UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

UN MODELO GENERATIVO RECURRENTE PROFUNDO SEMI-SUPERVISADO PARA LA ESTIMACIÓN DEL TIEMPO DE VIDA REMANENTE EN ACTIVOS FÍSICOS

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA

IGNACIO NICOLÁS MARTÍNEZ SALAZAR

PROFESOR GUÍA:
ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:
VIVIANA MERUANE NARANJO
JUAN TAPIA FARIAS

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MECÁNICA
POR: IGNACIO NICOLÁS MARTÍNEZ SALAZAR
FECHA: 2019
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

UN MODELO GENERATIVO RECURRENTE PROFUNDO SEMI-SUPERVISADO PARA LA ESTIMACIÓN DEL TIEMPO DE VIDA REMANENTE EN ACTIVOS FÍSICOS

Las estrategias tradicionales de mantenimiento, como el mantenimiento correctivo de la falla imprevista de equipos y el mantenimiento programado, se están volviendo menos capaces de satisfacer la creciente demanda industrial en el área de confiabilidad, debido de que requiere establecer modelos de ecuaciones explicitas y un gran conocimiento acerca de técnicas de procesamiento de señales y experiencia necesaria en pronostico de fallas. Los recientes desarrollos en modelos de aprendizaje profundo han brindado una oportunidad para crear métodos avanzados para la predicción del RUL (Remaining Useful Life) para big data, esto es gracias a su capacidad de procesamiento de datos y extracción de características debido a sus múltiples estructura de capas.

El objetivo general es desarrollar un modelo generativo de aprendizaje profundo semi-supervisado para la predicción de la vida remanente de equipos. Los objetivos específicos son: Definir el modelo generativo a utilizar. Explorar, analizar e identificar la arquitectura de una red neuronal recurrente óptima. Desarrollar una función de pérdida para entrenar el modelo. Implementar el modelo a través de un código computacional en la librería de TensorFlow. Comparar el modelo propuesto con el estado del arte para modelos que realizan la misma tarea.

La metodología a seguir consta de tres etapas, la primera consiste en un pre-procesamiento de los datos, la segunda en generar el modelo y emplearlo con un conjunto de datos con el objetivo de extraer su espacio latente y a su vez generan etiquetas para éstos. Por último, se hace la predicción de la vida remanente de los equipos utilizando el espacio latente obtenido y las etiquetas generadas para un conjunto de datos de validación.

Para entrenar el modelo se propone el conjunto de datos C-MAPSS, el desempeño de este proceso es medido mediante la función *Root Mean Squared Error* y una función de cuantificación de error. Los resultados obtenidos son comparables con el estado del arte en el área, haciendo la salvedad de que los modelos presentados por [4] y [24] son supervisados, mientras que la MRLSTM-GANs es semi supervisada. Desde el punto de vista de la industria, recopilar datos hasta la falla tiene altos costos asociados, tanto por los equipos de medición como por los activos que resultan dañados. Es en este sentido, el modelo propuesto presenta una ventaja, ya que no es necesario tener una gran cantidad de mediciones hasta la falla para realizar las predicciones al ser de aprendizaje semi supervisado. Permitiendo de esta manera reducir gastos y el tiempo empleado en la obtención de datos, la cual puede llegar a durar años.

Agradecimientos

Agradecimiento a programa 2 “geometallugical modeling and mine planning del proyecto CSIRO Chile international Centre of excellence in Mining and Mineral Processing", Codigo 10CEII-9007.

Tabla de contenido

1. Introducción

Los sistemas de pronóstico de fallas y gestión de salud (PHM, por sus siglas en inglés, que corresponde a Prognostic and Health Management) permiten una estimación de la confiabilidad de un sistema basándose en su condición de vida actual. Para ello, es necesario recolectar una gran cantidad de datos en tiempo real desde equipos mecánicos [1]. Los datos de monitoreo de condiciones contienen información útil que revela la degradación de los equipos, por lo tanto, permiten predecir con precisión el futuro comportamiento del proceso de degradación y nos ayuda a calcular con mayor exactitud la vida útil restante de los equipos (RUL, por sus siglas en inglés, Remaining Useful Life)[2]. La estimación precisa del RUL es crucial en sistemas PHM, un sistema que predice fallas permite a los propietarios de los equipos tomar decisiones informadas a la hora de programar una mantención, con el fin de anteponerse a los hechos y prevenir grandes daños. Esto se traduce en una reducción significativa en costos tanto operacionales como de mantenimiento [3].

Las estrategias tradicionales, como el mantenimiento correctivo de la falla imprevista de equipos y el mantenimiento preventivo programado, se están volviendo menos capaces de satisfacer la creciente demanda industrial en el área de confiabilidad, debido de que requiere establecer modelos de ecuaciones explicitas y un gran conocimiento acerca de técnicas de procesamiento de señales y experiencia necesaria en pronostico de fallas [1]. En cambio, las tecnologías PHM, también conocidas como mantenimiento basado en condición (CBM, por sus siglas en ingles, que corresponde a Condition-Based Maintenance), están mostrando habilidades prometedoras para la aplicación en las industrias [4]. Generalmente, los métodos existentes para PHM pueden ser agrupados dentro de tres categorías: métodos basados en modelos (model-based approaches), métodos basados en datos (data-driven approaches) y una combinación de ambos (hybrid approaches).

Los métodos basados en modelos necesitan modelos de física de falla y modelos de propagación de daños para estimar el RUL de componentes o equipos. Sin embargo, la respuesta dinámica de sistemas mecánicos y el proceso de propagación de daño comúnmente son muy complejos, provocando que los modelos utilizados sean muy difíciles de construir. Por otra parte, los métodos basados en datos utilizan datos históricos de monitoreo de condiciones para modelar las características de la degradación y de como podría cambiar el proceso de inicio y propagación de daño, y así predecir la vida remanente de un equipo. El objetivo de los métodos basados en datos es modelar la relación entre la edad de un equipo, datos de monitoreo de condición, la degradación de los equipos y la vida remanente de estos, sin utilizar modelos de física de fallas [1, 4].

Todos los métodos mencionados requieren de complicadas técnicas de procesamiento de datos para extraer las características desde los datos provenientes de los distintos tipos de sensores o precisan de conocimiento de sistemas dinámicos para la construcción de modelos de ecuaciones explícitas. Esto requiere de un procesamiento y análisis manual de los datos mediante una persona experta en el área, lo que hace a estos métodos no aptos para un procesamiento y extracción au-

tomática de las características de los datos para big data. Los recientes desarrollos en modelos de aprendizaje profundo han brindado una oportunidad para crear métodos avanzados para la predicción del RUL para big data, esto es gracias a su capacidad de procesamiento de datos y extracción de características debido a sus múltiples estructura de capas. Por lo tanto, en el ámbito de métodos basados en datos se han propuesto diferentes modelos caracterizados como redes neuronales profundas para la predicción de daño. Entre ellos se puede encontrar la Red Neuronal Artificial Profunda (Deep ANN) propuesta por Tian [5], la Red Neuronal Profunda Totalmente Conectada propuesta por Ren et al.[6], la Long Short Term Memory (LSTM) propuesta por Yuan et al. [7], entre otras.

Es por ello, que se propone automatizar el análisis de datos de monitoreo en el proceso de degradación y propagación de daño para la predicción del RUL, utilizando técnicas de aprendizaje semi-supervisado, en particular, se formula un modelo generativo de entrenamiento adversario y semi-supervisado con regularización de modos para datos de series de tiempo, por lo que, las redes neurales a utilizar serán LSTM. El data set utilizado para el entrenamiento y validación de nuestro modelo es C-MAPSS turbofans, del cual se selecciona un porcentaje de etiquetas a utilizar en el proceso de entrenamiento. Una vez entrenado el modelo, se extrae el espacio latente para realizar la predicción de la vida remanente de las turbinas en cuestión, para ello el espacio latente ingresa a una red perceptrón multicapa totalmente conectada, la cual realiza la predicción del RUL. El mismo porcentaje de etiquetas utilizados para la primera parte del entrenamiento del modelo generativo, se utiliza para entrenar la MLP totalmente conectada y el resto de las etiquetas faltantes son generadas mediante el modelo generativo. Los resultados obtenidos son comparados con otros modelos de aprendizaje profundo como los son CNNs y LSTMs.

Lo nuevo que se propone en el presente trabajo de tesis tras el desarrollo del modelo generativo es:

- Regularización de modos aplicado al modelo generativo para series de tiempo.
- El discriminador tiene la capacidad de hacer predicciones y no solo clasificaciones.
- Aplicar aprendizaje semi-supervisado al modelo generativo sobre series de tiempo.

1.1. Objetivo General

Desarrollar un modelo generativo de aprendizaje profundo semi-supervisado para la predicción de la vida remanente de equipos.

1.2. Objetivos Específicos

- Definir el modelo generativo a utilizar.
- Explorar, analizar e identificar la arquitectura de una red neuronal recurrente óptima.
- Desarrollar una función de pérdida para entrenar el modelo.
- Implementar el modelo a través de un código computacional en la librería de TensorFlow.
- Comparar el modelo propuesto con el estado del arte para modelos que realizan la misma tarea.

1.3. Alcances

En el presente trabajo de tesis se pretende hacer un seguimiento del proceso de degradación de motores de aviación tipo turbofan, mediante la predicción de la vida remanente de los equipos, utilizando un data set de la NASA denominado C-MAPSS.

Para el desarrollo de esta propuesta de investigación, se implementarán técnicas de aprendizaje semi-supervisado, en particular un modelo generativo de entrenamiento adversario y semi-supervisado con regularización de modos, el cual solo extrae las características del data set, para posteriormente predecir el RUL mediante un perceptrón multicapa totalmente conectado, en ambos casos se considera solo un porcentaje de las etiquetas reales a la hora del proceso de entrenamiento.

El alumno trabajará con datos provenientes de una base de datos de la NASA llamado C-MAPSS (Commercial Modular Aero-Propulsion System Simulation), por lo que solo procesará y analizará los datos mencionados

El alumno solo procesará y analizará los datos otorgados por el profesor guía del presente trabajo de tesis. No se realizarán análisis de física de falla ni propagación de daño utilizando las técnicas tradicionales basadas en mantenimiento predictivo. El alcance principal es demostrar que la metodología propuesta permite la predicción automática del RUL, sin la necesidad de que alguien experto en el área supervise el modelo.

2. Metodología

En la figura 2.1 se muestra el flujo de trabajo a seguir. Se resume principalmente en tres etapas principales.

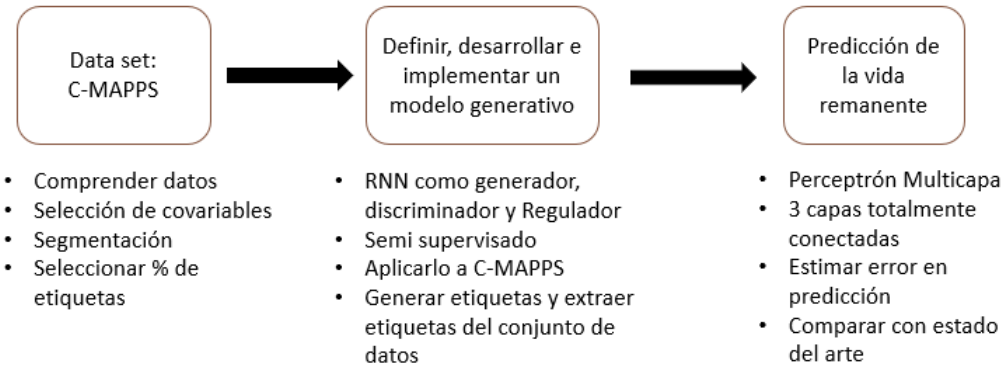


Figura 2.1: Metodología.

A continuación se detallan los distintos pasos a seguir para el desarrollo del presente trabajo de memoria, profundizando lo mostrado en la figura 2.1:

I. Análisis del conjunto de datos, lo que contempla:

   i. Compresión de los datos entregados y de sus covariables que afectan a la degradación de los equipos.

   ii. Selección de las covariables más influyentes en la predicción del RUL.

   iii. Segmentación del data set en ventanas de tiempo dados los ciclos de funcionamiento hasta la falla.

