UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES E INFORMÁTICOS

TEMA:

DESARROLLO DE UNA RED NEURONAL DIFUSA PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo la obtención del título de Ingeniero en Sistemas Computacionales e Informáticos.

ÁREA: Software

LÍNEA DE INVESTIGACIÓN: Inteligencia Artificial

AUTOR: Alexis Roberto Jarrín Núñez

TUTOR: Ing. Mg. Edison Homero Álvarez Mayorga

Ambato – Ecuador

Octubre – 2020

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: DESARROLLO DE UNA RED NEURONAL DIFUSA PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Alexis Roberto Jarrín Núñez, estudiante de la carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, octubre 2020

Firmado electrónicamente por:
EDISON HOMERO
ALVAREZ MAYORGA

.....

Ing. Mg. Edison Homero Álvarez Mayorga

TUTOR

AUTORÍA

El presente Proyecto de investigación titulado: **DESARROLLO DE UNA RED NEURONAL DIFUSA PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL** es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, octubre 2020

.....

Alexis Roberto Jarrín Núñez

C.C: 1804370466

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Alexis Roberto Jarrín Núñez, estudiante de la Carrera de Ingeniería en Sistemas Computacionales e Informáticos, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado DESARROLLO DE UNA RED NEURONAL DIFUSA PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, octubre 2020

Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.

PRESIDENTA DEL TRIBUNAL

Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA

Ing. Franklin Oswaldo Mayorga Mayorga

DOCENTE CALIFICADOR

Firmado electrónicamente por:
RUBEN EDUARDO NOGALES PORTERO

Ing. Rubén Eduardo Nogales Portero

DOCENTE CALIFICADOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y proceso de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, octubre 2020

.....

Alexis Roberto Jarrín Núñez

C.C: 1804370466

AUTOR

DEDICATORIA

El presente trabajo de investigación lo dedico principalmente a Dios que guiándome de su mano me ha permitido tomar las riendas de mi formación académica, a mis Padres que, gracias a su apoyo, a su confianza, a su dedicación por cada uno de sus hijos no lo hubiese logrado, a mis hermanas que siempre han estado ahí cuando lo he necesitado brindándome un consuelo y unas palabras de apoyo. A mi primo y a mi tío que fueron las personas que siempre estuvieron conmigo en los momentos más complicados de mi vida.

Alexis Roberto Jarrín Núñez

AGRADECIMIENTO

Agradezco a Dios y a la vida que me han permitido culminar el camino emprendido con los ojos llenos de ilusión y una mochila cargada de trabajo duro, noches largas y amigos increíbles que durante las largas jornadas educativas estuvieron ahí para brindarme su apoyo, a docentes que realmente han sido una inspiración dentro de la Facultad para lograr cumplir mis objetivos.

Agradezco a cada familiar que siempre estuvo pendiente al haber iniciado este camino y por haberme brindado su apoyo cuando más lo necesite.

Agradezco Al Ing. Edison Álvarez que ha sido el precursor durante el desarrollo de esta investigación.

Alexis Roberto Jarrín Núñez

ÍNDICE GENERAL DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

Uno de los puntos más importantes para la realización de esta investigación, es su escasa relevancia dentro del país, a pesar de brindarnos herramientas útiles y productivas para automatizar procesos dentro de la industria, evitando perdidas innecesarias tanto en producción, partes, piezas y sobre todo pérdidas económicas, el motivo de la realización de este proyecto radica en la industria, específicamente en industrias que utilizan motores rotativos trifásicos de corriente alterna, ayudando así a mantener un equilibrio dentro de la producción y el excelente estado físico y funcional de la maquinaria, brindando seguridad y confianza a los operadores de la misma.

El presente trabajo es una investigación con el aval del departamento de Investigación de la Facultad de Ingeniería en Sistemas Computacionales e Informáticos de la Universidad Técnica de Ambato, pretende ser de utilidad para abrir camino a la investigación de sistemas con redes neuronales artificiales basados en lógica Difusa, para poder aprovechar las utilidades de las herramientas informáticas para conocer el estado de un motor de corriente altera en tiempo real, permitiendo así a los operadores planificar la producción, así como, los días de mantenimiento de la maquinaria brindando soluciones inmediatas y evitando perdidas en producción.

Al aplicar lógica Difusa mediante redes neuronales artificiales, estamos dando paso a la Inteligencia Artificial y la infinidad de soluciones que esta nos puede brindar, en el caso específico de esta investigación podemos trabajar en tiempo real y con muestras que sobrepasan la capacidad humana de realizar este procesamiento, con esto los beneficiarios de este tipo de proyectos pasan a ser las empresas que implementan este tipo de soluciones para cuidar su producción y sus pérdidas económicas.

Palabras clave: Red Neuronal, Lógica Difusa, Motores Trifásicos, Fallos.

ABSTRACT

One of the most important points for the conduct of this research, is its low relevance within the country, despite providing us with useful and productive tools to automate processes within the industry, avoiding unnecessary losses in both production, parts, parts and above all economic losses, the reason for the realization of this project lies in the industry, specifically in industries that use three-phase rotary motors of alternating current , thus helping to maintain a balance within the production and excellent physical and functional condition of the machinery, providing safety and confidence to the operators of the machinery.

This work is research with the endorsement of the Research department of the Faculty of Engineering in Computer and Computer Systems of the Technical University of Ambato, aims to be useful to make way for the research of systems with artificial neural networks based on Diffuse logic, to be able to take advantage of the utilities of computer tools to know the state of an altera current engine in real time, thus allowing operators to plan production as well as machinery maintenance days providing immediate solutions and avoiding production losses.

By applying Diffuse logic through artificial neural networks, we are giving way to Artificial Intelligence and the myriad of solutions that it can provide us, in the specific case of this research we can work in real time and with samples that exceed the human capacity to carry out this processing, with this the beneficiaries of this type of projects become the companies that implement this type of solutions to take care of their production and their economic losses.

Keywords: Neural Network, Diffuse Logic, Three-Phase Motors, Faults.

INTRODUCCIÓN

El presente trabajo de investigación denominado: “DESARROLLO DE UNA RED NEURONAL DIFUSA PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL”, se desarrolla mediante los siguientes capítulos:

Capitulo I.- Marco Teórico, dentro del capítulo uno se presenta los antecedentes investigativos en donde se sustenta la base de esta investigación, además de citar ejemplo que servirán para obtener un amplio conocimiento previo.

Capitulo II.- Metodología, dentro del capítulo dos se presenta los materiales y métodos a utilizar para desarrollar la investigación de una manera práctica, además de proceder a la recolección de información y por último el procesamiento y análisis de los datos.

Capitulo III. Resultados y Discusión, una de las partes fundamentales del proyecto de investigación realizado, en donde, se discute los resultados presentados dentro de la investigación y si corresponden a la metodología utilizada.

Capitulo IV.- Conclusiones y recomendaciones, se presentan las conclusiones del proyecto una vez llegado a su fin, y después de haber procesado los resultados obtenidos, además, se presenta recomendaciones importantes para una futura investigación.

Capitulo V.- Materiales de Referencia, Se presenta la bibliografía utilizada para la realización de la investigación.

CAPÍTULO I

MARCO TEÓRICO

1.1 Tema de Investigación

“DESARROLLO DE UNA RED NEURONAL DIFUSA PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL”

1.2 Antecedentes Investigativos

En base a una búsqueda en los repositorios y biblioteca de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, después de revisar distintas fuentes bibliográficas como: Tesis, artículos científicos, revistas científicas no se encontraron trabajos de investigación similares.

Sin embargo, en el repositorio de la Universidad de las Fuerzas Armadas ESPE, Departamento de Eléctrica y Electrónica, Carrera de Ingeniería Electrónica, Automatización y Control, se encontró un trabajo desarrollado en el año 2018 bajo el título: “Diseño y simulación de un controlador neuro-difuso ANFIS para un convertidor dc-dc zeta”, Elaborado por el Sr. Zapata Sinaluisa Jonathan Alexis, el cual indica lo siguiente:

“…El controlador neuro-difuso ANFIS tiene la ventaja del conocimiento experto de un sistema de inferencia difusa y la capacidad de aprendizaje de las redes neuronales, en donde el modelo dinámico no lineal del convertidor ZETA y el controlador ANFIS se derivan para desarrollar un modelo de simulación que demuestran que el controlador ANFIS tiene un mejor desempeño, una buena regulación de voltaje con menor impulso, error de estado estable cuando se le somete a variaciones en el voltaje de entrada, carga o referencia”. [1]

En este trabajo de titulación se entiende que el sistema neuronal difuso mediante un controlar ANFIS logra regular el voltaje de salida en relación con su referencia, carga o entrada, que en comparación al convertidor Zeta tiene una mejor regulación de voltaje un error de estado estable y mejores tiempos de establecimiento de acuerdo a como entre el voltaje o los cambios y anomalías que se puedan producir durante el proceso.

Así mismo en los repositorios de Escuela Politécnica Nacional, Facultad de Ingeniería Mecánica, se encontró un trabajo desarrollado en el año 2017 bajo el título: "Diseño, desarrollo e implementación de un sistema adaptativo neuro-difuso aplicado al brazo robot Mitsubishi RV-2AJ con visión artificial, utilizando un controlador basado en procesador ARM", elaborado por la Srta. Celi Sánchez Carmen Johanna, en el cual se habla lo siguiente:

"…El proceso se basa en una cámara que adquiere la imagen del patrón y la envía a un procesador de 32 bits ARN conocido como RaspberryPi, esta tarjeta a través de las técnicas de visión artificial realiza el pre-procesamiento de las imágenes y calculas los momentos invariantes de HU, que sirven en el siguiente proceso como enteradas a la red neuronal. Estos procedimientos inteligentes entregan al sistema la clasificación de los patrones y con el algoritmo neuro-difuso se establece la velocidad de penetración de la herramienta..." [2]

En este trabajo se habla de la implementación de un algoritmo basado en inteligencia artificial, en donde debe tener la capacidad de diferenciar los patrones mediante el preprocesamiento de imágenes que sirven de entrada para la red neuronal y con el algoritmo neuro-difuso se logra realizar el proceso de maquilado sobre el patrón especifico.

En el año 2019 el Sr. Delgado Guerrero Jonathan Stalin de la Universidad de Guayaquil realizaron el proyecto denominado: "Prototipo de un Sistema Web Inteligente basado en redes neuronales difusas para medir el estado cognitivo del Estudiante en la asignatura Simulación de Sistemas por parte del Docente de la Cátedra", en el que se detalla lo siguiente:

"… Se desarrolló una aplicación con tecnología open source y módulos de inteligencia artificial con la aplicación de lógica difusa y redes neuronales, en donde se puede obtener de forma más rápida el estado cognitivo del estudiante…"[3]

Con el desarrollo de este sistema web con redes neuronales y lógica difusa se logra obtener diferentes datos del estudiante que servirán para optimizar el estado cognitivo del estudiante mediante un valor de salida que se referenciarán entre sí.

