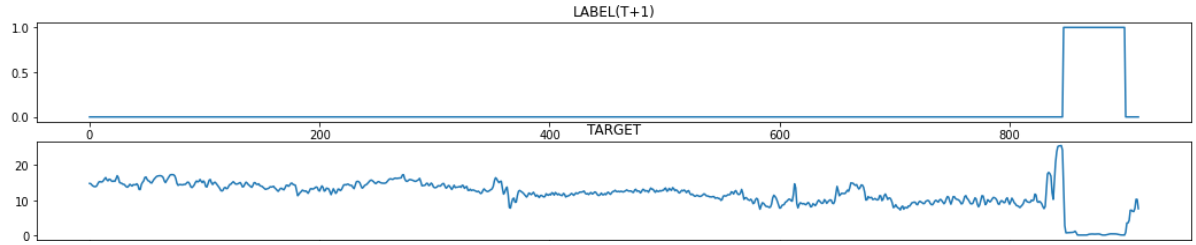


*Figura 5: Gráficas de la variable TARGET y la etiqueta LABEL(T+1)*

cantidad de casos negativos y positivos que afectarían el desempeño del modelo de aprendizaje profundo. Para solucionar el desbalance decidimos probar una estrategia básica de ataque de este problema que consta de reducir la cantidad de muestras de la clase mayoritaria a la misma cantidad de muestras de la clase minoritaria. Para este caso entonces tomamos solo desde la muestra 793 y hasta la muestra 900, de esta forma el conjunto de datos queda balanceado con 54 casos negativos y 54 positivos.

**Modelo para el dataset de la industria**

En contra parte a los datos académicos, el conjunto de datos provisto por la industria consta de una sola serie de datos, por lo que no se creyó conveniente el uso de una red neuronal convolucional debido a la escasa cantidad de datos, por lo que se implementó un modelo de red neuronal Perceptrón Multicapa [9]. Para optimizar el modelo se utilizó nuevamente el procedimiento de Optimalidad de Pareto. Como resultado de esto se obtuvo un modelo Perceptrón Multicapa que consta de dos capas ocultas, la primera de 140 unidades y la segunda de 50 unidades. Las funciones de optimización y pérdida se conservaron del modelo explicado anteriormente. Debido a la poca cantidad de datos del conjunto, para entrenar el modelo no se utilizó una partición de validación, sino que las 108 muestras formaron el conjunto de entrenamiento mientras que para test se utilizó el conjunto completo de 912