II. Definir, desarrollar e implementar un modelo generativo y aplicarlo a un conjunto de datos, lo que incluye:

   i. Desarrollar un modelo generativo de entrenamiento adversario semi supervisado con regularización de modos con redes neuronales recurrentes, a través del lenguaje de programación Python y la librería TensorFlow.

   ii. Implementar el modelo propuesto a través del conjunto de datos C-MAPSS.

   iii. Utilizar porcentajes de etiquetas del $2\,\%$, $10\,\%$ y $20\,\%$ en el proceso de entrenamiento del modelo.

   iv. Con el modelo entrenado, generar las etiquetas de los datos sin etiquetar, en concreto, se generarán el $98\,\%$, $90\,\%$ y $80\,\%$ de las etiquetas que serán complementarias a las mencionadas en el punto anterior.

v. Extraer el espacio latente del conjunto de datos.

III. Predicción de la vida remanente de los equipos mediante una MLP, esto comprende:

i. Desarrollar una MLP totalmente conectada para la predicción del RUL.

ii. Entrenar la red neuronal desarrollada con el espacio latente extraído del modelo generativo.

iii. Al entrenar la MLP, utilizar el mismo porcentaje de etiquetas provenientes del conjunto de datos y complementarlas con las generadas por el modelo generativo.

iv. Realizar la predicción del RUL y cuantificar la precisión de la predicción mediante la métrica root mean square error.

v. Comparar los resultados obtenidos con otros modelos de aprendizaje profundo como CNNs y LSTMs.

3. Antecedentes

En esta sección se presentan técnicas y modelos utilizados para el desarrollo de la presente tesis. Como herramienta principal se utilizan las llamadas redes adversarias generativas (GANs), la cual forma parte de los modelos generativos que a su vez pertenecen a la rama del aprendizaje de maquinas no supervisado.

Los modelos generativos consisten en que un modelo cualquiera toma un conjunto de datos de entrenamiento provenientes de una distribución desconocida p_{data} y aprende a representar una estimación de dicha distribución de alguna manera. El resultado de lo mencionado anteriormente es una distribución de probabilidad p_{model}, la cual puede ser estimada de manera explicita como lo muestra la figura 3.1 o de manera implícita y utilizada solamente para generar ejemplos a partir de p_{model}, tal como se muestra en la figura 3.2 [8]. Las GANs se centran principalmente en la generación de muestras, aunque es posible diseñar una GANs que haga ambas cosas.

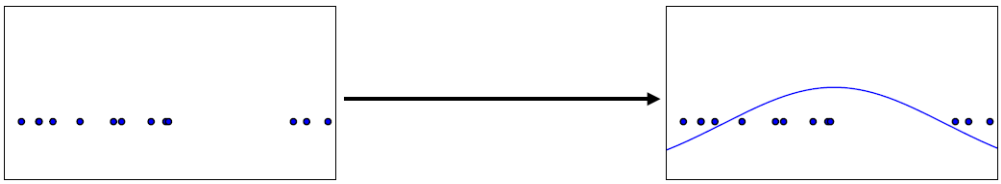


Figura 3.1: Esta figura ilustra el proceso para una colección de muestras de datos unidimensionales y un modelo gaussiano. Fuente: [8].

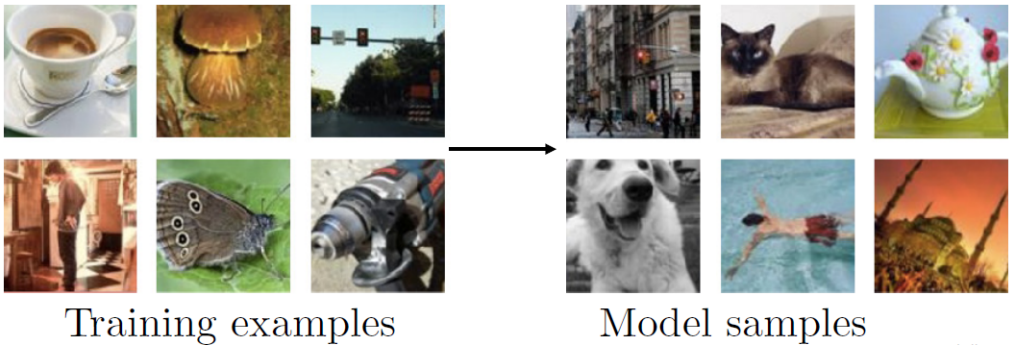


Figura 3.2: Algunos modelos generativos pueden generar muestras a partir de la distribución del modelo. En esta ilustración del proceso, mostramos muestras de ImageNet (Deng *et al.*, 2009, 2010; Conjunto de datos de Russakovsky *et al.*, 2014). Un modelo generativo sería ser capaz de entrenar en ejemplos como se muestra a la izquierda y luego crear más ejemplos de la misma distribución que se muestra a la derecha. Fuente: [8].

3.1. Redes Adversarias Generativas

La idea básica de las redes adversarias generativas o GANs por sus siglas en inglés (Generative Adversial Networks), es establecer un juego no cooperativo entre dos jugadores, un jugador es llamado generador y el otro discriminador [8]. El generador crea muestras provenientes de una estimación de la distribución de los datos de entrenamiento, mientras que el discriminador obtiene muestras tanto del generador como del conjunto de entrenamiento y tiene que poder distinguir de donde provienen cada una de estas, o dicho de otro modo determinar si son verdaderas o falsas [9]. Por lo que el objetivo del generador es engañar al discriminador y que este último clasifique las imágenes generadas como verdaderas.

A medida que avanza el juego, el generador aprende a producir muestras cada vez más reales y el discriminador por su parte aprende a reconocer cada vez mejor los datos generados de los datos reales, todo esto con el objetivo de que al termino de la competencia los datos generados sean indistinguibles de los datos reales.

Para conocer la distribución p_{model} del generador sobre los datos de entrenamiento, los cuales poseen una distribución desconocida p_{data}, se define una distribución de probabilidad a priori p_z sobre las variables de entrada, las cuales son generadas de manera aleatoria. Luego se representa una asignación del espacio de datos como $G(z;\theta_g)$, en donde G es una función diferenciable representada por una red neuronal con parámetros θ_g. Se define una segunda red neuronal $D(x;\theta_d)$, en donde D también es una función diferenciable con parámetros θ_d. Se tiene que $D(x)$ representa la probabilidad de que x provenga de los datos de entrenamiento en lugar de p_{model}, por lo que D se entrena para maximizar la probabilidad de asignar la etiqueta correcta en ambos casos [10]. En el juego, se tiene que G y D representan al generador y discriminador respectivamente.

Para efecto del proceso de entrenamiento, ambos jugadores poseen funciones de costo que son definidas en términos de los parámetros de ambos jugadores. El discriminador desea minimizar la función $J^{(D)}(\theta^{(D)},\theta^{(G)})$ ilustrada en la ecuación 3.1, en donde sólo puede controlar o modificar los parámetros $\theta^{(D)}$. El generador por su parte desea minimizar $J^{(G)}(\theta^{(D)},\theta^{(G)})$ y sólo puede controlar los parámetros $\theta^{(G)}$.

$$J^{(D)}(\theta^{(D)},\theta^{(G)}) = -\mathbb{E}_{x\sim p_{data}}\log D(x) - \mathbb{E}_z\log(1-D(G(z))) \tag{3.1}$$

La ecuación 3.1 no es otra cosa que el costo estándar de *cross-entropy*, el cual es minimizado cuando se entrena un clasificador binario con una salida sigmoidea [8]. La única diferencia es que el clasificador utiliza dos conjuntos de datos, uno proveniente del conjunto de entrenamiento donde los datos poseen una etiqueta con el valor de 1 y otro de los generados por el generador a los cuales se les asigna la etiqueta de 0, dichas etiquetas representan si un dato es real o falso respectivamente [11]. El proceso es ilustrado en la figura 3.3.

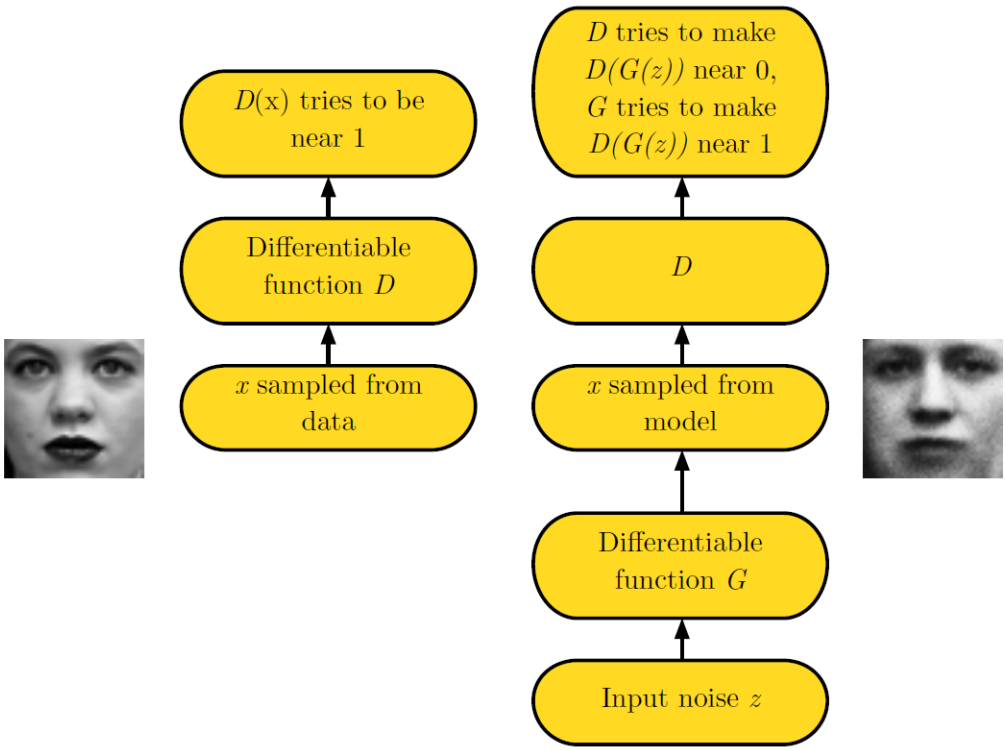


Figura 3.3: El juego se juega en dos escenarios. En la izquierda se presenta el escenario del primer jugador, el discriminador. A la derecha se muestra el escenario del segundo jugador, el generador. En el primer escenario el objetivo del discriminador es que $D(x)$ sea cercano a 1, mientras que en el segundo es hacer que $D(G(z))$ se acerque a 0, mientras que el generador se esfuerza porque la misma cantidad se acerque a 1. Fuente: [8].

Existen varios tipos de juegos para entrenar una GANs, en todos ellos la función de costo para el discriminador es la misma, la diferencia radica en la función de costo del generador. A continuación se presentan dos de ellos.

3.1.1. Juego minimax

La versión más simple del juego es la llamada *zero-sum game* o también conocida como *minimax game*, en donde la suma de todos los costos de los jugadores es siempre cero [8]. Según esta versión del juego la función de costo del generador esta dada por:

$$J^{(G)} = -J^{(D)} \tag{3.2}$$

Se define una función de valor para resumir el juego, la cual especifica la recompensa del discriminador. Dicha función se ilustra en la ecuación 3.3.

$$V(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)}) \tag{3.3}$$

El juego es llamado minimax porque la solución de éste implica la minimización en un loop externo y la maximización en un loop interno [8], tal como se muestra en la ecuación 3.4:

$$(G^*, D^*) = \arg\min_{\mathrm{G}} \max_{\mathrm{D}} V(\theta^{(D)}, \theta^{(G)}) \tag{3.4}$$

El costo usado por el generador en el *zero-sum game* es útil para análisis teóricos, pero en la practica no funciona de buena manera al no proporcionar el gradiente adecuado para que G aprenda de manera correcta. Lo que quiere decir que al inicio del entrenamiento, cuando G es pobre, D puede clasificar las muestras provenientes del generador como falsas con una alta confiabilidad porque son claramente diferentes a las muestras provenientes del conjunto de entrenamiento.