Contextualización del problema

El uso de redes neuronales difusas a nivel mundial ha ido en aumento, la han utilizado en pronósticos de precio de la energía, así como en el uso de problemas de clasificación y modelización, el uso de estas redes hoy en día ayudan a detectar algunos problemas como el citado por Monzón Jorge – Pisarello María en el 2015 en el que trata sobre "La Identificación de latidos cardiacos anómalos bajo lineamientos de una arquitectura de red neuronal difusa y se denomina Sistema de Inferencia Difuso basado en Redes Adaptativas o ANFIS", en donde la arquitectura propuesta sirve como base para la elaboración automática de conocimientos en la forma de reglas difusas IF-THEN, en el cual el sistema puede agrupar los pares de datos deseados de entrada – salida para identificar los latidos cardiacos anómalos en tiempo real mediante señales electrocardiográficas. [4]

En el Ecuador las redes neuronales difusas han sido poco tratadas debido a que no se realizan muchos estudios y aplicaciones basadas en estas redes, el caso más relevante ha sido: "Algoritmo neuro-difuso para la detección y clasificación de fallas en línea de transmisión eléctrica del sistema ecuatoriano usando simulaciones y datos de registradores de fallas" este algoritmo permite manejar adecuadamente los posibles problemas ante fallos y datos imprecisos usando una línea de transmisión de 230 kV de un sistema eléctrico en potencia, además de dotarlo de conocimiento para un buen aprendizaje y posible solución a situaciones no definidas usando datos tanto reales como simulados. [5]

Dentro de la Provincia de Tungurahua, específicamente en la ciudad de Ambato, el uso de redes neuro difusas se ha implementado muy poco, debido a que, las empresas cuentan con problemas inesperados causando posibles daños en la maquinaria rotativa dejando pérdidas materiales y económicas, debido a esto se implementara una red neuronal difusa que se encargara de la detección de fallos en tiempo real mediante la alimentación de datos obtenidos mediante picos de información de cuando ocurre el fallo.

Fundamentación Teórica

Mantenimiento Industrial

El campo del mantenimiento industrial constituye el eje principal en la industria, se encarga de regular la calidad en la producción, al igual que la cantidad, que a lo largo del tiempo ha sido cambiando y mejorando, es decir, enfocándose en mejorar la calidad de la producción como inversión para mejorar los estándares de la industria.

Historia del mantenimiento

El mantenimiento abarca un campo amplio en donde se aplica métodos y técnicas para mantener el correcto funcionamiento de máquinas, equipos, motores, sistemas y servicios. En la antigüedad, se realizaban actividades que indirectamente se conocían como mantenimiento, como, por ejemplo, coser pieles, afilar armas y herramientas, remedar vestidos, etc.

En la época de la revolución industrial el mantenimiento era correctivo debido a la urgencia de los hechos, las pérdidas de las primeras calderas y los accidentes ocasionaron que se implemente diferentes talleres mecánicos debido a la presión de las aseguradoras enfocándose en mejores cuidados. En 1925 se plantea la necesidad de implantar el manteamiento basando científicamente en la industria americana, nace la idea de realizar los mantenimientos antes de que se produzcan desgaste o roturas, evitando así perdidas e interrupciones en el proceso de fabricación, a este proceso se lo llamo mantenimiento preventivo. [6]

En los años sesenta, el auge de las industrias aeronáutica, espacial y electrónica empiezan a implementar el mantenimiento preventivo, tomando como referencia el estado tanto físico como lógico de un equipo o los componentes que este posee.

En la actualidad el mantenimiento ha entrado en una fase de desarrollo conocida como su tercera generación, en donde se ha implementado sistemas de control, además de equipos con la capacidad de inspeccionar el funcionamiento de maquinaria, en donde se conoce el estado real de cada uno de estos, con la realización de mediciones continuas o mediante el estado de algunos parámetros: temperatura, ultrasonido, endoscopia, ruido, análisis físico, análisis químicos, vibraciones, tecnografía y temperaturas.

Además de inclusión de aplicaciones de sistemas de información que ordenan, organizan, guardan y comparan la información obtenida mediante tratamientos de datos, con esto se llegara a la utilización de sistemas expertos, con un extenso campo de estudio y facilitando los mantenimientos en condiciones adversas.

Objetivos del Mantenimiento

Al mencionar la palabra mantenimiento, esta se la puede conceptualizar como un trabajo de reparación, control, prevención que garantice el correcto funcionamiento de una máquina, de un proceso, de una instalación o de un sistema en general tomando en cuenta cada uno de sus componentes.

En consecuencia, el mantenimiento industrial preventivo, correctivo, predictivo se lo realiza en máquinas, instalaciones industriales móviles, equipos, servicios, sistemas y todo tipo de producto o bien.

Los principales objetivos del mantenimiento industrial se los puede plasmar en los siguientes ítems.

- Controlar, evitar, mejorar, reducir y reparar fallos que se puedan presentar en el transcurso o posterior uso de una maquinaria o cualquier tipo de bien.
- Controlar y evitar cualquier tipo de accidente que se pueda producir debido al mal funcionamiento de algún equipo.
- Reducir la cantidad de fallos y detenciones, además de la gravedad de estos en cualquier tipo de maquinaria o sistema.
- Mantener la calidad de producción de productos o servicios de la maquinaria de acuerdo con las condiciones establecidas que permitan la operabilidad de estas.
- Conservar la seguridad del personal de operaciones.
- Disminuir costos de reparación y producción debido al paro de la maquinaria.
- Conservar y alargar el tiempo de vida útil de maquinaria, bienes o servicios. [6]

Mantenimiento predictivo

El mantenimiento predictivo consta de variables que van relacionando el estado físico de una maquina o equipo con su funcionamiento, monitoreando y dando seguimiento para detectar cualquier tipo de anomalía en alguna de sus partes, evitando así que el daño o la anomalía produzcan daños irreversibles en la maquinaria o sistema.

La característica principal del mantenimiento predictivo tiene como base detectar fallas leves, para así programar un mantenimiento preventivo o correctivo dependiendo la falla que este presente, logrando así el continuo desarrollo de la producción y extendiendo el tiempo de funcionamiento y vida útil de partes, piezas de un equipo o maquinaria.

Al producirse fallos leves y lentos, se puede obtener indicios de un fallo grave o agudo, mediante datos que se recogen cuando el equipo se encuentro en perfecto estado de funcionamiento, algunos parámetros ha tener en cuenta son: ruido, temperatura, velocidad, humedad, vibraciones, temperatura, fluidos, viscosidad, corriente, estado de rodamientos, aceite, etc. Mediante este tipo de mantenimiento podemos dar seguimiento a un fallo, llevando un registro de datos actualizado, donde nos permita programar una parada o reparación del equipo para precautelar el buen funcionamiento de la maquinaria, además de disminuir costos de personal en mantenimiento y producción. [6]

Parámetros aplicados al mantenimiento predictivo

La industria maneja algunos parámetros en cuanto al mantenimiento predictivo, los más utilizados son los siguientes:

Termografía: El análisis termográfico estudia los factores que inciden en el comportamiento de la temperatura en un equipo o maquinaria afectando su funcionamiento, este parámetro ayuda a identificar la falla fácilmente evitando una técnica invasiva y arrojando resultados inmediatos al encontrarse con una temperatura mayor a la que el fabricante recomienda, indicando así, su funcionamiento anormal.

La medición de temperatura se la realiza mediante ondas electromagnéticas, en donde, la temperatura es proporcional a la energía emita, es decir, a mayor calor mayor energía emitida, al tratarse de ondas electromagnéticas que son visibles a simple vista.

Se necesita un equipo que ayude a transformar la energía en imágenes visibles para el ojo humano. [7]

Vibraciones: El análisis de vibraciones se basa en el uso de maquinaria rotativa tomando en cuenta una variación de vibraciones en condiciones normales, cuando empieza a darse problemas en el funcionamiento de la maquinaria el nivel de vibración cambia detectando así un posible fallo en alguno de sus componentes, dando paso a una revisión mediante datos estadísticos, la velocidad de giro, poleas, rodamientos, correas, además tomando en cuenta la fecha de cuando estos fueron recolectados, para realizar este tipo de análisis se requiere de un equipo especializado llamado analizador de vibraciones, este equipo nos ayuda a desglosar los datos recolectados en: frecuencia, amplitud, espectro de vibración, dando paso a los posibles daños como: solturas mecánicas, desgaste de bandas, anomalías eléctricas, daño en rodamientos, mal funcionamiento de bombas, engranes desgastados, problemas mecánicos, desbalance, desalineamiento. [7]

Aceite: El análisis de aceite es una parte fundamental dentro del funcionamiento de una máquina, ya que ayuda a lubricar engranes, controlar la temperatura, determinar el nivel de impurezas. Cuando el aceite va perdiendo sus propiedades tanto químicas como físicas, la maquinaria va presentando fallos debido a su alto grado de contaminación, que está relacionado directamente por: exceso de agua, presencia de impurezas, desgaste de engranes, combustible, material insoluble, etc. La pérdida de propiedades del aceite está relacionada directamente con: la viscosidad, constante dieléctrica, detergencia, logrando determinar mediante pruebas químicas y físicas los tiempos exactos de lubricación y mantenimiento de la maquinaria. [7]

Ultrasonido: El análisis de ultrasonido se basa en el estudio de ondas sonoras con frecuencias que sobrepasan los parámetros normales debido a un mal funcionamiento de la maquinaria, el oído humano es capaz de percibir ondas sonoras entre los 20Hz y 20kHz, por tal razón se necesita un instrumento llamado detector de ultrasonido que sea capaz de convertir estas ondas que se atenúan rápidamente en sonidos audibles, ayudando así a detectar posibles fallos como: fricción, fugas en válvulas, arco eléctrico, fugas de fluidos, perdidas al vacío, las principales aplicaciones de detección de ultrasonido se la realiza en: rodamientos en maquinaria rotativa, turbulencia debido a fugas de presión e inspección de instalaciones eléctricas. [7]

Fallos en maquinaria rotativa

Los fallos en maquinarias rotativas se pueden dar por diversos factores que implican averías leves y graves que con el tiempo pueden desembocar en daños irreversibles, además, paro en la maquinaria rotativa generando grandes pérdidas a nivel de producción, entro los principales modos de falla se puede encontrar el más común, que es la vibración, a medida que un fallo se va presentando la vibración en la maquinaria suele presentar un incremento de esta, a pesar de que las vibraciones son una de las más frecuentes señales dinámicas que presentan la maquinaria rotativa al momento de su fallo, también se debe proceder a la medición de distintos parámetros de acuerdo a las características de cada máquina como por ejemplo, la temperatura, presión, ruido, etc. Una vez tomado este tipo de parámetros de puede implementar un sistema de monitoreo personalizado con un alto grado de certeza.

Fallas en motores de inducción

Fallos por barras rotas (BRB)

Uno de los principales y comunes fallos que se presentan en motores rotativos son las barras, que pueden estar con fisuras o completamente rotas, los principales problemas por los cuales se producen estos fallos pueden ser: sobre esfuerzo, daños de fábrica, esfuerzos térmicos, arranques anormales, máximo esfuerzo mecánico en rodamientos, por lo general este tipo de fallos directamente en la línea de alimentación al generarse una constante armónica en el campo magnético del entrehierro. La componente armónica modifica el espectro de alimentación de corriente asociada directamente con el estado físico sano del motor de inducción. Al presentarse las barras rotas en un motor de inducción, las fallas no causan daños inmediatos, pero presentan efectos secundarios en otros componentes como: excentricidad, aislamiento de piezas mecánicas en el motor, aumento de la temperatura provocando sobrecalentamiento, además de problemas en el devanado. [8]

Si el motor tiene barras rotas, crea condiciones no balanceadas o de asimetría que generan un campo magnético giratorio adicional en retraso, que gira a velocidad de deslizamiento y corta los bobinados del estator, induciendo en ellos un voltaje y una corriente con la misma frecuencia del campo rotatorio, cuya ubicación está dada por:

$$fsb = f1\ (1 \pm 2ks)$$

Donde:

fsb: Frecuencia de bandas laterales debido a las barreras rotas.