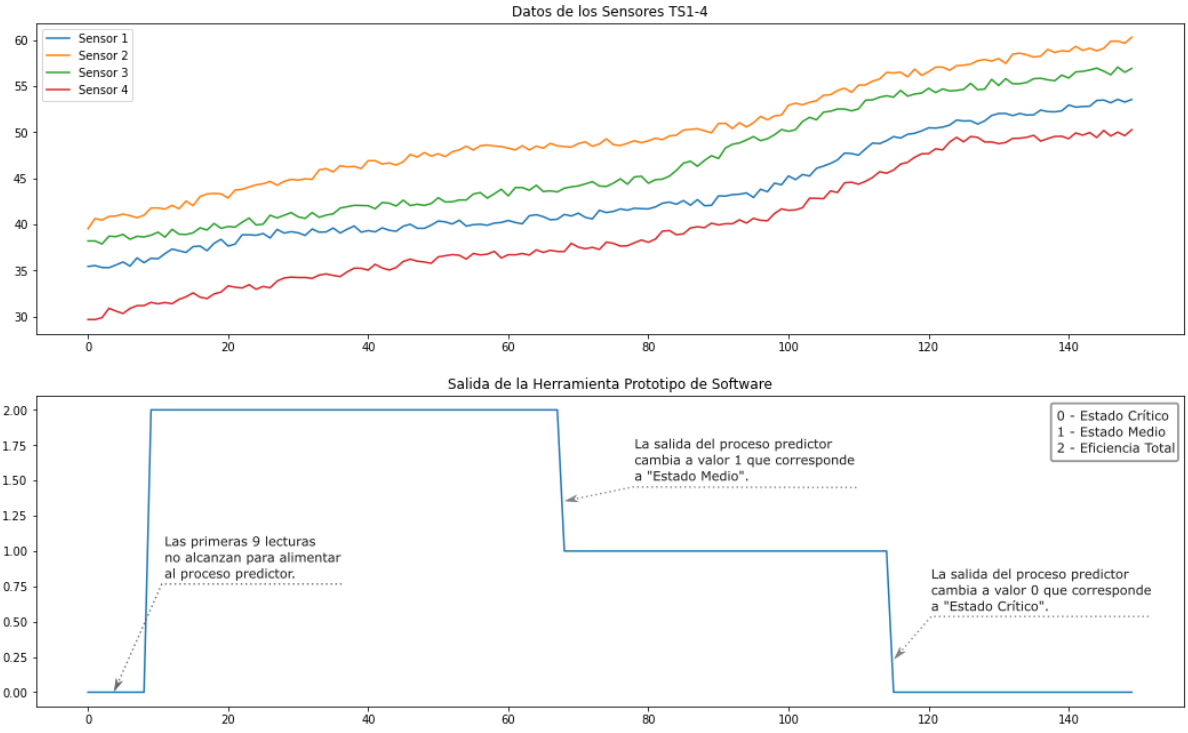


*Figura 6: Arriba las series sintéticas que alimentan el proceso simulador y abajo las salidas del proceso predictor.*

muestras. La precisión alcanzada con el total del conjunto de datos fue de 97.81%.

**Pruebas con el dataset de la literatura**

Para las pruebas con el conjunto de datos de la literatura se alimentó el proceso simulador del prototipo con series de datos sintéticas. Estás se confeccionaron a partir del análisis de datos realizado para que simulen el comportamiento de un equipo que comienza a operar con bajas temperaturas y estas se incrementan con el paso del tiempo. De esta forma la etiqueta de estado del componente a medida que pasa el tiempo debería cambiar de "Eficiencia Total" a "Estado medio" y por último a "Estado Crítico".

En la figura 6 se puede observar un buen comportamiento del proceso predictor. En principio no realiza predicciones hasta tener 10 lecturas almacenadas, ya que esta es una necesidad del modelo convolucional. Al comienzo las predicciones tienen valor 2, que corresponde a la etiqueta "Eficiencia Total". Luego, al incrementarse los valores de temperatura podemos apreciar que, a partir de la predicción 69 la salida cambia a valor 1, que corresponde a la etiqueta "Estado Medio". Y a partir de la predicción 115, la salida tiene valor 0, que corresponde a la etiqueta "Estado Crítico". Entonces, la salida predicha varía para el estado del COOLER a medida que los valores reportados por los sensores aumentan.

**Pruebas con el dataset de la industria**

Las pruebas en este caso se realizaron con el modelo resultante de la optimización de Pareto en el proceso predictor. Ante la falta de datos y la complejidad de elaborar series sintéticas con 119 variables que se comportan de forma totalmente distinta, se utilizó para alimentar el proceso simulador la serie completa de datos provista, incluso con los datos iniciales que no se utilizaron en el entrenamiento. De esta forma se obtuvo una serie de 912 pasos de tiempo, una longitud suficiente para las pruebas en el prototipo.

Los resultados fueron muy satisfactorios, más aun teniendo en cuenta lo reducido del conjunto de datos provisto. Se puede observar en la gráfica superior de la figura 7 que el valor de la variable TARGET desciende por debajo de 3 en el paso de tiempo 848, por lo que se observa en la segunda gráfica de la figura 7 que la variable LABEL(T+1) cambia su valor de 0 a 1 un paso de tiempo antes, es decir en el paso 847. Mientras podemos apreciar en la gráfica inferior de la figura 7 que el prototipo comienza a informar un estado futuro 1 qué significa "ESTADO CRÍTICO" a partir del paso de tiempo 831, es decir 16 pasos de tiempo antes de que los valores sean críticos. Teniendo en cuenta que el paso de tiempo para este conjunto de datos es de 30 minutos entre muestras, el prototipo está prediciendo una falla 8 horas antes en este caso.