Otro tipo de juego es presentado a continuación.

3.1.2. Juego heurístico no saturado

En el juego minimax, el discriminador minimiza la *cross-entropy* pero el generador maximiza la misma *cross-entropy*, lo cual hace que el gradiente del generador desaparezca. Para evitar este problema, una aproximación es continuar utilizando *cross-entropy* para el generador pero esta vez realizar una minimización [8].

Para definir la función de costo del generador, en lugar de anteponerle el signo negativo a la función de costo del discriminador, se cambia el objetivo utilizado para definir el costo de *cross-entropy*. El nuevo costo para el generador es el ilustrado en la ecuación 3.5.

$$J^{(G)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_z \log D(G(z)) \tag{3.5}$$

En este juego el generador maximiza la log-probability de que el discriminador cometa un error, en cambio en el juego minimax el generador minimiza la log-probability de que el discriminador sea correcto. Una consideración a tener en cuenta con esta modalidad de entrenamiento es que hay que asegurarse que el jugador que se encuentre perdiendo tenga un gradiente robusto [8].

3.1.3. El proceso de entrenamiento

El entrenamiento de una GANs consiste en el descenso de gradiente estocástico simultáneo, también conocido como SGD por su siglas en ingles (Stochastic gradient descent). En cada paso se muestrean dos minibatch, uno de valores de x proveniente del conjunto de entrenamiento y el otro de los valores de z extraídos de la *prior distribution* sobre las variables latentes. Luego se realizan dos pasos de gradiente simultáneamente, uno actualiza $\theta^{(D)}$ para reducir $J^{(D)}$ y otro

actualiza $\theta^{(G)}$ para reducir $J^{(G)}$ [8]. En ambos casos es posible utilizar cualquier algoritmo de optimización basado en gradiente, Adam [12] es una buena opción.

La solución para un problema de optimización corresponde a un mínimo local, un punto en el espacio de parámetros donde todos los puntos vecinos tienen un costo mayor o igual. La solución al juego realizado por la GANs es cuando se alcanza el equilibrio de Nash, el cual en este contexto corresponde a una tupla $(\theta^{(D)}, \theta^{(G)})$ que es un mínimo local de $J^{(D)}$ con respecto a $\theta^{(D)}$ y un mínimo local de $J^{(G)}$ con respecto a $\theta^{(G)}$.

Para poder alcanzar el equilibrio de Nash, Goodfellow [10] propone un algoritmo que optimiza la ecuación 3.4, bajo el supuesto que los modelos tienen suficiente capacidad y tiempo de entrenamiento. El algoritmo se ilustra en la tabla 3.1.

Tabla 3.1: Algoritmo de entrenamiento con descenso de gradiente estocástico de redes adversas generativas. El número de pasos a aplicar al discriminador, k, es un hiperparámetro [10].

For número de pasos **do**

For k pasos **do**

- Extraer una muestra de m ejemplos $z^{(1)}, ..., z^{(m)}$ como entrada de ruido para $p_{model}(z)$.
- Extraer una muestra de m ejemplos $x^{(1)}, ..., x^{(m)}$ como datos generadores para la distribución $p_{data}(x)$.
- Actualizar el discriminador ascendiendo su gradiente estocástico:

$$\nabla\theta_d \frac{1}{m} \sum_{i=1}^{m} [\log(D(x^{(i)})) + \log(1 - D(z^{(i)}))]$$

End For

- Extraer una muestra de m ejemplos $z^{(1)}, ..., z^{(m)}$ como entrada de ruido para $p_{model}(z)$
- Actualizar el generador descendiendo su gradiente estocástico:

$$\nabla\theta_g \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z^{(i)})))]$$

End For

3.2. Redes convolucionales generativas adversarias profundas

La GAN propuesta por GoodFellow [10] utiliza perceptrones multicapas como redes neurales tanto para el generador como para el discriminador. Este tipo de arquitectura es aplicada a conjuntos de datos relativamente simples, como lo son MNIST (Digitos escritos a mano), CIFAR-10 (Imágenes naturales) y TFD (Toronto Face Dataset) [11]. Las imágenes que se generaron con esta

arquitectura presentaban problemas de ruido y eran incomprensibles al momento de visualizarlas.

Radford [13] propone una familia de arquitecturas para generar imágenes de alta resolución llamada DCGAN por su siglas en inglés (*Deep Convolutional Generative Adversial Networks*). En donde, tanto el generador como el discriminador son redes convolucionales profundas que cumplen con una determinada arquitectura, la cual resulta en un entrenamiento estable, permitiendo entrenar modelos generativos más profundos y con una alta resolución. Algunas de las ideas claves [11] de la arquitectura DCGAN fueron:

- Usar *Batch Normalization* [14] en todas las capas del generador menos en la última y en todas las capas del discriminador menos en la de entrada. Lo anterior estabiliza el aprendizaje normalizando el input de cada unidad para tener media cero y varianza de la unidad.
- Reemplazar las capas de reducción o *pooling* con pasos convolucionales para el discriminador y con pasos fraccionados para el generador, permitiendo a la red aprender su propio submuestreo espacial.
- Remover las capas completamente conectadas para arquitecturas profundas.
- Usar la función de activación ReLU para todas las capas del generador menos la última que usa una función Tanh. Usar la función de activación LeakyReLU para el discriminador en todas sus capas.

Un ejemplo de la arquitectura del generador del modelo propuesto por Radford es presentado en la imagen 3.4. Por otra parte, el discriminador posee una estructura similar, pero en sentido contrario, fluyendo los datos de izquierda a derecha en relación a la imagen 3.4.

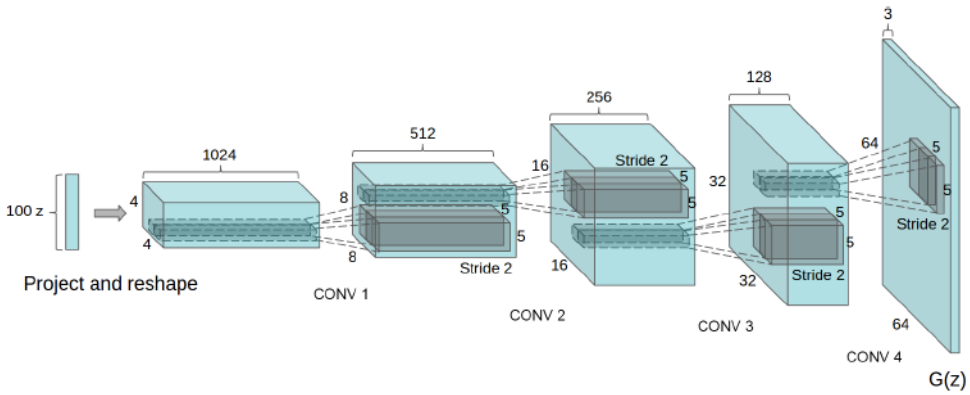


Figura 3.4: DCGAN generador. Un vector z de dimensionalidad 100 es proyectado en una representación convolucional espacial con muchos mapas de características. Una serie de 4 pasos convolucionales fraccionados convierten esto en una representación de alto nivel dentro de una imagen de 64 x 64 pixeles. Fuente: [13].

3.3. Redes Adversarias Generativas con regularización de Modos

Las redes adversarias generativas generalmente son consideradas muy difíciles de entrenar debido a su inestabilidad durante el entrenamiento y la sensibilidad a los distintos hiperparámetros. Por otra parte un patrón común de falla observado mientras se entrena una GANs son las pérdidas de modos, es decir, que el generador produce muestras correspondiente a pequeñas regiones de alta probabilidad bajo la distribución de datos p_{data}.

El problema de la pérdida de modos es causado por dos factores:

- Las áreas cercanas de los modos faltantes rara vez son visitadas por el generador.
- Tanto los modos faltantes como los no faltantes tienden a corresponder un alto valor del Discriminador.

A raíz de lo anterior [15] introduce las redes adversarias generativas con regularización de modos o MRGANs por sus siglas en ingles (*Mode Regularized Generative Adversial Networks*). La cual consiste en agregar una señal de entrenamiento supervisado al generador, asumiendo que el generador $G(z) : Z \rightarrow X$ genera muestras desde una distribución a priori en el espacio Z seguida de una transformación G en el espacio muestral X. En conjunto con G, también se entrena un codificador denominado $E(x) : X \rightarrow Z$ el cual es entrenado para minimizar el error de reconstrucción entre la muestra proveniente del conjunto de entrenamiento y la generada por el generador a partir del espacio latente del encoder, esto mediante la función de costo ilustrada en la ecuación 3.6.

$$J^{(E)} = \mathbb{E}_{x \sim p_{data}}[\lambda_1 d(x, G \circ E(x)) + \lambda_2 \log D(G \circ E(x))] \tag{3.6}$$

En donde, el término $\mathbb{E}_{x \sim p_{data}} d(x, G \circ E(x))$ corresponde a un regularizador en el proceso de entrenamiento, en el que se asume d como alguna métrica de similaridad en el espacio de datos. Mientras que la expresión $\log D(G \circ E(x))$ es un regularizador de modos utilizado para penalizar aún más los modos faltantes. Por otra parte las constantes λ_1 y λ_2 son ponderadores de ambas expresiones y corresponden a hiperparámetros.

La pérdida del generador para el caso de la MRGANs se ilustra en la ecuación 3.7, la cual posee el objetivo de optimización de la GANs tradicional, mostrado en la ecuación 3.5 más los regularizadores de modos propuestos.

$$J^{(G)} = -\mathbb{E}_z \log D(G(z)) + \mathbb{E}_{x \sim p_{data}}[\lambda_1 d(x, G \circ E(x)) + \lambda_2 \log D(G \circ E(x))] \tag{3.7}$$

Un esquema del flujo de información de la MRGANs es presentado en la figura 3.5. En la práctica, el Encoder tiene la misma estructura que el discriminador, sólo cambia la dimensionalidad del vector de salida, que para el caso del Encoder tiene la misma forma que el espacio Z.

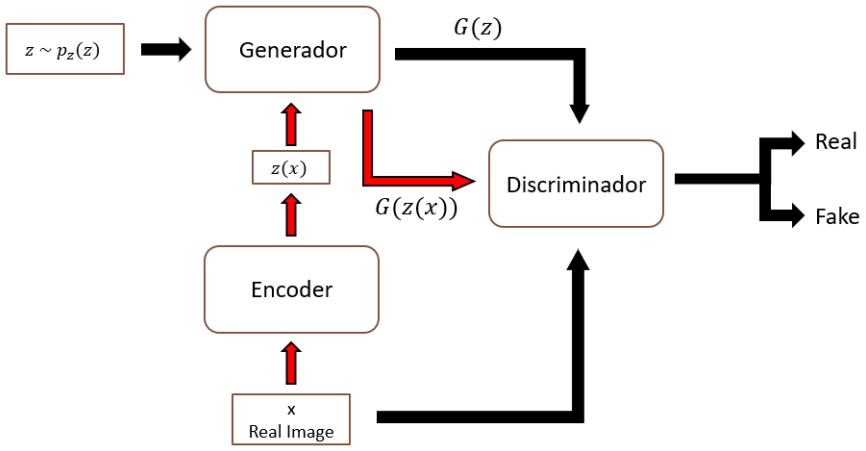


Figura 3.5: Arquitectura de MRGANs propuesta por [15]. Fuente: Elaboración propia.

3.4. Redes Adversarias Generativas Recurrentes

Los tipos de GANs nombrados hasta el momento, procesan imágenes correspondientes a un instante en el tiempo, por lo cual las redes utilizadas como generador y discriminador son principalmente perceptrón multicapa y redes neuronales convolucionales. Existen otra clase de datos en donde este tipo de redes recién mencionadas no aplican para su procesamiento, dichos datos son secuenciales continuos o series de tiempo. Se entiende como una secuencia de datos, valores que son medidos en determinados instantes y son ordenados cronológicamente, por lo que se hace necesario que las redes que los procesen capten esta estructura. Las redes neuronales recurrentes (RNNs) constituyen una herramienta apropiada para modelarlos, se trata de un tipo de redes que tienen cierta memoria y que, por lo tanto, un sentido temporal.