$f1$: Frecuencia de la red de alimentación del motor.

k: Valor entero (1,2,3…) depende de la banda de frecuencia que se desea obtener.

s: Desplazamiento.

Fallas por excentricidad

Las fallas que se presentan por excentricidad se produce cuando existe un desbalance en el eje de rotación en relación con el centro de masa de una máquina, provocando que se genere una fuerza armónica actuando perpendicularmente al eje de rotación.

Al generarse esta fuerza armónica que actúa directamente en el entrehierro, donde se aprecia excentricidades que se generan cuando existe excentricidad entre el rotor y el estator en maquinaria rotativa comúnmente en motores eléctricos. Los principales problemas que se presentan por este tipo de excentricidad son: las vibraciones, rodamientos desgastados, mayor vibración, además de rigidez en el rotor. Además, existen diferentes causas que pueden provocar excentricidad en motores rotativos, uno que es muy poco común, es el daño de fabrica es decir viene con defectos de manufactura, además de, rodamientos desgastados, ejes desalineados, causas térmicas, etc. Al entrar en el campo de la excentricidad se puede producir además el fenómeno llamado resonancia que se produce cuando la frecuencia de rotación coincide con cualquier frecuencia natural del sistema convirtiéndose así en un caso muy crítico de excentricidad.[9]

Inteligencia Artificial

La inteligencia artificial (IA) es una ciencia en la que se estudia el comportamiento del cerebro humano y la inteligencia, trata acerca de la automatización mediante métodos matemáticos, actividades como la toma de decisiones, resolución de problemas y el aprendizaje, esto implica que la inteligencia artificial se rodea de misticismo, creencias, esperanzas y teorías que con el apoyo de un correcto estudio, de la tecnología y de la ciencia la humanidad pueda alcanzar a desarrollar genes de clones y autómatas artifíciales. [10]

El campo de la inteligencia artificial no solo implica imitar el comportamiento humano o sistemas de inteligencia artificial tales como un juego de ajedrez o simplemente la toma de decisiones, pero a pesar de todo esto han demostrado que son capaces de resolver problemas complejos.

La inteligencia artificial ha sido parte esencial por los últimos 60 años para avances científicos e intelectuales, el estudio de la información y el reforzamiento del conocimiento, el desafío de la inteligencia artificial incluye desarrollar en maquina la capacidad de almacenar, aprender, percibir información clasificando información que ya se conoce y aprender de lo desconocido para así poder comunicarse utilizando lenguaje humano e interactuar con el entorno físico.

Acorde a la creciente necesidad de desarrollar aplicaciones capaces de plantear mecanismos que permitan llegar a la posible solución de problemas complejos, mediante la ejecución de procesos que se denominan inteligentes, dando lugar a la ingeniería de sistemas inteligentes, mediante esto se puede decir que: "… La inteligencia artificial comprende la investigación científica y tecnológica de los sistemas inteligentes…".[6]

Se denomina un sistema inteligente a un ente que es capaz de percibir, razonar, aprender, adaptarse, tomar decisiones y actuar racionalmente para satisfacer metas en un determinado entorno. Se define como entidad a máquinas, humanos u cualquier otro animal capaz de razonar y tomar decisiones acore a las circunstancias que se presenten.

Tópicos relacionados con Sistemas Inteligentes

Fig. 1.1 Tópicos de Sistemas Inteligentes

Fuente: [6]

Debido a que los sistemas inteligentes abarcan una gran complejidad además del desarrollo de una investigación científica se requiere el apoyo de diferentes áreas del conocimiento como, por ejemplo: Filosofía, Psicología, Ciencias de la Comunicación, Neurociencias, Física, Cibernética, Matemáticas, Electrónica y Comunicaciones por citar algunos ejemplos.

Desde el punto de vista Filosófico, se cree que algunos métodos computacionales son necesarios y suficientes para la inteligencia, no así, John Searle teórico de IA argumenta que estos métodos están fundamentalmente equivocados, denominándole Inteligencia Artificial Fuerte en donde la tesis central es que la mayoría de procesos que realiza el cerebro son idénticos a lo que se realiza en un computador, en donde si el cerebro es capaz de generar conciencia, el computador también debería ser capaz de generar conciencia, es decir que simplemente tratan la sintaxis mas no la semántica, por lo que un computador no logra explicar el fundamento de la intencionalidad o el significado de lo que está procesando. [6]

En cambio, desde el punto de vista Tecnológico los sistemas inteligentes no necesariamente necesitan imitar o emular procesos característicos de los seres inteligentes denominándola, así como Inteligencia Artificial Débil. [6]

Dentro del siguiente mapa mental se presentan los tópicos que trata actualmente la inteligencia artificial basada en el punto de vista Tecnológico:

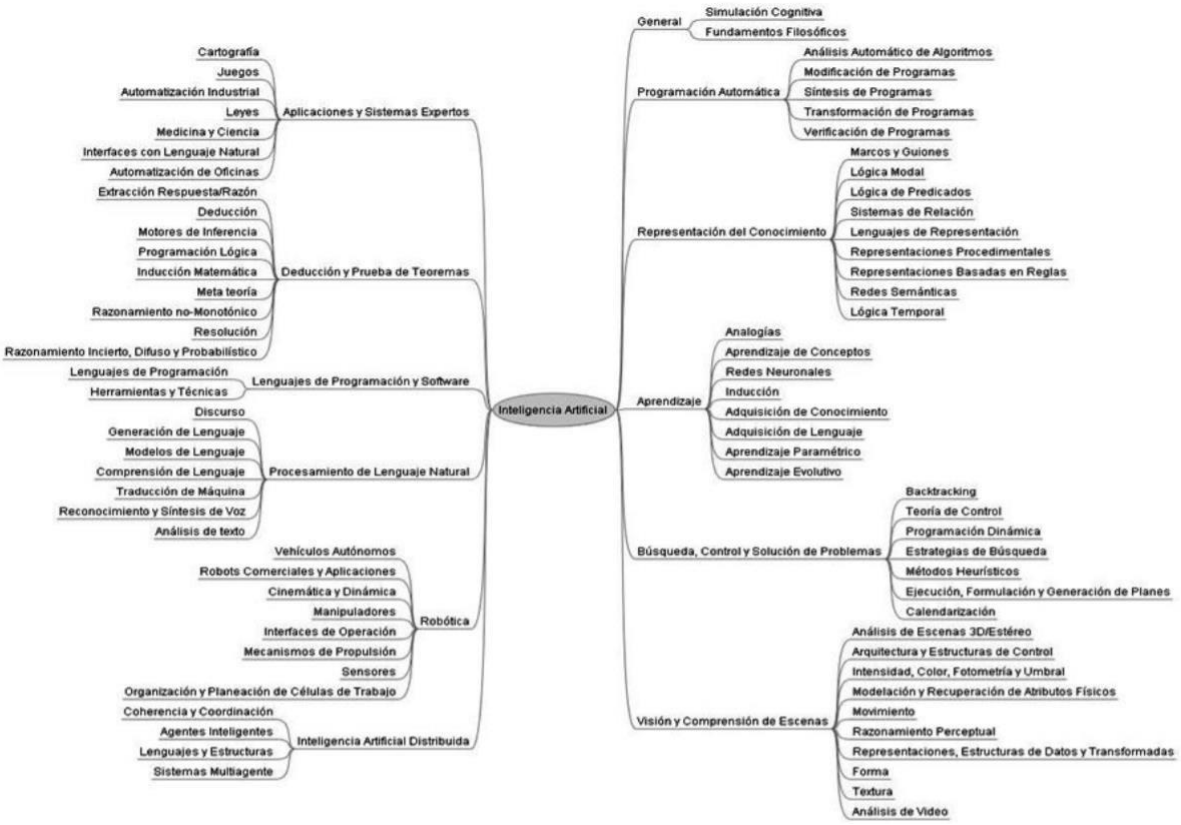


Fig. 1.2 Tópicos de Inteligencia Artificial

Fuente: [6]

Progreso de la Inteligencia Artificial

En la actualidad hay un progreso que ha marcado la IA en áreas como juegos, tratamiento de imágenes, automatización de vehículos, traducción del lenguaje que utilizan un algoritmo dado ya que se trata de problemas discretos a un problema dado.

El progreso de la IA se ha dado en distintos campos que a lo largo del tiempo han generado tecnologías cognitivas como, por ejemplo:

Ingeniería del conocimiento: es trata de recolectar y almacenar el conocimiento humano mediante reglas heurísticas en una estructura de datos para resolver problemas complejos como por ejemplo Seguros que guardan el historial de experiencia de sus clientes para una atención de reclamos. [11]

Robótica: Son dispositivos que actúan de forma mecánica para realizar la función o labor programada para interactuar con el mundo real, pueden llegar a tener similitud a un ser humano en cuanto a su aspecto físico o la forma apropiada para la función que realiza, el uso más común es el automatizar tareas, ensamblaje de piezas y manipulación de material.[11]

Procesamiento de voz: proceso que consiste en transformar el audio de los seres humanos en texto que utilizan aplicaciones para realizar funciones mediante comandos es el caso de Google.[11]

Análisis de imágenes: permite identificar y procesar las imágenes del mundo real, así como objetos o personas por ejemplo el reconocimiento facial. [11]

Aprendizaje Automático: se compone de algoritmos, métodos, técnicas y herramientas para analizar distintos tipos de datos identificado patrones y modelos predictivos de diferentes datos, el caso de uso de este aprendizaje es la identificación de casos fraudulentos mediante patrones ya conocidos. [11]

Redes Neuronales Artificiales

Las redes neuronales artificiales entran en un campo muy amplio e importante de la Inteligencia Artificial, teniendo su principal característica el comportamiento similar al cerebro humano refiriéndose exclusivamente al uso de neuronas y sus conexiones para su comunicación, el principal objetivo es desarrollar modelos artificiales que ayuden a obtener una solución a problemas complejos que utilizan algoritmos matemáticos convencionales. [12]

Bernard Widrow define una RNA como un conjunto de entradas y salidas que son procesadas mediante un proceso de entrenamiento con características similares, donde cada uno de los elementos contiene un peso W o peso sináptico, en donde si los pesos son ajustados puede haber una variación del comportamiento de la red y por ende puede alterar el total de la red en relación con la entrada – salida deseada. En la siguiente figura se presenta una red el esquema de una red neuronal.[12]

Esquema Red Neuronal Artificial

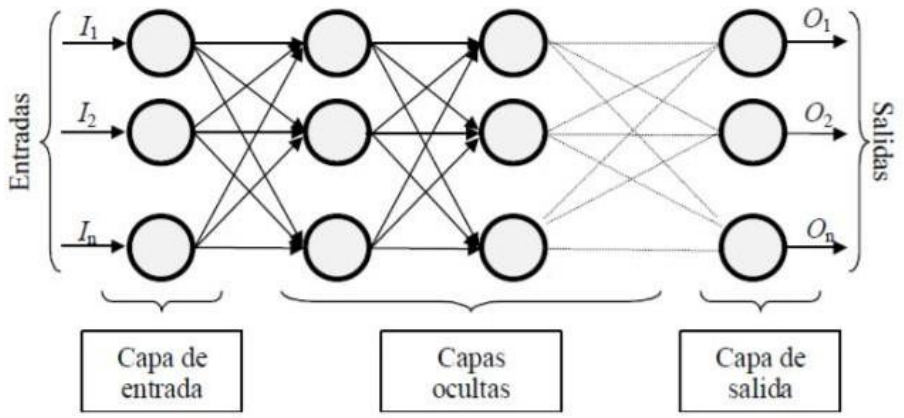


Fig. 1.3 Esquema Red Neuronal Artificial conectada

Fuente: [8]

En la figura 1.3 se puede observar que una red neuronal está constituida por entradas y salidas, además de neuronas interconectas en tres capas, en donde los datos ingresan por una "capa de entrada", se procesan los datos a través de una "capa oculta", que puede estar estructurarse por más de una capa y se obtienen los resultados por una "capa de salida". [12]

Una Red Neuronal Artificial consta de neuronas, con una memoria local, conectadas mediante canales de comunicación, que constan de datos numéricos interpretados de acuerdo con su codificación y uso.