**Conclusiones y Trabajos Futuros**

Como conclusión de este Proyecto Final de Carrera podemos asegurar que se han cumplido los distintos objetivos propuestos a lo largo del mismo, desde la obtención de distintos conjuntos de datos que permitan realizar los análisis y pruebas necesarios para asegurar su utilidad en el trabajo, y la selección de modelos de aprendizaje profundo que una vez optimizados han permitido una alta precisión en las predicciones.

Finalmente, el objetivo principal se cumple con creces dados los muy buenos resultados obtenidos a partir de las pruebas con el prototipo planteado, tanto en el conjunto de datos obtenido de la literatura como en el obtenido de la industria, más aun teniendo en cuenta las grandes diferencias que existen entre estos conjuntos de datos y modelos.

Estos resultados obtenidos confirman que el modelo de prototipo planteado no solo realiza correctamente las tareas de simulación y predicción, sino que también puede leer distintos tipos de modelos para implementarlos con distintos conjuntos de datos, y de esta manera reutilizar la

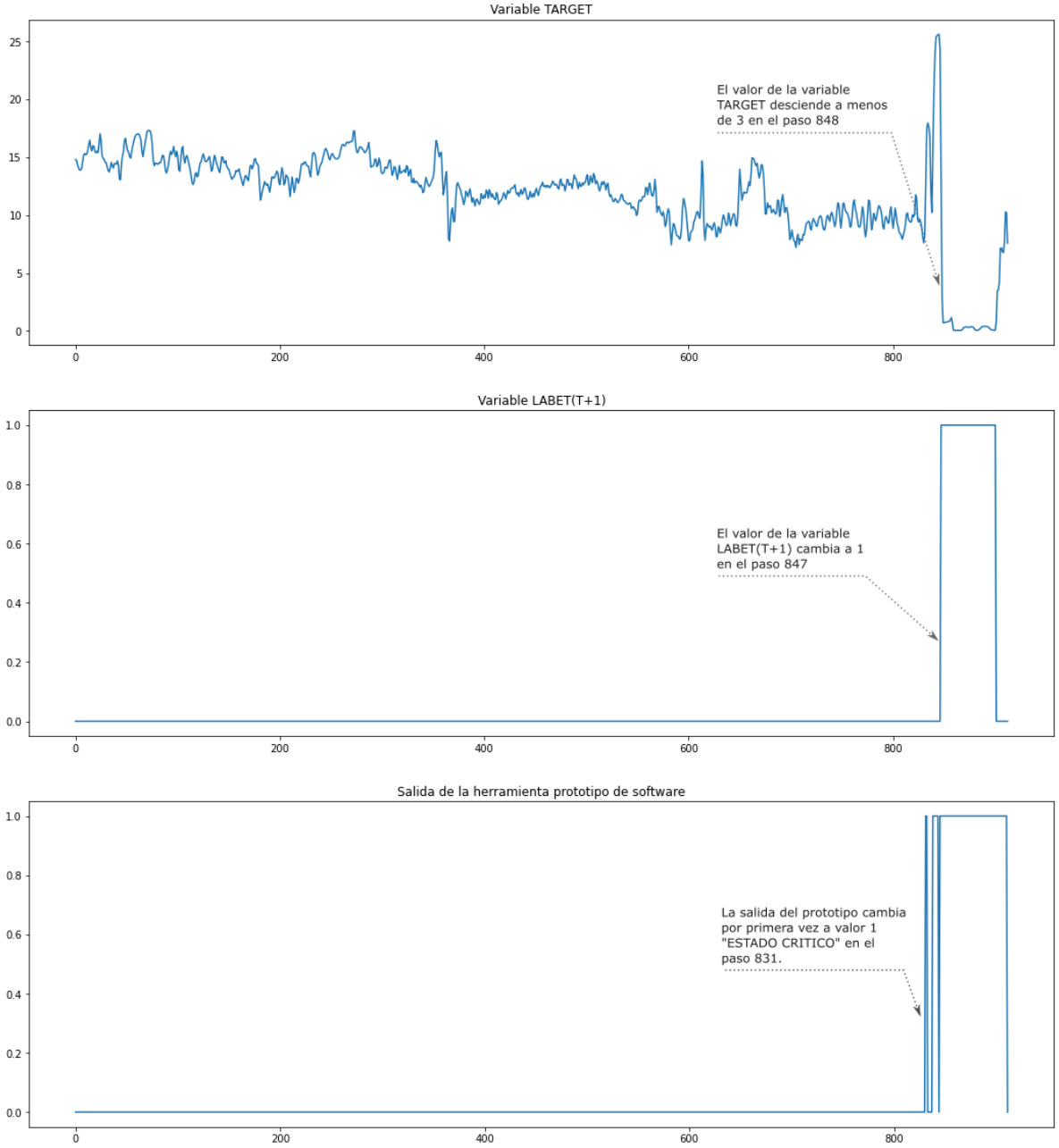


*Figura 7: Gráfica de la serie provista, la variable TARGET y la salida del prototipo.*

herramienta reduciendo los tiempos de prueba. Los formatos de archivo utilizados, JSON y H5, para almacenar tanto los modelos como los parámetros entrenados son ampliamente utilizados por lo que se pueden realizar las pruebas, ajustes y entrenamiento de modelos con una gran variedad de herramientas que luego permiten generar estos archivos para ser interpretados por el prototipo.

Durante el desarrollo de este trabajo fueron surgiendo varias ideas para proyectos futuros. El principal trabajo futuro comprende la continuación y mejora del prototipo de herramienta de software ya que este trabajo es una versión inicial. Las próximas versiones deben comprender el agregado de interfaces que permitan una mejor experiencia del uso tanto en la configuración de lo que comprende al modelo y conjuntos de datos, como así también a la entrega de informes y gráficas de forma sencilla.