Con el propósito de generar datos secuenciales, se introducen las Redes Adversarias Generativas Recurrentes, en donde Mogren [16], con el objetivo de producir música polifónica utiliza una GANs con dos diferentes redes neuronales recurrentes profundas, una representa al generador y otra al discriminador, la red recurrente utilizada es una *Long Short-Term memory* (LSTM) [17]. Estas tienen una estructura interna con puertas que ayudan con el problema de pérdida de gradiente y además aprenden dependencias más largas [18]. Al igual que en el entrenamiento tradicional de la GANs propuesta por Goodfellow el generador es entrenado para generar datos que sean indistinguibles de los datos reales, mientras que el discriminador es entrenado para identificar los datos generados de los reales.

El entrenamiento del modelo corresponde al juego mini-max mencionado en la sección 3.1.1,

la función de pérdida para el discriminador es la definida en la ecuación 3.1. Este es entrenado para minimizar el promedio de la entropía cruzada negativa entre la predicción por paso de tiempo y las etiquetas de la secuencia, para el caso de las secuencias reales las etiquetas corresponden a unos, y para el caso de las secuencias generadas las etiquetas son ceros [19]. La función de pérdida para el generador es la mencionada en la ecuación 3.2.

La arquitectura propuesta por Mogren es la ilustrada en la figura 3.6, en donde, como input para cada celda del generador se tiene un vector aleatorio, concatenado con las salidas de las celdas previas. Alimentar las celdas con la información de salida de celdas previas es una práctica común cuando se entrenan RNNs [20].

El discriminador consiste en una red recurrente bidireccional, permitiéndole tener en cuenta información en ambas direcciones para sus decisiones, tal como se aprecia en la figura 3.6.

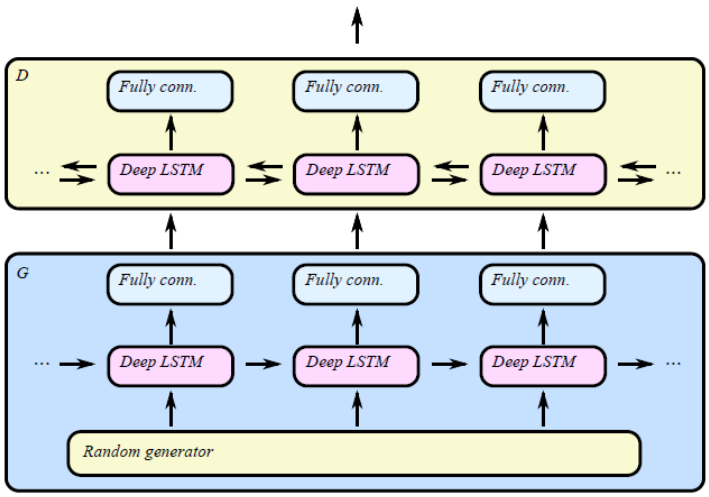


Figura 3.6: Arquitectura de C-RNN-GAN. Fuente: [16].

4. Propuesta de GAN Recurrente Semi Supervisada con regularización de modos para la estimación del RUL

En la presente sección se presentará el modelo propuesto a un nivel general, explicando y discutiendo su arquitectura y flujo de información.

El modelo propuesto se basa en un modelo generativo de entrenamiento adversario, el cual se desarrolló en el lenguaje de programación Python 3.5 y la biblioteca de TensorFlow. A diferencia de una GANs tradicional, la cual procesa imágenes correspondientes a un instante de tiempo, nuestro modelo procesa datos temporales, por lo que se utilizan redes LSTM tanto como generador y discriminador. Además, debido a la inestabilidad que presentan las GANs se considera un regularizador para estabilizar las pérdidas en el proceso de entrenamiento y que los modos con baja probabilidad de ser visitados por el generador sean visitados, para ello, se implementa un regularizador de modos el cual también es una red LSTM.

Con el modelo generativo se pretende procesar un conjunto de datos temporales, los cuales consisten en mediciones provenientes del monitoreo continuo de los equipos hasta la falla. Esto permite cuantificar las mediciones sobre la condición del equipo dada la vida remanente, se considera dicha característica como etiquetas de los datos temporales. Se busca que el modelo sea capaz de encontrar el espacio latente que explica la degradación de los equipos y generar etiquetas para los datos dado un cierto porcentaje de éstas. Por lo que, tentativamente no sería necesario contar con todas las etiquetas de los datos para el proceso de predicción del RUL, dando la posibilidad de reducir costos en cuanto a la toma de datos.

La predicción del RUL se llevará a cabo mediante una MLP totalmente conectada, a la cual le ingresa el espacio latente de los datos en conjunto de las etiquetas tanto reales como generadas.

Un esquema de la arquitectura de nuestro modelo generativo que llamaremos MRLSTMGANs semi-supervisado es presentado en la figura 4.1. Al generador entran dos distribuciones de datos, una proveniente de una distribución a priori $P_z(z)$ cuyos datos se encuentran en un rango entre 0 y 1, y otra procedente de las imágenes del conjunto de entrenamiento sin etiquetas, las cuales pasan a través de un Encoder que captura el espacio latente $z(x)$ de dichas imágenes. Lo anterior se hace de tal modo que ambas distribuciones tengan las mismas dimensiones al momento de ingresar al generador y así no tener problemas de incompatibilidad en el ingreso de datos.

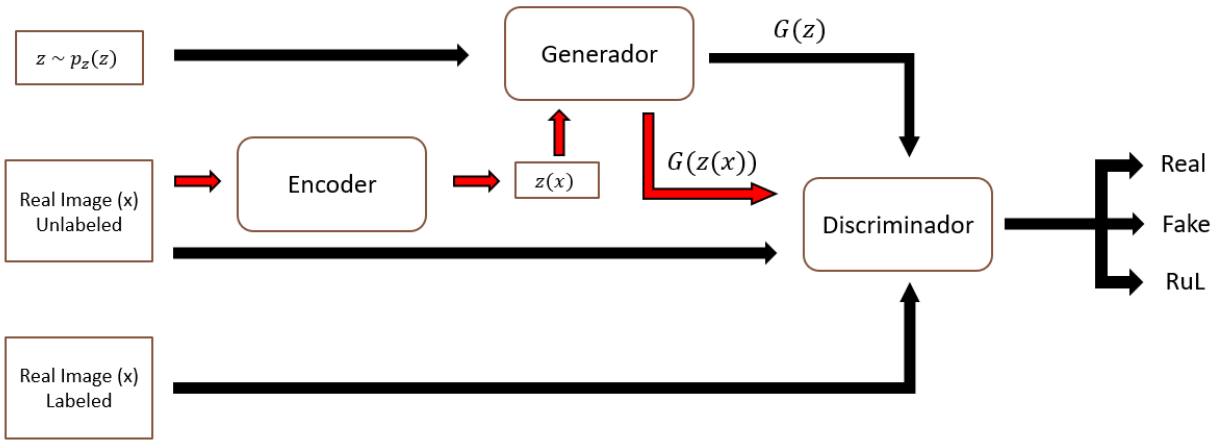


Figura 4.1: Arquitectura de MRLSTMGANs. Fuente: Elaboración propia.

El generador por su parte tiene dos salidas de datos, una de ellas corresponden a las imágenes generadas a partir de la distribución a priori $P_z(z)$, esta salida es denominada como $G(z)$ y la otra salida proveniente de las imágenes generadas a partir del espacio latente del Enconder, designada por $G(z(x))$. Ambas tienen las mismas dimensiones que las imágenes provenientes del conjunto de entrenamiento y son nuevas entradas para el discriminador.

Por otra parte, el discriminador tiene 4 entradas durante el proceso de entrenamiento, las dos correspondientes al generador y las otras dos correspondientes al conjunto de datos, de las cuales una entrada es con etiquetas y la otra sin etiquetas. A diferencia de una GANs, el discriminador propuesto cumple dos funciones, una es clasificar las imágenes como verdaderas o falsas y la otra es la predicción del RUL. La primera función nombrada tiene como objetivo que el generador aprenda a generar imágenes cada vez más reales o similares a las del conjunto de entrenamiento y de la misma manera que el discriminador como tal aprenda a diferenciar de donde provienen las imágenes que le ingresan, y así ir captando de mejor manera el espacio latente de los datos. De este modo, identificamos dos procesos de entrenamiento dentro del discriminador, uno no supervisado (clasificar imágenes) y uno supervisado (estimar el RUL), por lo que se considera que el modelo es semi-supervisado.

4.1. Funciones de costo

Para entrenar el modelo propuesto se definen funciones de costos tanto para el Generador, Discriminador y Enconder. Para el Generador se define la formula presentada en la ecuación 4.1. El error de reconstrucción de la imagen proveniente del Encoder es representado por el término $d(x, G \circ E(x))$, en donde, d es una métrica de similitud, que para el caso del presente trabajo de tesis es la presentada en la ecuación 4.2.

$$J^{(G)} = -\mathbb{E}_z \log D(G(z)) + \mathbb{E}_{x \sim p_{data}}[\lambda_1 d(x, G \circ E(x)) + \lambda_2 \log D(G \circ E(x))] \tag{4.1}$$

$$\mathrm{d} = \sqrt{\sum_{i=1}^{n} (x_i - G(E(x_i)))^2} \tag{4.2}$$

La función de costo para el discriminador es la presentada en la ecuación 4.3, en la cual los dos primeros términos forman parte de un aprendizaje no supervisado. Además se añade a modo de un estabilizador de pérdidas un término proveniente del aprendizaje supervisado correspondiente al último término de la ecuación.

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\mathbb{E}_{x \sim p_{data}} \log D(x) - \mathbb{E}_z \log(1 - D(G(z))) + \mathbb{E}_{(x,y) \sim p_{data}(x,y)} \|D(x) - y\|_2^2 \tag{4.3}$$

Para el Encoder la función de costo se ilustra en la ecuación 4.4, la cual esta contenida como se observa dentro de la función de costos del Generador, actuando de esta manera como un regularizador de modos, con tal de que los modos dentro del conjunto de datos con baja probabilidad de ocurrencia sean reproducidos por el generador, ampliando de esta manera el rango o la capacidad del generador de replicar las imágenes del conjunto de datos en cuestión.

$$J^{(E)} = \mathbb{E}_{x \sim p_{data}}[\lambda_1 d(x, G \circ E(x)) + \lambda_2 \log D(G \circ E(x))] \tag{4.4}$$

4.2. Proceso de entrenamiento

En la figura 4.2 se muestra el proceso de entrenamiento del modelo a nivel global. Una vez realizado el entrenamiento, se extrae el espacio latente de los datos y se generan las etiquetas correspondientes para una posterior predicción del RUL.

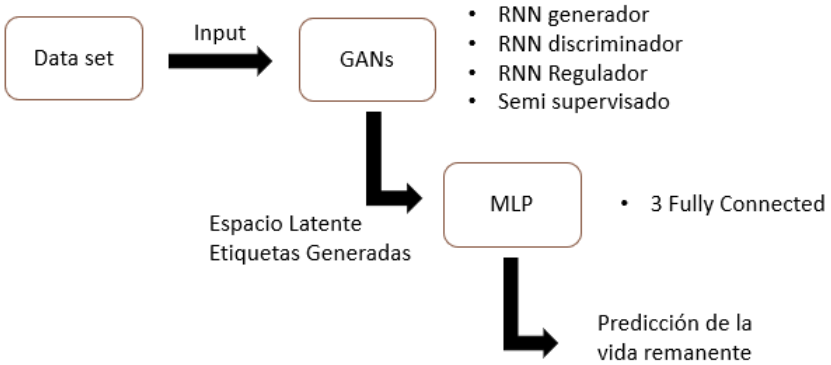


Figura 4.2: Proceso de obtención del RUL. Fuente: Elaboración propia.

Al momento de entrenar el modelo se consideran distintos escenarios en cuanto al porcentaje de etiquetas a utilizar, esto para cuantificar la variación en los resultados dependiendo del nivel de etiquetas a considerar. Al extraer el espacio latente de los datos, se conserva la relación dato-etiqueta. Para los datos sin etiquetas, el modelo las genera, ya que tiene la capacidad de predecir el RUL mediante el discriminador, por lo que se hace una relación entre dato original y etiqueta generada, esto con el fin de que al momento de concluir el entrenamiento cada dato tenga su etiqueta correspondiente.