Dentro de la Literatura se puede encontrar un enfoque: Las RNA son sistemas que se adaptan fácilmente que logran relacionarse mediante un grupo de datos que tienen entradas y salidas para luego asociarse en un conjunto de datos con características similares a las entradas.

No requiere de una entrada del exterior (exógena), es decir, pueden relacionarse entre algunas variables sin necesidad de una consideración explícita. Una de las ventajas es que tienen buen entrenamiento así algunos datos contengan ruidos o errores de medición o simplemente cuando el número de entradas es considerablemente limitado.

Una forma en que las Redes Neuronales Artificiales se pueden clasificar es por sus capaz, requerimientos y la dirección que toma el flujo datos. [13]

Una Red Feedforward agrupa los nodos en diferentes capas, empezando por la capa de datos de entrada, pasando después por las capas ocultas(multilayer) con uno o varias neuronas y por último terminando por la capa de datos de salida, pasando por un proceso en donde las neuronas de una capa se conectan con las neuronas de capas adyacentes en donde la información pasa de las neuronas de entrada a las neuronas de la salida. Las neuronas de una capa no pueden conectarse entre sí. Cada una de las conexiones contiene un peso sináptico como representación del vigor de las conexiones entre las neuronas y tener una relación clara entre entradas y salidas(inputs-outputs).[13]

En el siguiente grafico se puede observar una Red Neuronal Feedforward:

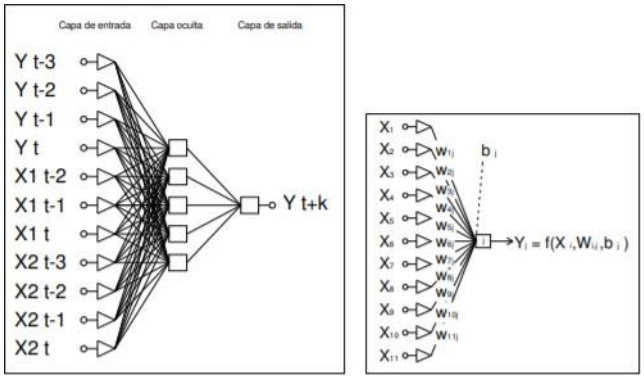


Fig. 1.4 Nodo General de una RNA multilayer perceptrón (MLP)

Fuente: [9]

En la Figura 1.4 se puede observar una Red Feedforward con una capa oculta que consta de 5 neuronas en la capa de entrada, 4 en la capa oculta y una neurona en la capa de salida. En donde W_{ij} se les asigna como en peso de las conexiones entre la neurona $_i$ anterior y la neurona $_j$. Para la salida se obtiene una función de activación f(X, W, b_j) entra la salida dela neurona $Y_{j,}$ donde b_j es el valor del umbral de la neurona en cuestión j.

Se denomina función de activación a la forma funcional en que se determina la respuesta mediante las entradas que esta recibe, para cumplir con esta respuesta pasa por un proceso de aprendizaje en donde se encuentra los umbrales **b** y los pesos W encontrando un mínimo error en la función predeterminada. Para este proceso se utilizan dos tipos de entrenamientos, no supervisado y el supervisado(data-driven) en

un continuo proceso de simulación, en donde se ajusta de forma iterativa los umbrales y los pesos de cada uno de las neuronas. [13]

Sistemas Basados en Lógica Difusa

Los sistemas basados en Lógica Difusa (Fuzzy Logic System, FLS), son los únicos sistemas capaces de trabajar y tratar variables lingüísticas y variables numéricas de forma simultánea en un modo formal. Las variables lingüísticas se caracterizan principal por utilizar un adjetivo para calificarlas, un claro ejemplo, la radiación en el ambiente es alta, por otro lado, las variables de forma numérica se caracterizan por contener un valor numérico que puede ser fijo o variable, por ejemplo, 303.15ºK.

Un Sistema Basado en Lógica Difusa utiliza un mapeo no lineal de un vector de entrada en una salida escalar, para definir un mapeo no lineal se establecen Teoría de Conjuntos Difusos y Lógica Difusa. Un FLS se puede interpretar de forma matemática en funciones base difusas en una combinación lineal como el perceptrón multicapa. Las bases difusas pueden obtenerse mediante datos numéricos o de forma lingüística, para cualquiera de los dos casos adoptan la forma de reglas If-Then. [14]

Para dar solución a un determinado problema se identifica dos tipos de conocimientos, el uno es el conocimiento objetivo, en el que trata modelos matemáticos mediante ecuaciones, un claro ejemplo, los movimientos de un robot, y el conocimiento subjetivo, que consta de información lingüística que contrario a los modelos matemáticos no se pueden cuantificar, un ejemplo de este conocimiento seria una regla en la que sea capaz de detectar el movimiento dentro de un radio de campo o dimensión de objetos que se mueven lentamente. [14]

Al enfrentarnos a un problema el conocimiento subjetivo no es aplicado como tal, pero gran parte de este conocimiento se utiliza para evaluar la solución que este da, es decir se combinan los dos conocimientos para llegar a la solución de problemas reales, a todo este proceso hace referencia la Lógica Difusa o Fuzzy Logic (FL). A continuación se plantea un gráfico de un FSL que procesa señales en FL. [14]

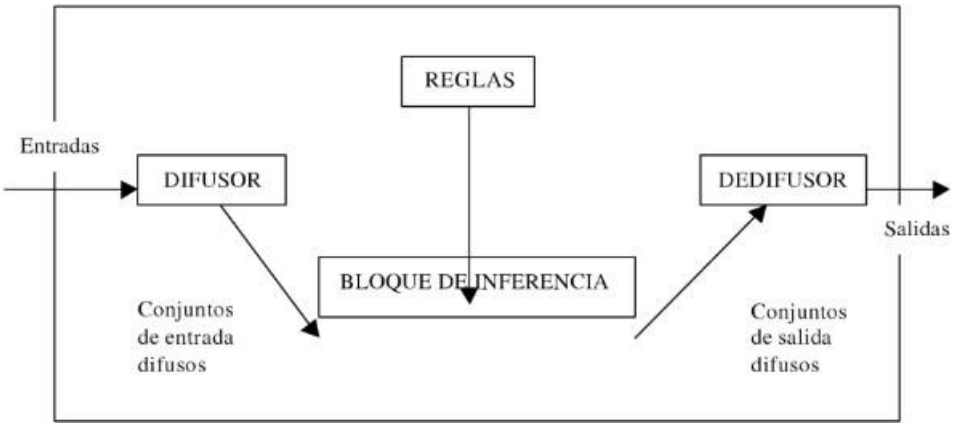


Fig. 1.5 FLS – Sistema Basado en Lógica Difusa

Fuente: [10]

En el grafico 1.5 se puede observar un FLS que contiene Reglas, Difusor, Bloque de Interferencia y Dedifusor, en el que cada componente realiza lo siguiente:

El difusor es el encargado de mapear datos en conjuntos, difusos lo que servirá para pasar a las reglas que se interpretaran lingüísticamente ya que son asociados con ellas. Las reglas pueden ser ingresadas manualmente o se deducen de los datos numéricos ingresados, para luego ser tratadas mediante sentencias IF-THEN, pueden existir reglas que necesiten conocimientos de:

- Relación entre valores numéricos y variables lingüísticas.
- Determinación de variables lingüísticas.
- Correspondencia lógica para variables lingüísticas.
- Implicaciones
- Combinación de reglas.

El bloque de inferencia se encarga de representar conjuntos difusos y combinar reglas. El dedifusor representa los conjuntos de salida en números, ya que muchas veces la salida FLS se encarga de obtener un número.[14]

Redes Neuronales Difusas

Las redes neuronales difusas son sistemas basados en aspectos de las redes neuronales ya que son sistemas capaces de aprender y diseminar, además de incorporar Lógica Difusa implementando razonamientos lógicos basados en reglas de inferencia, con variables lingüísticas de ser necesario.

Uno de los principales enfoques de las redes neuronales difusas es cambiar el desarrollo binario del problema por un tratamiento difuso. Como se sabe el cerebro humano recoge, organiza, procesa e interpreta la información sensorial imprecisa e incompleta del exterior, la Teoría de conjuntos se asemeja a este método sistemático que trabaja con información lingüística, que se la puede interpretar usando etiquetas lingüísticas mediante la computación matemática definidas por las funciones de partición. El componente principal de un Sistema de Inferencia Difuso (FIS – Fuzzy Inference System) es la obtención y selección de reglas If-Then, que permite conceptualizar los datos y el conocimiento experto humano para aplicarlo a una aplicación determinada, a pesar de que un FIS está estructurado en forma de reglas If-Then que no es capaz de adaptarse a las condiciones externas cuando estas se modifiquen a través de la representación del conocimiento, razón por la cual se ha introducido nuevas topologías de aprendizaje de las RNA conocidas como Redes Neuronales Difusas(RNDs).[14]

Características de las Redes Neuronales Difusas

Entre las principales reglas de las redes neuronales difusas tenemos las siguientes:

- Conocimiento humano. – utilizado para solucionar problemas reales mediante reglas de If-Then.
- Modelos biológicos. – utiliza la base de modelos de redes neuronales biológicas como inspiración, en donde las RNA es una de las bases de las RNDs, interactuando con problema de reconocimientos de patrones, percepción o clasificación.
- Computación Numérica. – la computación se basa prácticamente en métodos números, sin embargo, existe una clara perspectiva futura en un enfoque hacia una computación simbólica acercándose a la Inteligencia Artificial.
- Dominio de Aplicación. – desarrollo de señales, reconocimiento de sistemas no lineales, control adaptativo, reconocimiento de patrones.
- Aprendizaje. - proceso de obtener reglas libre de modelos de datos numéricos.
- Computación Intensiva. – las RNDs dependen en su mayoría de una computación intensiva en donde se encuentra reglas o tipo de similitudes en conjuntos de datos, sin antes haber asumido un conocimiento del problema.