También es necesario continuar las pruebas con otros datasets a fin de seguir evaluando el desempeño del prototipo en diferentes

situaciones, en el caso de las pruebas con series de datos sintéticas se pueden simular distintos tipos de fallas en sensores o la inclusión de ruido en las señales para evaluar el comportamiento y el resultado de las predicciones en estos casos.

**Agradecimientos**
A nuestro Director de Proyecto Final de Carrera Dr. Mariano Rubiolo, por su compromiso con nuestro trabajo y constante acompañamiento durante el desarrollo del mismo. Este trabajo se realizó en el marco del PID SIUTNFE0007748 (2020), bajo la dirección del Dr. Rubiolo.

**Referencias**

[1] Olarte, W. (2010). Importance of the Industrial maintence inside the processes of production. *Scientia Et Technica*, *44*, 354–356.

[2] Diez-Olivan, A., Del Ser, J., Galar, D., & Sierra, B. (2019). Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0. *Information Fusion*, *50*, 92–111. https://doi.org/10.1016/j.inffus.2018.10.005

[3] N. Helwig, E. Pignanelli and A. Schütze, "Condition monitoring of a complex hydraulic system using multivariate statistics," 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, Pisa, Italy, 2015, pp. 210-215, doi: 10.1109/I2MTC.2015.7151267.

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

[5] Team, K. *Keras documentation*. Keras. https://keras.io/.

[6] Zhang, Z., & Sabuncu, M. R. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*.

[7] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[8] Ngatchou, P., Zarei, A., & El-Sharkawi, A. (2005, November). Pareto multi objective optimization. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems* (pp. 84-91). IEEE.

[9] Haykin, S. S. (2009). *Neural networks and learning machines*. Upper Saddle River, NJ: Prentice Hall.