Todo esto es con el objetivo de que al momento de entrenar la MLP se utilicen todos los datos con sus respectivas etiquetas, independientemente si son originales o generadas. Luego de tener los resultados se hace la comparación con el estado del arte, principalmente con dos modelos uno de LSTM [24] y otro de CNN [4].

Para optimizar la predicción del RUL en los modelos propuestos (MRLSTM GANs y MLP), se utiliza la función error cuadrático medio o RMSE (por sus siglas en ingles, Root Mean Squared Error) como función de pérdida. La función se presenta en la ecuación 4.5.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_p - y_t)_i^2} \tag{4.5}$$

Donde,

- N es la cantidad de muestras en el conjunto de datos de prueba.
- y_p es la predicción del RUL.
- y_t es el RUL verdadero.

Para evaluar la capacidad de predicción del modelo sobre el conjunto de datos se pueden emplear distintas métricas según el objetivo que se busque. En el escenario de la degradación de turbinas se busca tener una predicción conservadora y, por lo tanto temprana de la falla. Dado este contexto, es que se utiliza una función de cuantificación de resultados o *scoring function* [22], ilustrada en la ecuación 4.6. La cual da noción del desempeño del modelo penalizando el aumento de error de manera exponencial realizando la distinción para cuando el error es negativo o positivo, para ello las ponderaciones en el denominador del exponente de la función son distintas según sea el caso. Las penalizaciones son más altas para las predicciones tardías, es decir, para las predicciones más allá del tiempo de falla real. Esto, ya que el objetivo es prevenir la falla dado el contexto operacional de las turbinas de avión.

$$s = \begin{cases} \sum_{i=1}^{N} e^{-\left(\frac{d}{10}\right)} - 1 & \text{for} \quad d < 0 \\ \sum_{i=1}^{N} e^{\left(\frac{d}{13}\right)} - 1 & \text{for} \quad d \geq 0 \end{cases} \tag{4.6}$$

Donde,

- s es la medida calculada.
- N es la cantidad de muestras en el conjunto de datos de prueba.
- $d = y_p - y_t$ es la diferencia entre la predicción del RUL y el RUL verdadero.

En la figura 4.3 se muestra la función de cuantificación como una función de error, se observa que es asimétrica con respecto a cero y cuando el error es cercano a cero la penalización tiende al mínimo.

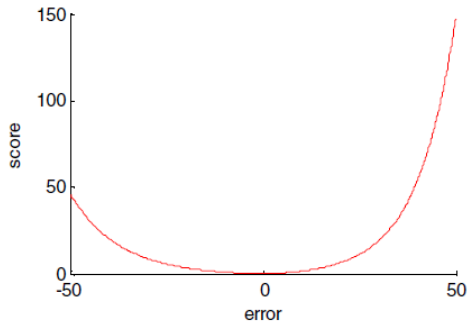


Figura 4.3: Score como una función de error . Fuente:[22]

5. Caso de Estudio: C-MAPSS

En la presente sección se presentará el desarrollo de los distintos algoritmos a utilizar, para luego dar paso a los resultados y su posterior análisis. De acuerdo a la metodología planteada en la sección 2, se comenzó con la comprensión del conjunto de datos.

5.1. Descripción del conjunto de datos

El modelo propuesto en el presente trabajo de tesis será utilizado sobre un conjunto de datos de la NASA llamado C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) [22], el cual es generado a través de un simulador de motores turbofan.

El pronostico de fallas es muy importante en el mundo de los motores de aviones, debido al alto costo asociado a un mal funcionamiento. Por costos no solo se entienden los asociados a los que producen pérdidas monetarias, sino que también a la potencial pérdidas de vidas humanas.

Una turbina a gas comúnmente esta conformada por cinco módulos: un ventilador, un compresor de baja presión, un compresor de alta presión, una turbina de baja presión y una de alta presión, tal como se muestra en la figura 5.1. A lo largo de la turbina, 21 sensores son usados para monitorear su rendimiento, a través de mediciones de velocidad, temperatura y presión en diferentes puntos.

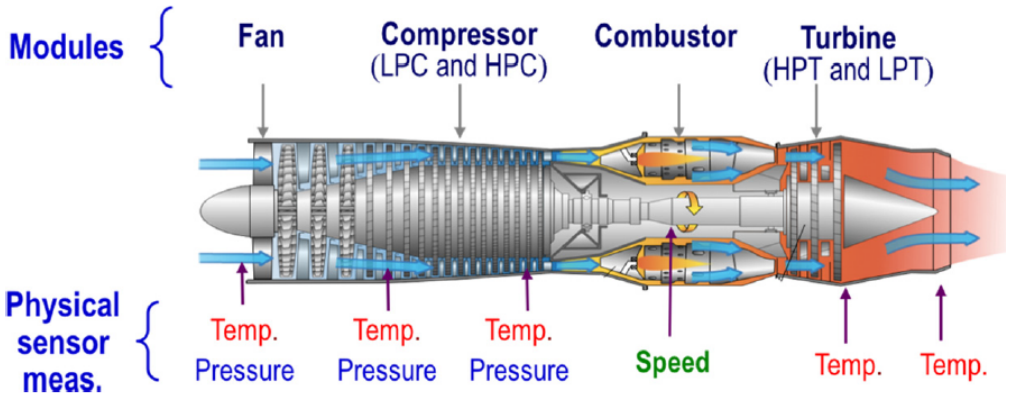


Figura 5.1: Diagrama de los módulos de una turbina de avión. Fuente: [22].

El conjunto de datos consiste en 4 sub conjuntos, cada uno con diferentes condiciones de operación y de fallas. En la tabla 5.1 se muestran el número de trayectorias tanto de entrenamiento como de prueba, el número de condiciones de falla y operación para cada conjunto. Esta distinción entre entrenamiento y prueba es con el objetivo de entrenar modelos de aprendizaje profundo. La complejidad de cada sub conjunto incrementa con el aumento de las condiciones de operación y de

fallas [23], por ende, los conjuntos FD002 y FD004 son considerados por ser un conjunto de datos complejo de analizar.

Cada trayectoria es considerada única proveniente de diferentes turbinas a gas, las que son operadas bajo condiciones normales y además presentan degradación a partir de un cierto número de ciclos de uso. En las turbinas del conjunto de entrenamiento la degradación ocurre hasta la falla, mientras que en el conjunto de prueba ocurre hasta un instante antes dejando una vida remanente en los equipos y cuyo valor es el RUL objetivo para el modelo.

Tabla 5.1: Conjunto de datos C-MAPSS [3].

Dataset	FD001	FD002	FD003	FD004
Trayectorias entrenamiento	100	260	100	249
Trayectorias de prueba	100	259	100	248
Condiciones de Operación	1	6	1	6
Condiciones de falla	1	1	2	2

Los datos son organizados dentro de una matriz de tamaño $Nx26$, donde N denota el largo total de la concatenación de todas las trayectorias a lo largo del eje temporal. La primera columna de la matriz representa el ID de cada turbina, la segunda columna representa los ciclos de funcionamiento, de la tercera a la quinta columna representan las configuraciones operacionales que tienen un efecto sustancial sobre el rendimiento [23], y las últimas 21 columnas son los valores de las mediciones realizadas por los sensores.

El objetivo de este conjunto de datos es estimar el RUL de cada turbina presente en las trayectorias de prueba. Tradicionalmente, la función objetivo del RUL puede ser representada como una degradación lineal a lo largo de los ciclos de operación de un equipo [21], pero en las etapas iniciales puede ser considerada como constante, es decir, despreciable. Lo cual es muy útil, ya que en los primeros ciclos de operación de la turbina como se mencionó anteriormente se considera que no hay desgaste. Es por ello, que para el conjunto de datos los primeros ciclos de operación se considera un constantes y con un valor de $R_{early} = 125$.

5.1.1. Pre-procesamiento de datos

No todos los sensores descritos en C-MAPSS entregan información valiosa sobre la degradación de los equipos, a raíz de esto se seleccionan los más relevantes. Básicamente se eliminan aquellos sensores que son constantes en el tiempo, los que corresponden (en una enumeración del 1 al 21) a los sensores 1, 5, 6, 10, 16, 18 y 19, quedando 14 sensores sobre los cuales se entrenará el modelo generativo propuesto para la extracción de características.

De los valores obtenidos de los 14 sensores a utilizar no todos tienen la misma escala, por esa razón, se aplica una normalización sobre cada sensor de manera individual. La formula a utilizar es la ilustrada en la ecuación 5.1.

$$\widetilde{x}_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{5.1}$$

De manera complementaria a la normalización se realiza un aumento de datos, lo que en inglés se conoce como *data augmentation*, la figura 5.2 representa la función objetivo de una trayectoria de entrenamiento completa.

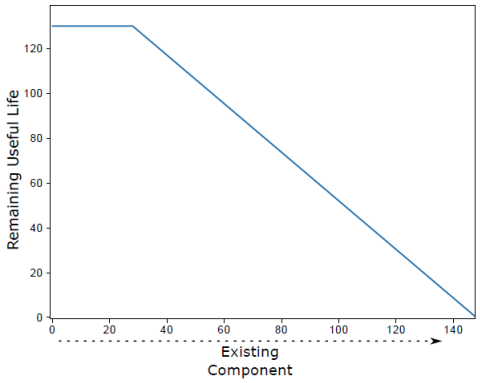


Figura 5.2: Antes del aumento de datos. Fuente: [3].

Al momento de aumentar los datos, se utiliza la trayectoria de entrenamiento completa para generar una trayectoria de entrenamiento parcial, tal como se muestra en la figura 5.3, en donde, para este caso en particular se generaron 3 trayectorias de entrenamiento parciales a partir de la original. Cada trayectoria parcial es obtenida a partir de un truncamiento de la trayectoria completa en puntos al azar a lo largo de la degradación lineal. Vale la pena recalcar que el conjunto de prueba tiene trayectorias similares a las trayectorias parciales generadas, ya que estas son truncadas hasta un punto temprano al punto de falla.

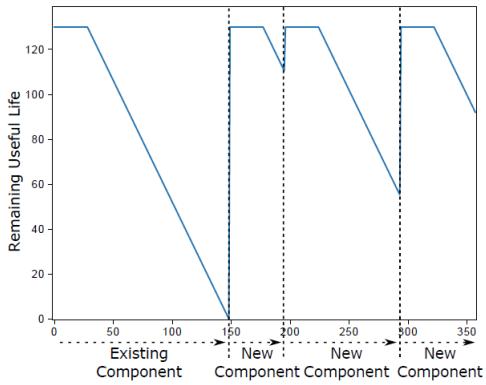


Figura 5.3: Después del aumento de datos. Fuente: [3].

Para efectos de la red neuronal recurrente y del proceso de entrenamiento como tal, es que se necesita separar las trayectorias por ventanas de tiempo o de ciclos en el caso de C-MAPSS, aumentando así la cantidad de datos a procesar. El criterio para definir las ventanas de tiempo en cada uno de los subconjuntos de datos es identificando la trayectoria que tiene el menor número de ciclos para luego, definirla como la ventana de tiempo o *timestep* del subconjunto. Para FD001 y FD003 la ventana de tiempo es de 30, para FD002 es 21 y para FD004 es de 19.

5.2. Modelo generativo de entrenamiento adversario semi-supervisado con regularización de modos para series temporales

El modelo descrito en la sección 4 es aplicado sobre el conjunto de datos C-MAPSS, para ello es necesario definir y ajustar los hiperparametros que lo controlan. Esto se hace mediante un proceso de optimización o *grid search* , en donde, se busca encontrar el conjunto óptimo de hiperparametros para el modelo. La métrica de desempeño utilizada es la función 4.5, la cual se valida mediante validación cruzada.