- Tolerancia a fallos. – la tolerancia a fallos se hace presente en las RNAs y en los FIS, es decir así se suprima una neurona de la red, actúa de manera en la que el sistema no se destruye. El Sistema continua con su proceso debido a que trabaja con una arquitectura redundante y paralela, a pesar de esto, el sistema se va deteriorando gradualmente.
- Caminos para alcanzar el mínimo. – existen varios caminos para llegar a mínimo error. Sin embargo, el camino que se tome no es relevante mientras que el sistema se vaya acercando hacia el mínimo.
- Resolución de problemas reales. – los problemas reales constituyen en su gran mayoría una magnitud considerable sobre hipótesis que no se conocen con certeza. Esto conlleva, por lo tanto, un camino donde se requiere la utilización de métodos convencionales donde se especifique toda la información necesaria para llegar a la solución.

  Las Redes Neuronales Difusas tienen la capacidad de desarrollar tratamientos con soluciones convincentes a problemas del mundo real, de hecho, la aplicación de estas redes van en aumento, aplicándose a situaciones complejas que no han sido resueltas mediante la aplicación de métodos convencionales. [14]

Evolución de las Redes Neuronales Difusas

El estudio de las RNDs es un campo que se encuentra en apogeo, se han implementado estudios e investigaciones que recientemente van teniendo relevancia. A continuación, se presente un cuadro de evolución de las RNA y las RNDs:

Años	Redes Neuronales	Sistemas Difusos
1940s	1943: Modelo Neuronal McCulloch-Pits	
1950s	1957: Perceptrón	
1960s	1960s: Adalina Madalina	1965: Conjuntos Difusos
1970s	1974: Nacimiento del algoritmo Backpropagation 1975: Cognitrón	1974: Controlador difuso

1980s	1980: Mapeo autoorganizativo 1982: Máquina de Boltzmann 1986: Boom del algoritmo Backpropagation	1985: Modelización difusa (modelo TSK)
1990s		1990: Modelización neuro - difusa 1991: ANFIS 1994: CANFIS Actualidad: generalización difusa de modelos de RNAs

Tabla. 1.1 Evolución de las RNDs

Fuente: [10]

Redes Basadas en Sistemas Difusos

Las principales redes basadas en sistemas difusas son: Espectro Neuro-difuso, MANFIS, y la red en la que se basa la investigación ANFIS.

MANFIS

La característica principal de este tipo de red consiste en una arquitectura que presenta multi-entrada y por consecuente multi-salida, tomando como base la estructura de una red ANFIS y agrupándolas como fueran posibles, es decir se utiliza varias redes ANFIS para obtener una red MANFIS.

Al utilizar una red MANFIS, se le puede dar algunas posibilidades, es decir se implanta en un sistema mono-salida, en donde, se toman algunas redes ANFIS de características distintas, para después hacer un cálculo promedio de todas las salidas que se puedan dar, reduciendo así el error que por lo general se da mediante la varianza. [14]

Espectro Nuero-Difuso

El espectro neuro-difuso permite canalizar el conocimiento antes de realizar una acción mediante una regla de IF-Then, como consecuencia de esto, los modelos resultantes de este proceso tienen la capacidad de ser entendibles en base a los conocimientos previos ya planteados.

Una de las principales características de este espectro es que contienen más utilidades que incluso las mismas RNAs al no presentar tanto inconvenientes de levantamiento y proceso de información. [14]

ANFIS

La red ANFIS (Adaptative Neuro Fuzzy Inference System) fue un modelo que se lo desarrollo en el año 1993 por J.R. Jang, este modelo es muy adaptativo a las necesidades que el problema presente en diferentes áreas, como extracción de datos, procesamiento de imágenes, clasificación de datos, detección de problemas, cálculo de precios, identificación de latidos cardiacos anómalos, la principal característica de este modelo es que se puede utilizar una regla de aprendizaje descomponiendo toda la estructura del conjunto de parámetros.

El modelo ANFIS en su estructura principal utiliza reglas de If-Then, para obtener este tipo de reglas difusas se utilizan dos tipos de métodos: el primero es el método de Sugeno en donde las reglas se producen de forma automática, el segundo método utiliza un algoritmo llamando Mendel, donde se producen reglas de acuerdo con las necesidades del problema, pero al dar un error muy significativo se utiliza el método de Sugeno.

Para la representación de las reglas de Sugeno se utiliza la siguiente simbología:

Regla n: donde fn = pnX + qn Y + rn;

Representado de la siguiente manera: X es representando mediante An y Y es representando mediante Bn, en donde An y Bn se definen como los conjuntos difusos de entrada, y pn, qn y rn se definen como constantes.

Al asignar un valor de cero a n las constantes pasan a ser cero dando como resultado un modelo Sugeno de orden cero, las distintas salidas que el modelo ANFIS pueda dar dependen de la combinación de los parámetros que puedan obtener n, dando así salidas lineales de aplicando la misma regla de manera lineal.[15]

Arquitectura del modelo ANFIS

En la arquitectura ANFIS de acuerdo con el modelo Sugeno se presentan 5 capas que se modifican mediante la manipulación de los parámetros en el aprendizaje, utiliza el algoritmo Backpropagation, es decir utiliza una combinación de las redes neuronales y la lógica difusa para encontrar la respuesta más óptima. A continuación, se representa la arquitectura de Sugeno con 5 capas:

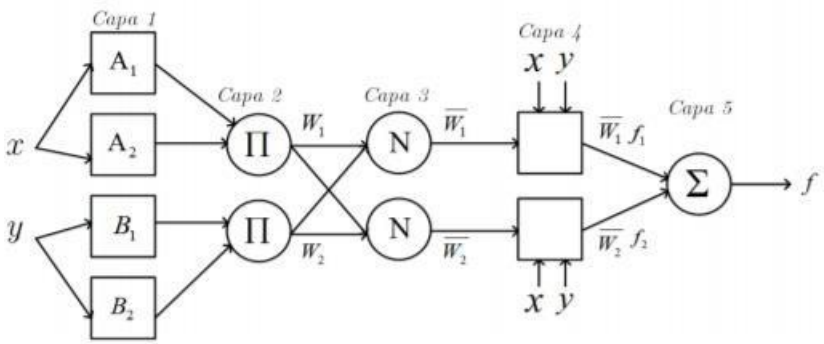


Fig. 1.6 Arquitectura ANFIS – modelo SUGENO

Fuente: [15]

En la figura 1.6 se puede apreciar la arquitectura ANFIS mediante el modelo Sugeno que consta de 5 capas detalladas de la siguiente manera:

Capa 1: Aquí se asigna el grado de pertenencia de cada conjunto difuso que va a ingresar, siendo x o y la entrada al nodo.

Capa 2: En este nodo se calculan todas las señales de salida tomando en cuenta el producto de todas las entradas en este nodo se representa con el símbolo π al ser un nodo fijo.

Capa 3: En esta capa se asigna N al nodo, aquí se calcula la intensidad de procesamiento del total de todas las reglas, en donde obtiene la relación del i-ésimo nodo con la i-ésima regla.

Capa 4: Aquí se realiza la adaptación de una función del nodo, obteniendo los parámetros del consecuente del polinomio Z.

Capa 5: Aquí se obtiene la salida global representada por el signo ∑, obteniendo así una red adaptativa que equivale a un sistema Sugeno de primer orden. [15]

Generación del Dataset

La generación del Dataset es la parte más importante y fundamental del proyecto, debido que se define el método óptimo para el ingreso de datos a la red y posterior entrenamiento, en una red neuronal difusa se utiliza una librería desarrollada en Python que contiene un modelo claro de la serie Mackey Glass para su predicción, aquí se entrena 1500 puntos de la serie donde traza la predicción versus la serie real, la curva de aprendizaje y las funciones de membrecía después del entrenamiento. [16]

Para establecer la capa de entrada se realiza un blob (Input Layer) que consta de los siguientes parámetros:

$$Input\ Layer = \boldsymbol{N} * channel\ \boldsymbol{K} * height\ \boldsymbol{H} * width\ \boldsymbol{W}$$

N: Cantidad de muestras de entrada.

K: *Los posibles numero de canales en la* $\boldsymbol{m}$*uestra.*

H: *La altura del dato de entrada de la* $\boldsymbol{m}$*uestra.*

$\boldsymbol{W}$: *El ancho del dato de entrada de la* $\boldsymbol{m}$*uestra.*

Ecuación 1: Arquitectura de un Blob

Fuente: [16]

De acuerdo con la ecuación 1, se describe que la entrada de un blob, en donde un RGB que cuenta con 3 canales, en donde H y W son valores asignados a cada pixel, dentro de los parámetros a utilizar, N representa el número total de muestras para el entrenamiento de nuestra red, H tendrán el valor de 1 y W se asignó el valor después de realizar el procesamiento de la muestra. [16]

Onda de corriente de un Motor

Para definir una onda de corriente AC se debe tener en cuenta cual es la estructura de esta, esta onda de corriente viene representada por la función coseno, obteniendo la siguiente forma:

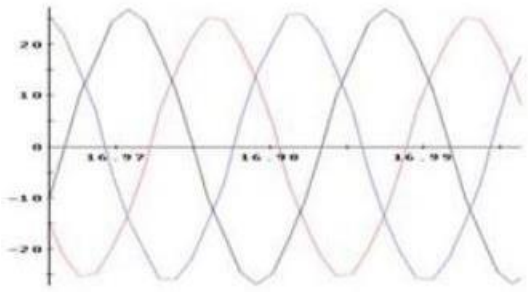


Fig. 1.7 Onda de corriente Motor AC

Fuente: [17]

De acuerdo con las características que presente una onda en función de su tiempo, esta puedo constar de la siguiente ecuación:

$$i_a(t) = I_{max} cos(\omega t)$$
$$Donde:$$

Ecuación 2: Ecuación de onda de corriente AC

Fuente: [17]

En la ecuación 2, una onda de corriente AC puede tomar los valores en función de su tiempo, en donde, $i_a(t)$ pasa a ser la señal de dominio en el tiempo, I_{max} la amplitud que puede llegar a tener la onda de acuerdo con las muestras procesadas, y, por último, ω que viene a ser la frecuencia con que la onda toma los valores de acuerdo con la muestra ingresada.