Los valores considerados para los distintos parámetros en el proceso de optimización son los siguientes:

- Learning rate discriminador: {0.01, 0.001, 0.0001, 0.005, 0.0005}
- Learning rate generador: {0.01, 0.001, 0.0001, 0.005, 0.0005 }
- Número de neuronas 1era capa discriminador: { 256, 128, 64 }
- Número de neuronas 2da capa discriminador: {128, 64, 32 }
- Número de neuronas 1era capa generador: { 32, 64, 128 }

- Número de neuronas 2da capa generador: {32, 64 }
- Número de neuronas encoder: { 32, 64, 128 }

Este proceso de optimización es aplicado para cada subconjunto de datos, esto debido a las diferentes arquitecturas que presentan. Los parámetros óptimos para cada uno son presentados en la siguiente tabla 5.2.

Tabla 5.2: Parámetros óptimos. Elaboración propia.

Data	Learning rate Discriminador	Learning rate Generador	N° neuronas 1era capa discriminador	N° neuronas 2da capa discriminador	N° neuronas 1era capa generador	N° neuronas 2da capa generador	N° neuronas Encoder
FD001	0.001	0.001	64	32	64	32	64
FD002	0.001	0.001	128	64	128	64	128
FD003	0.005	0.005	64	32	64	32	64
FD004	0.0001	0.0001	256	128	256	128	256

A modo de ilustrativo se detalla la arquitectura de las distintas redes desarrolladas para el conjunto FD002. La arquitectura del generador es mostrada en la figura 5.4. Como se aprecia la red LSTM posee dos capas, una con 128 neuronas y la siguiente con 64. Se entiende que esta configuración se conserva hasta las capas finales a lo largo del eje temporal, para luego extraer el espacio latente en la última capa del paso de tiempo final, el cual ingresa a una capa totalmente conectada con una función de activación tangente hiperbólica, la salida corresponde a una trayectoria generada con la misma dimensión que las del conjunto de datos.

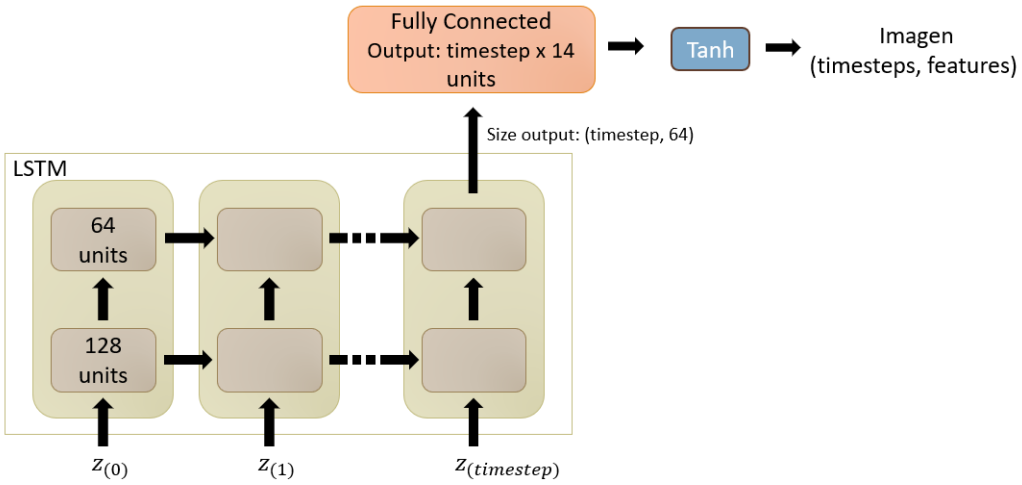


Figura 5.4: Arquitectura de Generador. Fuente: Elaboración propia.

La arquitectura del discriminador es presentada en la figura 5.5, al igual que para el caso del generador la red LSTM tiene un determinado número de entradas dado la ventana de tiempo de cada sub conjunto de datos. Las trayectorias son procesadas y se extrae el espacio latente de estas en la última capa correspondiente al último tiempo de la secuencia, con esta información se realizan dos

procesos distintos, uno de ello es clasificar la información con el fin de determinar la procedencia de las trayectorias ingresadas, es decir, si son generadas por el generador o si vienen del conjunto de datos reales. Para ello el espacio latente ingresa de manera consecutiva a dos capa totalmente conectadas reduciendo su dimensionalidad a 1, en donde, dicho valor determinara si la trayectoria es real o falsa. Por otra parte, el espacio latente se utiliza para la predicción del RUL como parte del aprendizaje semi-supervisado, para ello se realiza un agrupamiento global promedio, aplastando la matriz de salida a un valor por paso de tiempo para que posteriormente entre a una capa totalmente conectada que realiza la predicción del RUL.

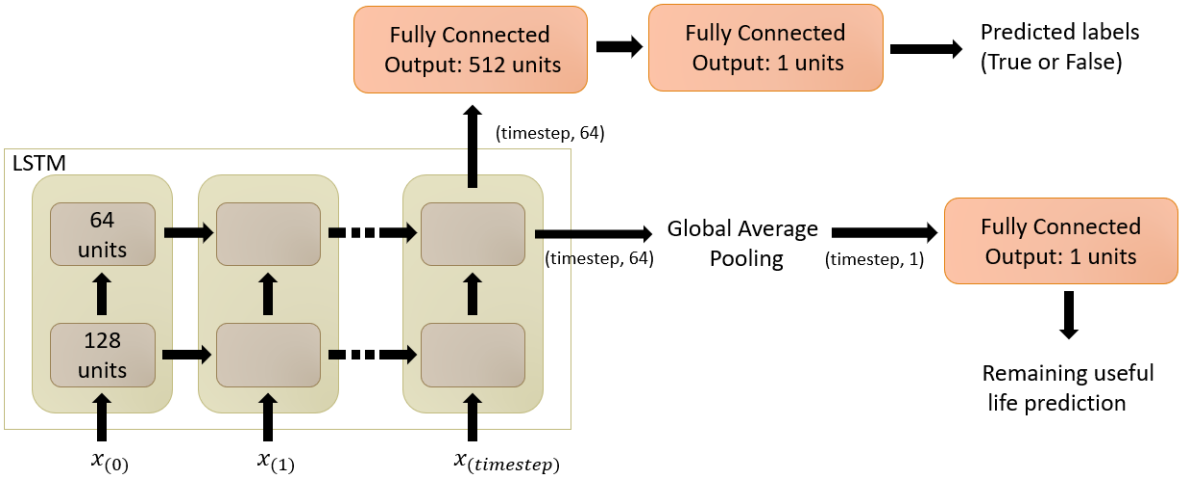


Figura 5.5: Arquitectura de Discriminador. Fuente: Elaboración propia.

En cuanto a la arquitectura del Enconder, se presenta en la figura 5.6. Básicamente, capta el espacio latente de las trayectorias provenientes del conjunto de datos con el objetivo de que el generador la reconstruya y así tener una estimación del error de reconstrucción de dichas trayectorias.

El Encoder es una red LSTM de una sola capa y de largo dado la ventana de tiempo de los inputs, la dimensión del espacio de salida debe ser la misma que de la variable z generada por la distribución a priori $P_z(z)$. Para ello, la salida de la LSTM ingresa a una capa totalmente conectada, la cual proyecta los datos a la dimensión requerida.

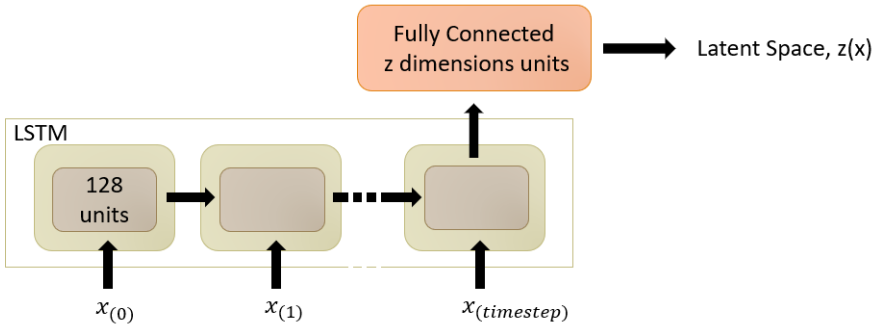


Figura 5.6: Arquitectura de Encoder. Fuente: Elaboración propia.

5.3. Presentación de resultados

En esta sección se mostrarán los resultados obtenidos tras aplicar la metodología mostrada en la imagen 4.2 sobre el conjunto de datos C-MAPSS y se compararán con el estado del arte del área en cuestión.

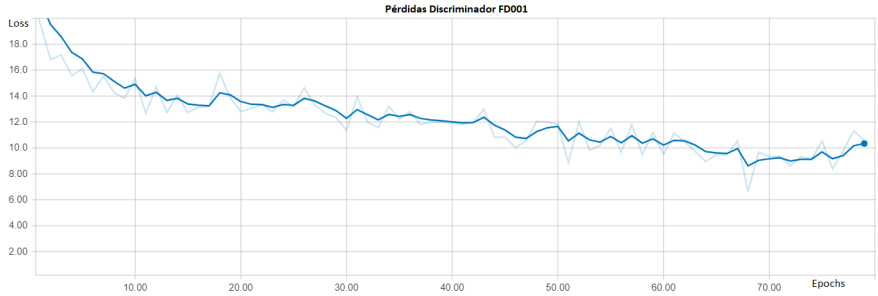
La tabla 5.3 muestra un resumen de los resultados obtenidos dado el porcentaje de etiquetas reales utilizadas en el proceso de entrenamiento, se destacan en verde los mejores resultados. Cabe destacar que el complemento al porcentaje de etiquetas reales es generado por el modelo generativo con el fin de utilizar todos los datos en el entrenamiento de la MLP, por lo tanto, en la obtención del RUL se utiliza un $100\,\%$ de etiquetas, por ejemplo, se utiliza el $20\,\%$ de etiquetas reales y el $80\,\%$ de etiquetas generadas.

Tabla 5.3: Resultados. Elaboración propia.

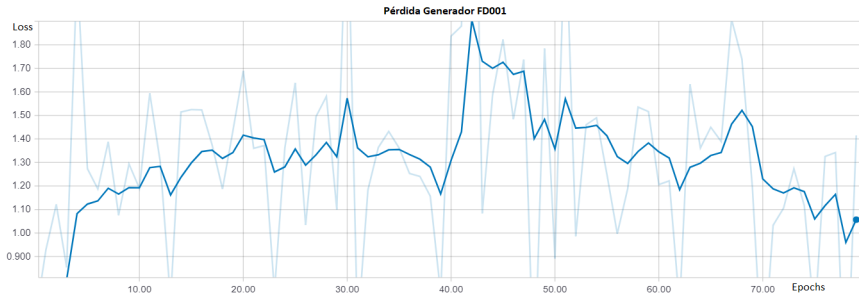
Data	% Etiquetas reales	RMSE	Score
FD001	2 %	12,92	260,98
	10 %	12,71	257,81
	20 %	**12,14**	**228,03**
FD002	2 %	18,95	3340,58
	10 %	18,72	3250,39
	20 %	**18,54**	**2521,88**
FD003	2 %	12,54	240,47
	10 %	12,25	264,79
	20 %	**12,12**	**206,81**
FD004	2 %	22,96	4529,16
	10 %	**21,80**	**2762,53**
	20 %	22,78	3596,20

A continuación se presentan los gráficos que cuantifican las pérdidas para el proceso de entrenamiento del modelo MRLSTM GANs semi supervisado, sólo se ilustran las pérdidas correspondientes a los mejores resultados por sub conjunto de datos.

Para el caso del sub conjunto FD001, las pérdidas tanto del discriminador y generador se muestran en la imagen 5.7 a) y b) respectivamente. El eje Y en ambos gráficos representa las pérdidas, mientras que el eje X las épocas de entrenamiento. Para el sub conjunto FD002, las pérdidas del discriminador y generador de ilustran en la figura 5.8 a) y b) respectivamente.

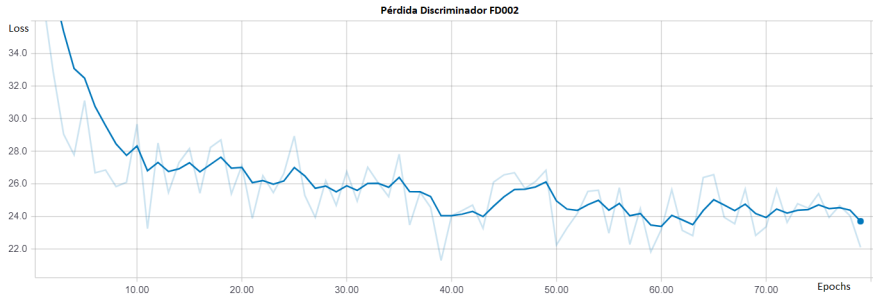


(a) Pérdida Discriminador para conjunto FD001

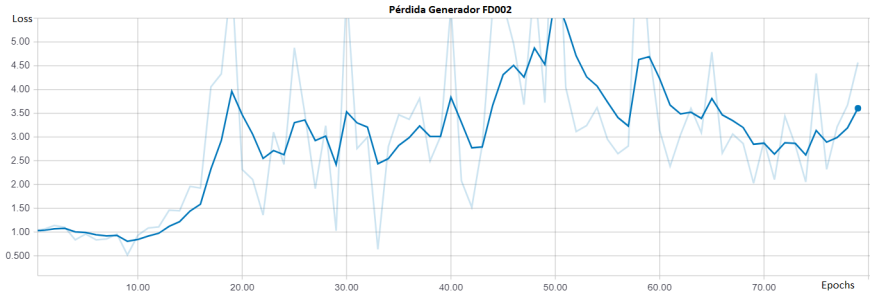


(b) Pérdida Generador para conjunto FD001

Figura 5.7: Pérdidas proceso de entrenamiento MRLSTM GANs semi supervisado para FD001



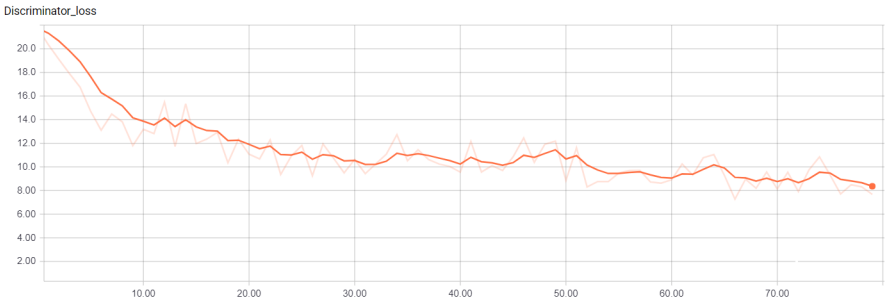
(a) Pérdida Discriminador para conjunto FD002



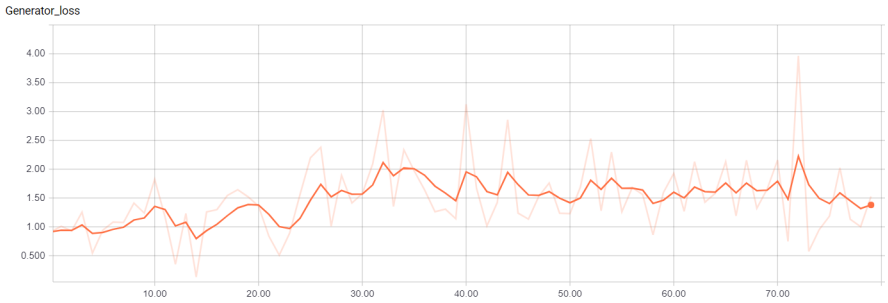
(b) Pérdida Generador para conjunto FD002

Figura 5.8: Pérdidas proceso de entrenamiento MRLSTM GANs semi supervisado para FD002

Para el sub conjunto FD003 y FD004, las pérdidas del discriminador y generador se muestran en la figura 5.9 y 5.10 respectivamente.

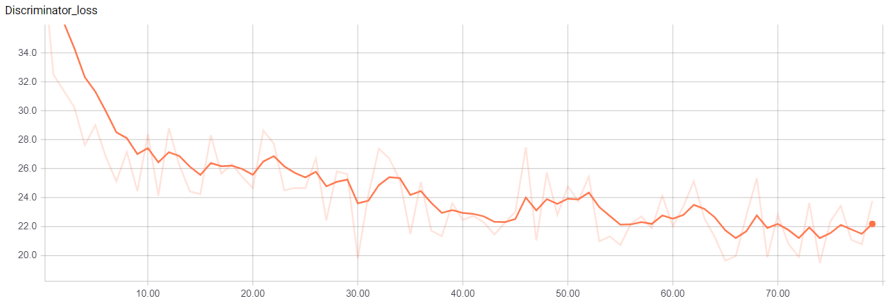


(a) Pérdida Discriminador para conjunto FD003

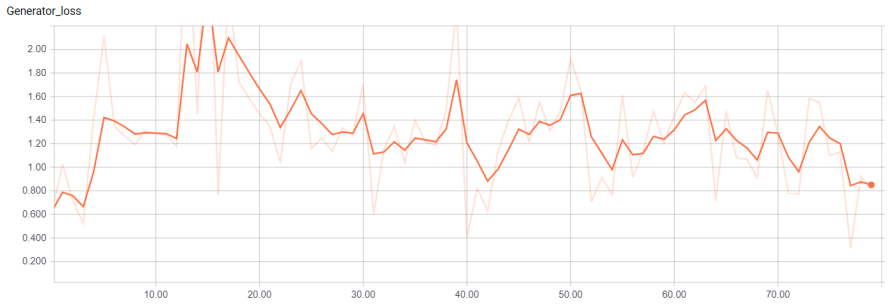


(b) Pérdida Generador para conjunto FD003

Figura 5.9: Pérdidas proceso de entrenamiento MRLSTM GANs semi supervisado para FD003



(a) Pérdida Discriminador para conjunto FD004



(b) Pérdida Generador para conjunto FD004

Figura 5.10: Pérdidas proceso de entrenamiento MRLSTM GANs semi supervisado para FD004

A continuación se muestran las pérdidas para el proceso de entrenamiento de la MLP totalmente conectada, nuevamente solo se mostraran los gráficos correspondientes a los mejores procesos de

entrenamiento, ya que, en general todas las curvas del proceso son similares. En las figuras 5.11 y 5.12 se muestran las pérdidas para los sub conjuntos FD001 y FD002 respectivamente.

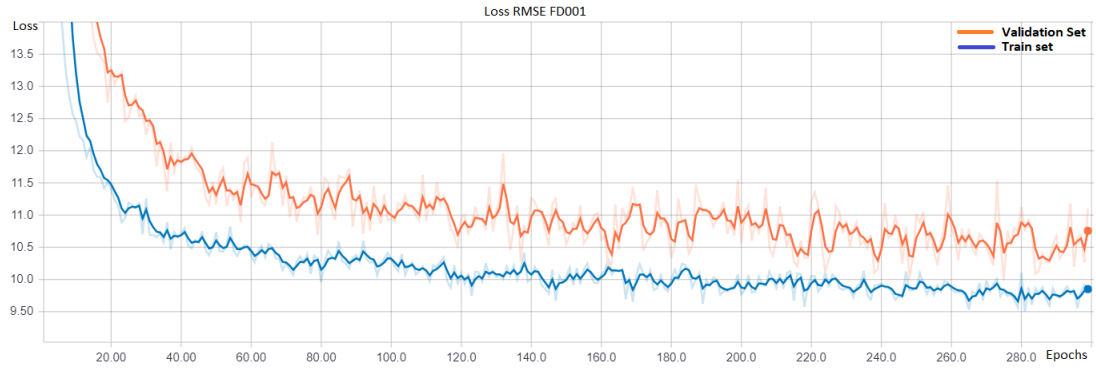


Figura 5.11: Pérdida RMSE para sub conjunto FD001. Fuente: Elaboración propia.

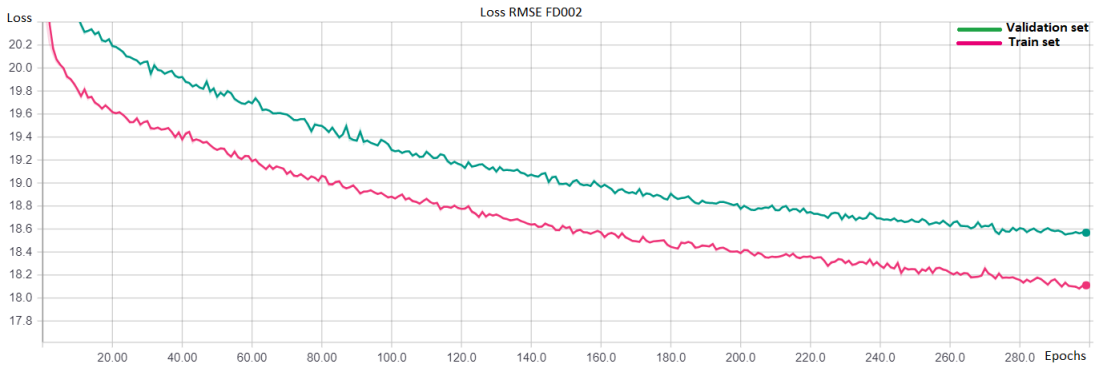


Figura 5.12: Pérdida RMSE para sub conjunto FD002. Fuente: Elaboración propia.

En las figuras 5.13 y 5.14 se muestran las pérdidas en el proceso de entrenamiento de la MLP para los sub conjuntos FD003 y FD004 respectivamente.

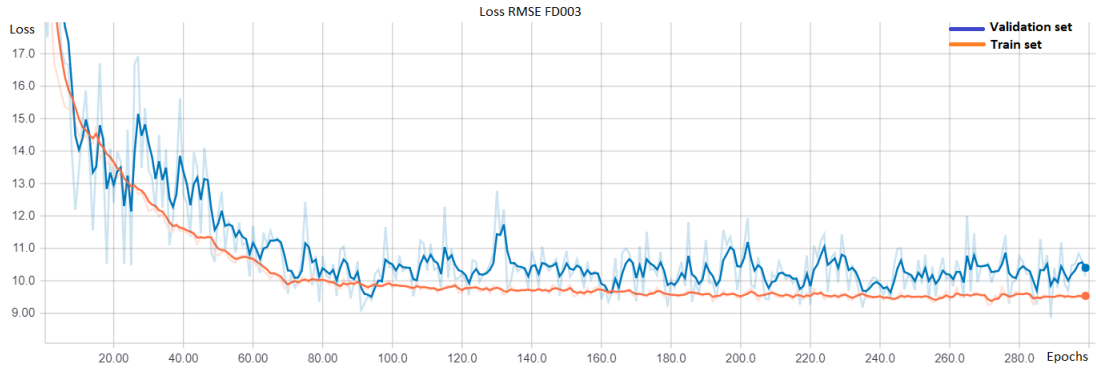


Figura 5.13: Pérdida RMSE para sub conjunto FD003. Fuente: Elaboración propia.

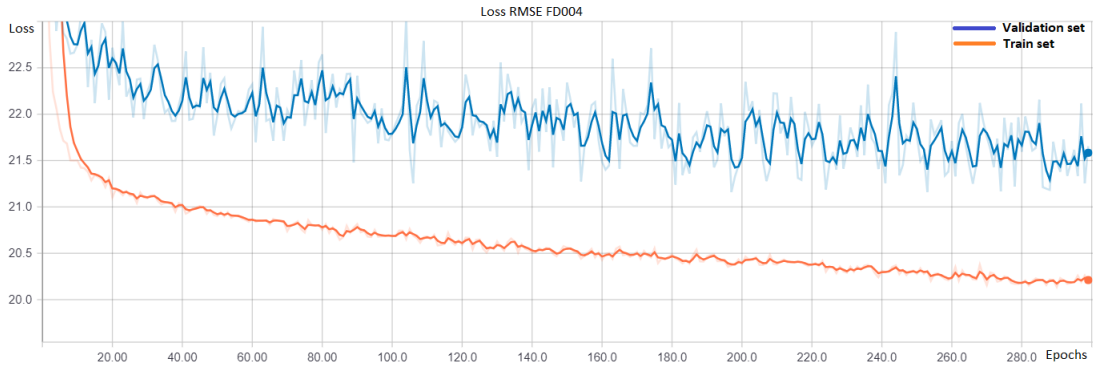


Figura 5.14: Pérdida RMSE para sub conjunto FD004. Fuente: Elaboración propia.