En un artículo publicado por HaiQui bajo el nombre de “Wavelet Filter-based weak signatura detección method and its application on Rolling element bearing prognostics” en el que habla sobre el ruido y la vibración, menciona que es recomendable añadir ruido blanco con el propósito de obtener la medición de la onda en una parte real y así mejor el entrenamiento de la red, aunque con pequeñas variaciones.[17]

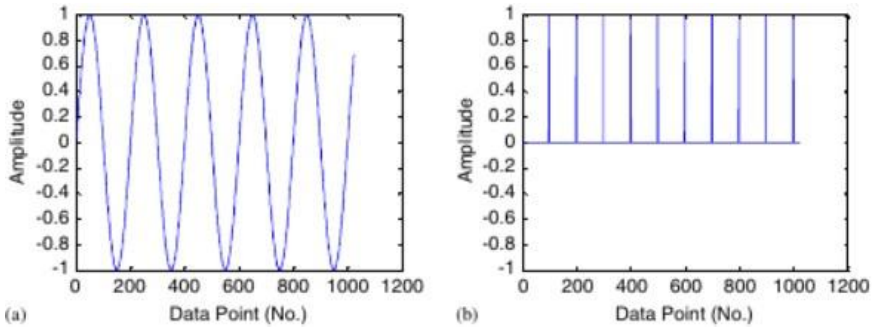


Fig. 1.8 (a) Grafica Onda sinusoidal Pura; (b) Grafica serie de impulsos.
Fuente: [17]

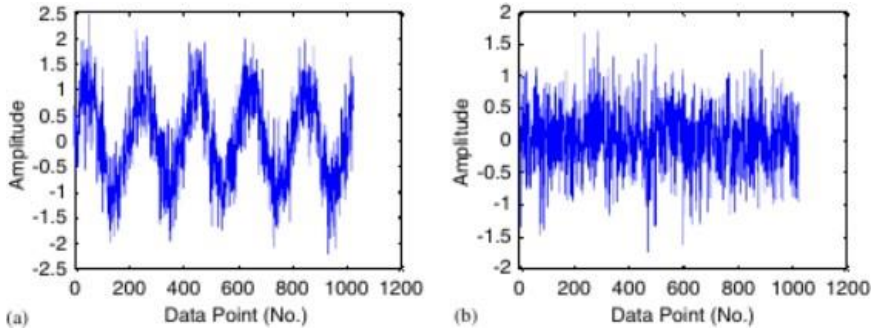


Fig. 1.9 Grafica de onda sinusoidal pura y serie de impulsos con ruido blanco

Fuente: [17]

Las características principales en cuanto al ruido blanco se las puede apreciar en la figura 1.9, en donde la característica principal se diferencia en el acercamiento a una onda real, teniendo en cuanta su amplitud.

En la Fig. 1.8 y 1.9 caso (a) en donde es la representación exacta de la onda de corriente de un motor que tiene las barras rotas, se calcula mediante la siguiente fórmula tomando como base la fórmula de una onda de corriente, pero agregando los siguientes términos;

$$i_a(t) = I_{max}cos(\omega t) + I_{lsb}cos[(1-2s)\omega t] + I_{usb}cos[(1+2s)\omega t]$$

Ecuación 3: Ecuación de onda de corriente AC respecto a un motor con barras rotas.

Fuente: [17]

En la ecuación 3, se calcula los valores de la onda de corriente de un motor que presenta barras rotas, tomando como base la ecuación 2, en donde I_{lsb} representa el máximo valor de la amplitud del lado más bajo de la banda de corriente, es decir, se representa el valor más bajo que esta puede tomar de acuerdo con la frecuencia vs dbs que ingresen como muestra, I_{usb}representa el valor de la amplitud del lado más alto de la banda de corriente.

Como se represente en la fig. 1.9 es necesario agregar ruido blanco a esta onda para obtener así la señal que más se asemeje a la vida real.

Tanto en la Fig. 1.8 y 1.9 caso (b) en donde es la representación exacta de una onda de corriente de motor con asimetría del eje, se añaden parámetros que interpreten la formula exacta de asimetría de eje tomando en cuenta la formula base de una onda.

$$i_a(t) = I_{max}cos(\omega t) + I_{lsb}cos[(\omega - \omega_r)t] + I_{usb}cos[(\omega + \omega_r)\omega t]$$

Ecuación 4: Ecuación de onda de corriente AC respecto a asimetría del eje.

Fuente: [17]

En la ecuación 4, se representa el estado de una onda de corriente AC respecto a asimetría de eje, partiendo de los conceptos ya definidos en las ecuaciones anteriores, añadiendo, ωr que representa la frecuencia de la banda del espectro de la onda, en función del tiempo, además, como es característico en asimetría de eje y barras rotas es necesario aplicar ruido blanco para tener una señal de onda acorde a la realidad.

1.3 Objetivos

Objetivo General

Desarrollo de una red neuronal difusa para la detección de fallos en maquinaria rotativa en tiempo real.

Objetivos Específicos

- Investigar el funcionamiento acerca de una red neuronal difusa que permita analizar los factores que intervienen en el desarrollo de esa red.
- Identificar los distintos tipos de fallos que se pueden presentar en motores trifásicos de corriente alterna.
- Diseñar una red neuronal difusa capaz de aprender mediante la alimentación de datos para la detección de fallos en tiempo real.
- Implementar una red neuronal difusa para la detección de fallos de motores trifásicos de corriente alterna en tiempo real.

CAPITULO II
METODOLOGÍA

2.1 Materiales

Para la elaboración de este proyecto se utilizó materiales como: Matriz de datos, un osciloscopio para la obtención de la matriz de datos en funcionamiento normal, datos con barras rotas y datos con fallo por excentricidad, un computador para el desarrollo de la red, recolección de datos mediante entrevistas, motores trifásicos para la obtención de las muestras, una metodología ágil para el desarrollo del proyecto.

Entre los materiales y elementos a utilizar en el proyecto es la manipulación de datos, cuantitativamente mediante parámetros estadísticos.

Además, se utilizarán libros, artículos científicos, sistemas como fuente documental, debido a que se llevara a cabo una investigación bibliográfica.

2.2 Métodos

Modalidad de Investigación

La investigación será Bibliográfica ya que se recurre a diferentes fuentes obtenidas de libros, documentos, artículos científicos, tesis desarrolladas en Universidades para tomar referencias en la construcción del Marco teórico respecto al tema de investigación.

La investigación tendrá la modalidad de experimental porque se manipularán datos que establecerán la causa y el efecto de los fallos en tiempo real de maquinaria rotativa mediante una red neuronal difusa.

Recolección de Información

La recolección de información se realizó mediante entrevistas, con el Departamento de Investigación de la Facultad de Ingeniería en Sistemas y el Departamento de Electrónica de la Universidad Salesiana, donde se obtuvo la matriz de datos puros iniciales que sirvieron para su posterior clasificación y entrenamiento de la red mediante parámetros estadísticos.

Para el desarrollo del proyecto el departamento de Electrónica de la Universidad Salesiana entrego una matriz de datos en Excel con las muestras estipuladas en su proyecto, que se enlaza con el presente proyecto, partiendo con una base de más de 5000 muestras, Estas muestras servirán como el punto de partida del modelo de datos que serán ingresados para el entrenamiento.

Para la Matriz de datos iniciales, se tomó una muestra adjunta a continuación, debido al tamaño de esta:

Espectro Frecuencia	**Db**	**Db**	**Db**	
	Sin Falla	Falla Eje	Falla Rodamientos	Falla Barras Rotas
60	23,84178	24,38027	24,34872	22,19278
60,5	23,66179	24,15823	24,17252	22,00043
61	23,13859	23,58863	23,65572	21,46923
61,5	22,02039	22,4166	22,54392	20,3356
62	20,58325	20,91859	21,11382	18,87922
62,5	18,74165	19,00659	19,28022	17,01362
63	16,2567	16,44001	16,80328	14,50608
63,5	13,38376	13,46697	13,9389	11,61074
64	9,96136	9,91337	10,5261	8,16594
64,5	5,75113	5,45693	6,31621	3,90026
65	1,00661	0,3267	1,56817	-0,91291
65,5	-4,59659	-5,9309	-4,04303	-6,59931
66	-11,35088	-14,63441	-10,79912	-13,42688
66,5	-18,91369	-23,43047	-18,33092	-20,86604
67	-27,90729	-30,35207	-27,19972	-29,14924
67,5	-35,63596	-32,10985	-34,09827	-33,73013
68	-41,76933	-32,46674	-39,34147	-36,10617
68,5	-43,37253	-35,19474	-40,91427	-35,28217
69	-44,59788	-36,16016	-40,67189	-34,10608
69,5	-45,81121	-35,8607	-39,59478	-32,91274
70	-48,04641	-33,2015	-38,84598	-32,63594
70,5	-46,44096	-31,3348	-38,51324	-32,3631
71	-42,70783	-30,01738	-38,43793	-32,22282
71,5	-41,25823	-29,30058	-38,37713	-33,34762
72	-39,57713	-29,03859	-38,40711	-34,54686
72,5	-37,75313	-29,14385	-38,48686	-35,86143
73	-36,70513	-29,72145	-38,11726	-38,52863
73,5	-35,99126	-30,49787	-38,04136	-40,57956
74	-35,59126	-31,46107	-38,24136	-42,05156
74,5	-35,73918	-32,34034	-38,56488	-44,83477
75	-35,98927	-33,16901	-38,84589	-46,80255
75,5	-36,35087	-33,94021	-39,07629	-47,81215

76	-36,14217	-34,89151	-39,65487	-55,20457
76,5	-36,15952	-35,78152	-40,4945	-60,58917
77	-36,55312	-36,56712	-41,6737	-62,50757
77,5	-36,36535	-38,31552	-43,89086	-56,92067
78	-36,54172	-40,27112	-47,10207	-52,6696
78,5	-37,49692	-42,37512	-51,84127	-52,5
79	-37,86198	-48,63413	-56,30409	-51,05132
79,5	-38,57065	-52,05222	-59,56807	-50,44011
80	-40,31145	-47,25862	-60,26887	-52,27371
80,5	-41,50372	-42,59151	-55,87263	-50,91132
81	-43,28423	-38,84631	-51,59587	-50,48218
81,5	-47,20743	-37,66711	-51,75907	-55,32058
82	-49,47925	-36,70778	-49,78554	-56,39192
82,5	-50,17463	-36,14241	-47,67227	-56,69294
83	-47,11543	-36,79041	-48,06587	-59,38254
83,5	-44,60067	-37,47427	-47,29668	-61,34947
84	-42,30873	-38,44565	-46,32221	-61,6992
84,5	-39,82713	-40,91125	-47,24701	-55,0032
85	-38,32735	-42,74439	-47,2191	-53,51847
85,5	-37,40665	-44,03429	-46,8816	-54,62189
86	-36,45145	-43,84069	-48,8096	-51,24749
86,5	-35,99732	-42,45428	-49,81375	-50,47149
87	-35,92751	-40,23856	-50,26859	-51,47108
87,5	-36,14351	-38,97456	-53,23499	-49,80708
88	-36,68979	-38,00144	-54,29598	-51,15922
88,5	-37,53619	-37,29264	-53,62558	-55,25202
89	-38,53749	-36,69882	-55,9094	-53,74458
89,5	-39,37968	-36,16339	-56,83009	-54,6603
90	-40,04368	-35,69299	-56,22049	-58,2971
90,5	-40,58201	-35,16506	-56,5814	-57,04196
91	-40,80245	-34,86996	-55,52025	-56,65873
91,5	-40,59445	-34,90036	-52,42265	-57,86033
92	-40,5997	-35,18695	-50,12512	-58,69696
92,5	-40,28976	-35,65947	-48,09769	-57,59303
93	-39,39696	-36,39867	-46,37129	-53,23943
93,5	-39,48987	-37,82018	-45,5861	-54,56815
94	-39,52482	-39,5794	-44,89892	-55,74565
94,5	-39,03842	-41,7922	-44,04452	-54,31205
95	-39,45567	-45,44781	-44,29248	-58,70665
95,5	-39,87835	-49,41971	-44,66412	-62,63238
96	-39,65435	-53,21171	-44,57132	-60,99078
96,5	-40,03477	-61,33871	-45,50993	-58,26691
97	-40,43104	-67,22764	-46,49653	-56,00629
97,5	-40,09984	-59,22924	-46,31893	-56,86069
98	-39,87424	-52,72904	-46,61323	-56,06289
98,5	-39,75767	-47,40582	-46,8505	-54,66123

99	-39,95767	-44,65542	-45,6793	-54,02283
99,5	-40,14343	-42,3892	-45,54665	-52,81989
100	-40,45392	-40,54524	-45,87353	-51,41372

Tabla 2.1: Referencia Matriz datos iniciales
Fuente: Departamento Electrónica Universidad Salesiana

Para la recolección de información se debe tener en cuenta los parámetros establecidos por el Departamento de Investigación de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, junto con el Departamento de Electrónica de la Universidad Salesiana, dentro de los cuales se establece: Datos puros de acuerdo a los espectros de frecuencia sin fallas, datos puros de acuerdo al espectro de frecuencia con barras rotas, datos puros de acuerdo al espectro de frecuencia con falla de rodamientos o asimetría de eje. Procesamiento estadístico de acuerdo a los parámetros estadísticos como media, desviación estándar, varianza, curtosis, asimetría estadística, media cuadrática, además de suavización de las ondas eléctricas. Mediante estos requerimientos establecidos la información recolectada se presentará en una tabla de acuerdo a la frecuencia y la amplitud de la onda, en circunstancias normales y en presencia de los fallos establecidos.

Para la parte Bibliográfica se tomará como base artículos en biblioteca y repositorio de la Facultad, además de repositorios virtuales de universidades de distintas partes del País, para el caso de las redes neuronales difusas se recolectará información de libros avalados de artículos académicos de Google Scholar, debido a las características del proyecto no se necesita muestra y población.

Procesamiento y análisis de Datos

El procesamiento y análisis de datos se realizó en base a los parámetros estadísticos establecidos para cada fallo y en condiciones normales, en base a la Matriz de datos Tabla2.1, se analizó datos normales en base a la frecuencia, es decir, en funcionamiento de a 60hz, en condiciones normales se presentó 23,84178 db. A medida que va aumentando los hercios, el valor de los dbs variaron presentando una tolerancia a fallos.

Para el fallo en barras rotas se presentaron 22,19278 db a 60 hz, a medida que los hercios van aumentando se presentó una disminución notable en los dbs.

Para el Fallos en Eje se presentaron variaciones en aumento y disminución de los dbs, dependiendo los hercios en los que la frecuencia se presente.

Una vez presentados los datos puros obtenidos mediante el osciloscopio, se procedió a procesar los datos mediante parámetros estadísticos, el ingreso para el procesamiento de los datos se realizó mediante vectores de 512 datos, a cada vector se agregó una etiqueta, se realizó el procesamiento obteniendo los parámetros estadísticos, guardándolos en otro vector que hereda la etiqueta del vector original de entrada. Se obtuvo un vector de 7 datos, asignándole al séptimo dato la clase a la que pertenece.

Desarrollo del Proyecto

El desarrollo del proyecto se realizará de acuerdo con las siguientes fases:

a. Descripción y utilización de una metodología ágil para el desarrollo del proyecto.

b. Descripción de tecnologías, estructura para el desarrollo, funcionamiento y entrenamiento de la red neuronal difusa.

c. Análisis, descripción de la arquitectura de la red y los tipos de datos a manejar.

d. Identificación y desarrollo de métodos para el procesamiento de los datos de muestra para el correcto funcionamiento de la red y la simulación de los fallos descritos.

e. Desarrollo de los métodos para la generación del dataset para los datos de ingreso.

f. Desarrollo y especificación de líneas de código utilizado para la creación y entrenamiento de la red.

g. Análisis y descripción del entrenamiento de la red con las muestras necesarias.

h. Análisis y evaluación del funcionamiento de la red con las muestras ingresadas.

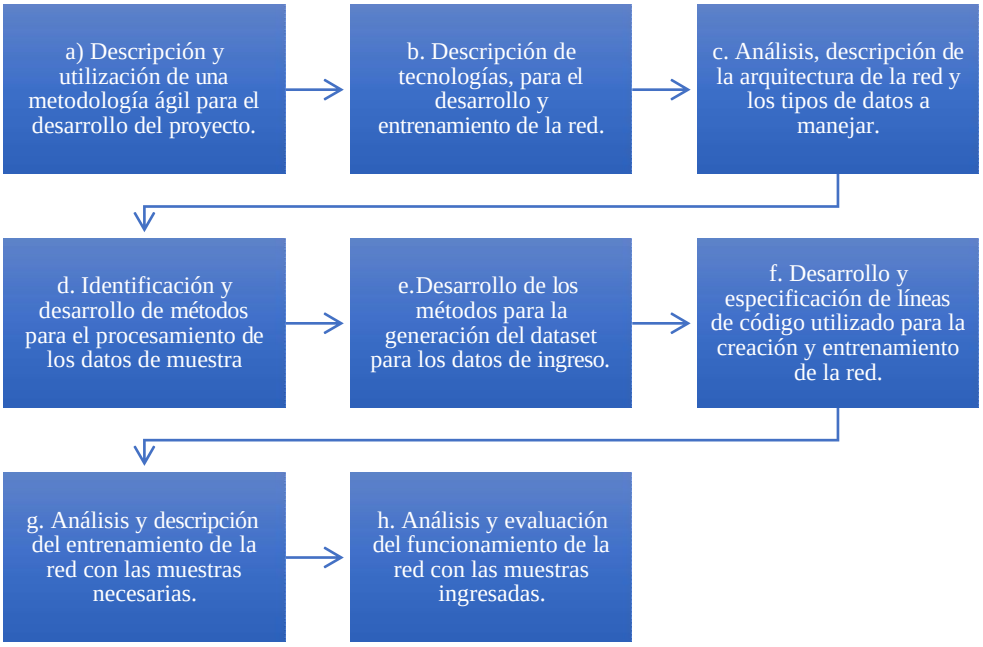


Fig. 2.1 Proceso desarrollo de la red
Fuente: Autor

CAPITULO III

RESULTADOS Y DISCUSIÓN

3.1 Análisis y discusión de resultados

Para el análisis y discusión de resultados se toma en cuenta que las muestras a analizar deben ser reales y en gran cantidad para que se pueda producir un procesamiento en donde influyan todos los parámetros a utilizar, la prueba se los realizo con muestras de motores reales de la Matriz de datos recolectada, logrando así entrenar a la red y demostrar la efectividad que cada una de estas tiene en la detección de fallos y en su funcionamiento normal.

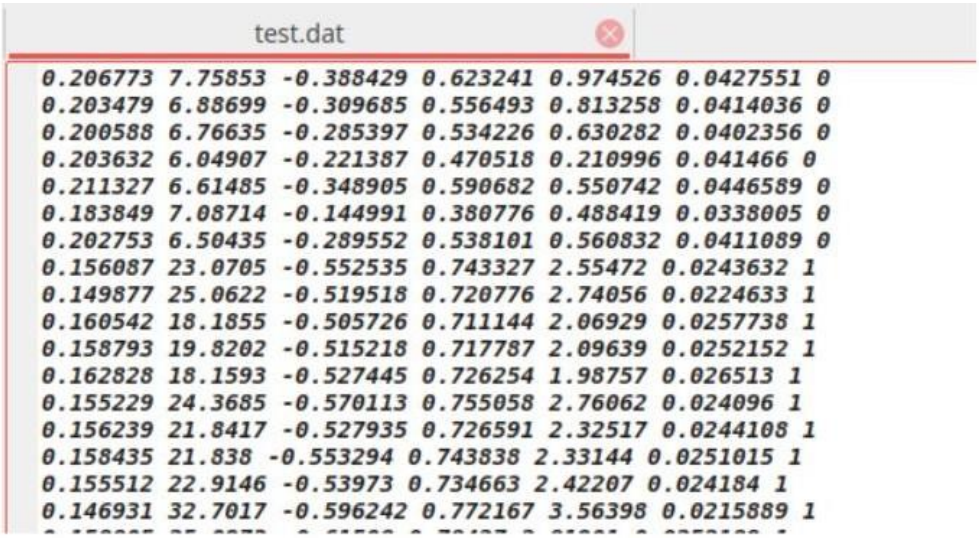


Fig3.1 Vectores de entrenamiento de la red
Fuente: Autor

Una vez que se realizó el procesamiento de los datos, guardamos en dos archivos, en el uno se guardó los datos de entrenamiento y en el otro los datos de comprobación, como se observa en la Fig3.1 podemos cada valor está separado por un espacio y donde las seis primeras columnas representan los parámetros estadísticos en orden: desviación estándar, curtosis, media, media cuadrática, , asimetría estadística, varianza, y el séptimo es la clase a la que pertenece, en funcionamiento normal o los dos tipos de errores.

3.1.1 Análisis de Redes Neuronales Difusas

Para el desarrollo de una red neuronal difusa, se evidencio que se basa en la arquitectura ANFIS, que utiliza técnicas basadas en aprendizaje neuro-adaptativas. Se aplicó la captura de datos dado un conjunto de entrada/salida, construyendo un sistema de inferencia que ajuste los parámetros de la función de membresía utilizando el algoritmo de backpropagation, permitiendo que el sistema difuso aprenda de los datos que está modelando, es decir utiliza una combinación de las redes neuronales y la lógica difusa para encontrar la respuesta más óptima.

El algoritmo Backpropagation utiliza el método de cálculo de gradiente para aprendizaje supervisado y proceder al entrenamiento de la red, este método aplica un clico de propagación y adaptación aplicando un vector de entrada a la red que se propagara por todas las capas de la red hasta generar la salida. La salida se compara con los datos esperados, generando una señal de error para cada salida.

Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida. Sin embargo, la neurona de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total.

3.1.1 Análisis de los tipos de fallos

Las ondas eléctricas adquiridas de los motores con carga nominal se encuentran en buen estado de funcionamiento y serán comparadas con las señales de corrientes obtenidas en cada falla. Para posteriormente interpretar sus características espectrales de fallas incipientes como es el caso en asimetría de eje, barras rotas en una fase.

Falla barras rotas

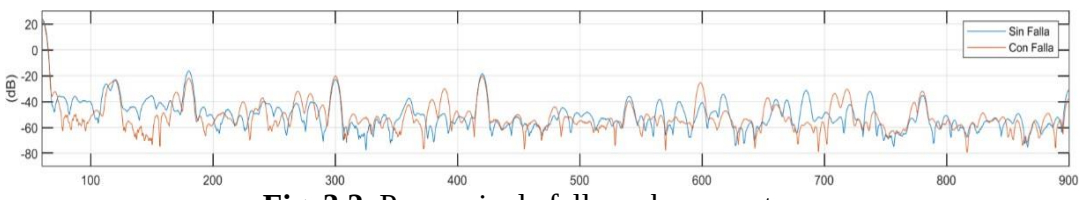


Fig. 3.2. Presencia de falla en barras rotas

En el motor se desarrolló el análisis de las corrientes trifásicas distorsionadas cuando el motor presenta barras rotas operado a carga nominal. Se observa que durante el rango de frecuencias de 75 a 175 [Hz] la amplitud disminuye drásticamente bajo los -40 dBv respecto las señales sin fallas exceptuando la amplitud del espectro en la frecuencia de 120 [Hz], además de existir un aumento en la amplitud espectral en las frecuencias de 360, 600 y 720 [Hz], características que se repiten en las fases. Cabe mencionar que con este tipo de falla el motor subió su velocidad respecto la nominal en 5%, disminuyendo el deslizamiento.