A modo de comparar con otros autores, a continuación se muestra una tabla resumen con los mejores resultados obtenidos por el modelo propuesto en el presente trabajo de tesis (Destacados en verde en la tabla 5.3) y los resultados obtenidos por otros modelos propuestos por diferentes autores.

Tabla 5.4: Desempeño de diferentes modelos sobre C-MAPSS. Elaboración propia.

Data	DLSTM [24]		DCNN[4]		MRLSTM GANs	
	RMSE	Score	RMSE	Score	RMSE	Score
FD001	16,14	338	12,61	273,70	12,14	228,03
FD002	24,49	4450	22,36	10412	18,54	2521,88
FD003	16,18	852	12,64	284,10	12,12	206,81
FD004	28,17	5550	23,31	12466	21,80	2762,53

5.4. Análisis e interpretación de los resultados

En cuanto a la primera etapa del presente trabajo de memoria correspondiente al entrenamiento del modelo propuesto MRLSTM GANs, al observar los gráficos de pérdidas durante el proceso de entrenamiento, tanto el discriminador como generador presentan tendencias en cuanto a su convergencia.

En las imágenes 5.7, 5.8, 5.9 y 5.10 se aprecia que para el caso del discriminador, todas las curvas correspondientes a éste decaen a medida que se entrena el modelo. Naturalmente, se busca que dichas pérdidas tiendan a cero, lo cual nos indica que el discriminador identifica de manera correcta la procedencia de los datos que le están ingresando. Esto es, ya que al momento de calcular la pérdida total del discriminador se hace una diferencia entre términos provenientes del conjunto de datos y de los términos generados por el generador, lo cual se aprecia en la ecuación 3.1. En donde, para el caso de las trayectorias provenientes del conjunto de datos se realiza entropía cruzada entre el valor que el discriminador da a la trayectoria en cuestión y la etiqueta de valor 1. Así mismo, para el caso en que la trayectoria proviene del generador, el discriminador realiza una entropía cruzada entre el valor calculado y una etiqueta de 0. En otras palabras, el discriminador intenta asignar valor de 1 para trayectorias reales y de 0 para generadas, por lo que, a medida que entrena cada vez se va acercando más a una perdida nula.

Luego, lo anterior no sucede en los resultados expuestos y esto es debido a que nuestro discriminador, aparte de clasificar también tiene la capacidad de predecir el RUL gracias al porcentaje de etiquetas que le ingresan. Es por ello que en el caso de las imágenes 5.7 y 5.9 a) las pérdidas del discriminador tienden a valores cercanos a 12, el cual prácticamente es el valor del error obtenido desde la función de optimización en el problema de estimar el RUL que por su parte se asemeja al encontrado tras el entrenamiento de la MLP para los mismos subconjuntos de datos. Lo mismo sucede para 5.8 y 5.10 a), en donde, la pérdida del discriminador tiende a valores cercanos a 20. Por lo que, la pérdida que presenta el discriminador en el entrenamiento de cada sub conjunto de datos tiene un comportamiento razonable, complementando además que cae de manera gradual evitando de esta manera que clasifique todo como falso o como verdadero durante fases tempranas en el proceso de entrenamiento.

Por otra parte, las pérdidas del generador tal como se aprecian en las figuras 5.7, 5.8, 5.9 y 5.10 b), tienen una tendencia ascendente con cierto carácter asintótico que da cuenta de la convergencia de éstas. Lo anterior es de esperar, ya que el discriminador asigna valores altos a las imágenes que considera reales y como la pérdida del generador es la entropía cruzada entre la imagen generada y 0, a medida que se entrena el modelo la pérdida del generador es más y más alta (el discriminador comienza a identificar como reales las imágenes generadas) alcanzando un punto en el cual exista poca variación entre una época y otra.

En cuanto al entrenamiento de la MLP totalmente conectada, no existe mayor dificultad en el

proceso de entrenamiento, en general, las pérdidas ilustradas en las figuras 5.11, 5.12, 5.13 y 5.14 no presentan sobreajuste o como se conoce en inglés overfitting. Se puede observar en las imágenes 5.11 y 5.13 que las curvas de entrenamiento y validación se encuentras próximas entre sí, lo que nos da cuenta de que los subconjuntos FD001 y FD003 son fáciles de entrenar, por lo que al modelo no le cuesta mayor trabajo encontrar los parámetros adecuados para cada conjunto de datos. Pero no así para el caso correspondientes a las imágenes 5.12 y 5.14, en donde se aprecia que las curvas de entrenamiento y validación se encuentran con una mayor separación, dejando en evidencia que los sub conjuntos FD002 y FD004 son más complicados de capturar sus características, siendo de esta manera congruente con la estructura de los datos considerando que están sometidos a distintas condiciones de operación y modos de fallas.

Si se observa la tabla 5.3, se da una tendencia de que a mayor porcentaje de etiquetas reales, mejores son los resultados. Lo cual, no es del todo cierto, ya que la GANs esta sujeta a mucha inestabilidad en su proceso de entrenamiento, por lo que entre una epoca y otra pueden variar los resultados y así obtener diferencias en la predicción del RUL. Este fenómeno se observa en las imágenes correspondientes a las pérdidas del discriminador para cada uno de los sub conjunto de datos, en donde se muestra la inestabilidad o la variabilidad de las pérdidas. Por lo que, la precisión en la predicción del RUL de cierta manera esta ligada a la inestabilidad de la GANs. A pesar de esto, los resultados expuestos en la tabla 5.3 no presentan gran variabilidad.

Lo que si se observa durante el entrenamiento de la MLP, es que esta se comporta de mejor manera y por lo tanto se obtienen mejores resultados al trabajar con las etiquetas provenientes del espacio latente del modelo generativo. En general a mayor porcentaje de etiquetas generadas en comparación con las reales se hacen estimaciones del RUL más precisas. Lo anterior no puede estar tan claro tras ver la tabla 5.3, ya que se da la tendencia de que a medida que aumentan las etiquetas reales mejoran los resultados pero aún así predominan las etiquetas generadas sobre las reales, cuando esta diferencia entre porcentaje de etiquetas disminuye y/o el porcentaje de etiquetas reales sobrepasa al de las generadas el modelo se comienza a comportar de peor manera, obteniendo resultados menos precisos que para el caso estudiado.

En la tabla 5.4 se muestran los resultados obtenidos versus los resultados de otros autores, se aprecia que en todos los casos nuestro modelo MRLSTM GANs tiene un mejor desempeño en comparación con los demás. En algunos sub conjuntos de datos la diferencia del RMSE no es significativa pero considerando que nuestro modelo es semi-supervisado y que a lo más se ocuparon un 20 $\%$ de etiquetas para el entrenamiento los resultados obtenidos son satisfactorios.

6. Conclusiones

En el presente trabajo se desarrollaron algunos métodos de aprendizaje de maquinas para la predicción de fallas, en particular, se desarrolló un modelo generativo de entrenamiento adversario semi supervisado basados en series de tiempo, el cual fue entrenado para extraer el espacio latente de los datos a estudiar (C-MAPSS) y además para la generación de etiquetas para los distintos sub conjunto de datos dentro del conjunto principal. Por otra parte, se desarrolló una red perceptrón multicapa totalmente conectada utilizada para la estimación de la vida remanente de los equipos teniendo como input el espacio latente obtenido del modelo propuesto, la MLP es cuantificada mediante las funciones de pérdidas detalladas en la sección 4.1.

La automatización en la lectura y procesamiento de datos, ya sea mediante técnicas de aprendizaje supervisado, no supervisado o una mezcla de ambos, son una gran herramienta para identificar, etiquetar y clasificar las características que representan a cada conjunto de datos. Lo cual a su vez posibilita a los modelos predecir eventos a partir del espacio latente de los datos. Permitiendo así, realizar un análisis de manera precisa y efectiva tras el previo procesamiento de los datos.

El estabilizar las pérdidas en el proceso de entrenamiento de la MRLSTM GANs, ya sea por los "trucos"mencionados por [8], por la regularización de modos o por las etiquetas entregadas al modelo, aseguró una correcta convergencia de las funciones de pérdidas de manera estable y gradual. Por lo que, cada una de las consideraciones tomadas fue útil para un correcto desempeño del modelo, viéndose claramente reflejado en los resultados para los distintos sub conjuntos de datos.

El modelo propuesto tuvo gran desempeño con resultados a nivel con el estado del arte en el área, con la particularidad de que los modelos presentados por [4] y [24] son supervisados, mientras que la MRLSTMGANs es semi supervisada. Desde el punto de vista de la industria, recopilar datos hasta la falla tiene altos costos asociados, tanto por los equipos de medición como por los activos que resultan dañados. Es en este sentido, el modelo propuesto presenta una ventaja, ya que no es necesario tener una gran cantidad de mediciones hasta la falla para realizar las predicciones al ser de aprendizaje semi supervisado. Permitiendo de esta manera reducir gastos y el tiempo empleado en la obtención de datos, la cual puede llegar a durar años.

Un trabajo a futuro es implementar la predicción de la vida remanente en tiempo real y así dar al usuario una estimación minuto a minuto del desgaste de sus equipos. Para lo anterior, habría que tener en consideración y dar énfasis al pre procesamiento de datos, en particular para la normalización de los datos, ya que no se contaría con un valor máximo o mínimo para realizarla.

Bibliografía

[1] J. Deutsch and D. He, "Using deep learning-based approach to predict remaining useful life of rotating components," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 1, pp. 11–20, 2018.

[2] Tian, Z., Wong, L., & Safaei, N. A neural network approach for remaining useful life prediction utilizing both failure and suspension histories. *Mechanical Systems And Signal Processing*, 24(5), 1542-1555, 2010.

[3] Jayasinghe, L., Samarasinghe, T., Yuen, C., & Ge, S.S. Temporal Convolutional Memory Networks for Remaining Useful Life Estimation of Industrial Machinery. *CoRR*, abs/1810.05644, 2018.

[4] Li, X., Ding, Q., & Sun, J. Remaining useful life estimation in prognostics using deep convolution neural networks. Reliability Engineering & System Safety, 172, 1-11. 2018.

[5] Z. Tian, "An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring," *J. Intell. Manuf.*, vol. 23, no. 2, pp. 227– 237, Apr. 2012.

[6] L. Ren, J. Cui, Y. Sun, and X. Cheng, "Multi-bearing remaining useful life collaborative prediction: A deep learning approach," *J. Manuf. Syst.*, vol. 43, pp. 248–256, Apr. 2017.

[7] M. Yuan, Y. Wu, and L. Lin, "Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network," in *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, 2016, pp. 135–140.

[8] I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. In *Neural Information Processing Systems*, Dec. 2016.

[9] J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, 1951.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

[11] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., Bharath, A. Generative Adversarial Networks: An Overview. *IEEE Signal Processing Magazine*, 35(1), 53-65, 2017.

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

[13] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proceedings of the 5th International Conference on Learning Representations (ICLR) - workshop track*, 2016.

[14] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.

[15] Tong Che, Li Yanran, Athul Paul Jacob, Yoshua Bengio Wenjie Li. Mode Regularized Generative Adversarial Networks. arXiv:1612.02136, 2016

[16] Olof Mogren. C-RNN-GAN: Continuous recurrent neural networks with adversarial training. 29 November 2016.

[17] Jürgen Schmidhuber and Sepp Hochreiter. Long short-term memory. *Neural computation*, 7(8):1735–1780, 1997.

[18] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

[19] Esteban, C., Hyland, S.L., & Rätsch, G. (2017). Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs. *CoRR*, abs/1706.02633.

[20] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[21] L. Peel, "Data driven prognostics using a Kalman filter ensemble of neural network models," in 2008 *International Conference on Prognostics and Health Management*, 2008.

[22] Saxena A, Goebel K, Simon D, Eklund N. "Damage propagation modeling for aircraft engine run-to-failure simulation". *Int. conf. prog. heal. manag. PHM* 2008.

[23] H. Richter, "Engine models and simulation tools," in *Advanced control of turbofan engines*, pp. 19–33, 2012.

[24] Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. Long Short-Term Memory Network for Remaining Useful Life estimation. *2017 IEEE International Conference On Prognostics And Health Management* (ICPHM) (2017).