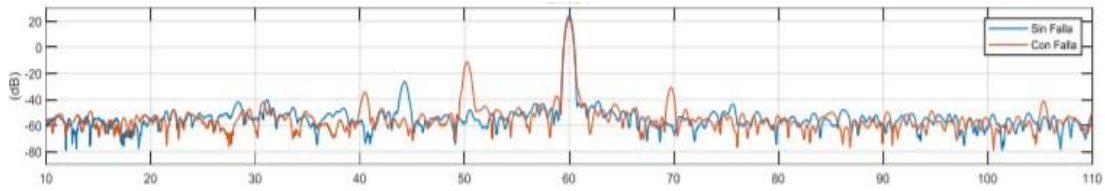


Fig. 3.3. Presencia de barras rotas, diagnóstico por picos lateral inferior y superior

Para un mejor análisis en falla por barras rotas, se fragmentó la señal a 100 Hz y se la centró en la frecuencia fundamental, cabe considerar que para definir como tal la falla por barras rotas se menciona que existiría falla de este tipo cuando los espectros sean mayores a -20 dBv del nivel de ruido.

En figura anterior se observa la existencia de un pico lateral superior a 70 [Hz] y dos picos laterales inferiores junto a la fundamental en 40 y 50 [Hz], teniendo este último una amplitud de aproximadamente -10 dBv en las tres fases que componen el sistema, lo cual quiere decir que el motor presenta una falla en el interior del motor por barras rotas en la jaula de ardilla.

Excentricidad

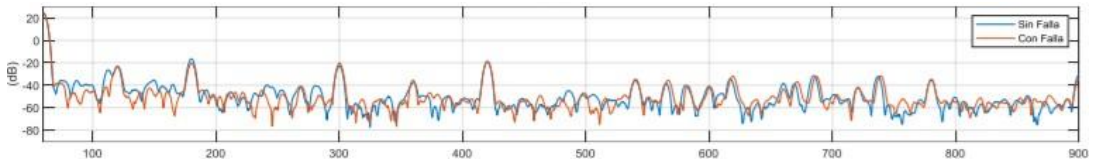


Fig. 3.4. Presencia de falla por excentricidad

En la Figura anterior se observa la variación de amplitud espectral en las tres fases del sistema en aproximadamente un rango de frecuencias desde los 68 a 110 [Hz], en los cuales se presenta una disminución de amplitud espectral menor a -40 dBV, además un pequeño incremento de amplitud en la frecuencia de 300 Hz respecto las señales sin fallas e indica la presencia de falla por excentricidad.

De la misma manera, se indica la existencia de variaciones de amplitud espectral bajo los -40 dBV aproximadamente en la frecuencia de 80 [Hz], siendo una característica propia en las fases, además se observa que; en la frecuencia de 300 Hz hay un ligero crecimiento en la amplitud espectral respecto las señales sin fallas.

3.2 Desarrollo de la propuesta

3.2.1 Descripción de la Metodología a utilizar

Metodología XP (Extreme Programming)

La metodología XP, es la metodología ágil más conocida debido a su impacto y resultados, ya que se encarga de guiar a equipos de trabajo pequeños o normalmente medianos, de entre dos a diez personas encargadas de la programación, que suelen trabajar en ambientes en que no están establecidos todos los parámetros o imprecisos.

Una de las principales características de la metodología XP, es la creación de historias de Usuario en donde el cliente describe las características que debe poseer el sistema, además de la funcionalidad que este debe tener. [18]

Para esta metodología se aplica el proceso llamado Planning game, en donde se define la fecha de entrega de las historias de usuario definidas por el cliente, posteriormente los desarrolladores deciden el alcance del proyecto para la entrega, costos de desarrollo e implementación, además de un cierto número de iteraciones que el cliente define de acuerdo con las historias de usuario para una entrega funcional.

Una de las características fundamentales de la metodología XP, es el trabajo en parejas en donde, cada funcionalidad es asignada a una pareja, el equipo de trabajo tiene que ir rotando constantemente para que todo el equipo de desarrolladores esté al tanto de toda la funcionalidad que el proyecto tiene. [18]

Roles Metodología XP

De acuerdo con Beck los desarrolladores o el equipo de trabajo deben tener asignados roles detallados a continuación:

Programador

Persona encargada de escribir las líneas de código y pruebas unitarias del sistema, pieza básica en el desarrollo de la metodología y del Sistema. [18]

Cliente

Perona encargada de escribir las historias de usuario, se encarga de detallar las especificaciones del sistema, tiene plena certeza en el cumpliendo de los requerimientos y en el grupo de desarrollo, define pruebas de funcionamiento. [18]

Encargado de pruebas (Tester)

Es la persona encargada de realizar las pruebas funcionales junto con el cliente, además de realizar pruebas frecuentemente y mantiene al equipo al tanto de los resultados. [18]

Encargado de seguimiento (Tracker)

Es la persona encargada de vigilar el proceso de desarrollo del proyecto, además de recoger y organizar información, comparar y estimar los tiempos de desarrollo con los tiempos dedicados al mismo, por último, informa el resultado del análisis desarrollado a todo el grupo sin afectar el proceso.[18]

Entrenador (coach)

Es la persona que está al tanto de todo el proceso, experto en XP, además de proveer las herramientas y métodos necesarios al equipo para llevar un proceso detalla y correcto indirectamente. [18]

Gestor (Big boss)

Persona que se encarga de todos los problemas que rodean al equipo, es decir, es el dueño del equipo, las funciones que este realiza depende del éxito del proyecto y que cada uno de los integrantes del equipo se encarguen de las labores asignadas sin necesidad de verificación o presión. [18]

Características XP

Las características con las que cuentan la metodología XP son las siguientes**:**

- Mejoras continuas en base al desarrollo de líneas de código.
- Realización de pruebas unitarias frecuentes incluyendo regresión.
- Interacción del cliente con el equipo de programación
- Corrección de secuencias y errores del sistema
- Retroalimentación del código y simpleza en el mismo.

Funcionamiento de la metodología XP

El primer paso para el correcto funcionamiento de la metodología XP es la planeación en donde el equipo de trabajo levanta requerimientos que el cliente los detalla mediante las historias de usuario, asignando prioridades, tiempo de desarrollo, costo y grupos de trabajo. En el siguiente paso se basa en el diseño, en base a las historias de usuario y siguiendo la norma MS que significa mantelo sencillo.

Codificación es el tercer paso de la metodología XP en donde se crean las líneas de código en base al diseño que se estableció mediante las historias de usuario, incluyendo pruebas unitarias una vez realizada una iteración ayudando a tener una retroalimentación de cada una de las historias definidas en el diseño.

Por ultimo las pruebas, empezando por las pruebas unitarias que se realizan cada vez que se termina una iteración son fundamentales ya que se las automatiza, una vez que el código se modifique o aumente las pruebas unitarias pasan a ser pruebas de regresión, cuando se ejecutan las pruebas unitarias estas se almacenan para realizar las llamadas pruebas de validación e integración ejecutadas por lo general todos los días, con la finalidad de corregir posibles errores que se vayan presentando, para después pasar a las pruebas de aceptación en donde el cliente se siente conformo o no con los requisitos, características y funcionalidades dados al iniciar el proyecto. [18]

	CMM	ASD	Crystal	DSM	FDD	LD	SCRUM	XP
Cambios requisitos del sistema	1	5	4	3	3	4	5	**5**
Colaboración	2	5	5	4	4	4	5	**5**
Resultados	2	5	5	4	4	4	5	**5**
Simplicidad	1	4	4	3	5	3	5	**5**
Adaptabilidad	2	5	5	3	3	4	4	**3**
Excelencia técnica	4	3	3	4	4	4	3	**4**
Prácticas de Colaboración	2	5	5	4	3	3	4	**5**
Media CM	2.2	4.4	4.4	3.6	3.8	3.6	4.2	**4.4**
Media Total	1.7	4.8	4.5	3.6	3.6	3.9	4.7	**4.8**

Tabla:3.1 Tabla comparación Metodologías

Fuente: H. J, Agile Software Development Ecosystems, Addison-Wesley

De acuerdo con las características de estas dos metodologías la más factible de usar en este proyecto será la metodología XP debido a que se realiza un trabajo rápido, iterativo, se trabaja en grupo, pero todos tiene la capacidad de estar al tanto del proyecto, además de que su código es limpio y sencillo.

Para la realización del proyecto se dividieron en las siguientes fases:

- Planificación
- Diseño
- Codificación
- Pruebas

Planificación del proyecto

Para la planificación del proyecto y al haber escogido la metodología XP, se trabajará mediante las historias de usuario definidas, tomando en cuenta los requerimientos. Los requerimientos fueron establecidos por el departamento de Investigación de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

Rol	**Descripción**
Programador	Autor del presente trabajo de Investigación
Usuario Final	Departamento de Investigación
Tester	Autor del presente trabajo de Investigación
Tracker	Autor del presente trabajo de Investigación
Entrenador	Autor del presente trabajo de Investigación
Consultor	Docente del departamento de investigación
Gestor	Autor del presente trabajo de Investigación

Tabla 3.2: Roles
Fuente: Elaborado por el autor

Historias de usuario establecidas por los requerimientos iniciales.

Historia de Usuario	
Código: H1	**Usuario:** Usuario Final
Iteración Asignada: 1	**Prioridad de Negocio:** Alta
Riesgo de Desarrollo: Alto	
Nombre de Historia: Dataset Entrenamiento	
Descripción: En el entrenamiento de una red neuronal difusa se requiere cierta cantidad de datos para que la red pueda trabajar de una forma funcional, es decir, pueda predecir los diferentes errores ya definidos y a su vez un funcionamiento normal en su motor.	
Tareas: • Diseño del Dataset • Creación del Dataset	

Tabla 3.3: Historia Dataset entrenamiento
Fuente: Autor

Historia de Usuario	
Código: H2	**Usuario:** Usuario Final
Iteración Asignada: 1	**Prioridad de Negocio:** Alta
Riesgo de Desarrollo: Alto	
Nombre de Historia: Generación de muestras	
Descripción: Para el entrenamiento de una red se necesita una cantidad de muestras considerables, para obtener este tipo de muestras es necesario crearlas mediante un sistema que me permita generar muestras para un correcto funcionamiento del motor, así como para los fallos propuestos a analizar, logrando así obtener muestras que se acerque a la realidad.	
Tareas: • Escoger algoritmos que realicen las simulaciones • Seleccionar algoritmos para añadir ruido Blanco • Establecer métodos para generar muestras • Establecer métodos para ingresar los datos mediante el Dataset.	

Tabla 3.4: Historia de usuario Generación de Muestras
Fuente: Autor

Historia de Usuario	
Código: H3	**Usuario:** Usuario Final
Iteración Asignada: 2	**Prioridad de Negocio:** Alta
Riesgo de Desarrollo: Alto	
Nombre de Historia: Arquitectura a utilizar	
Descripción: Se presenta la necesidad de escoger una arquitectura que mejor se acomode a los requerimientos y el tipo de trabajo propuesto para clasificar correctamente los fallos, una vez escogida la arquitectura se comprobara si es la óptima para el proyecto.	
Tareas: • Establecimiento de una arquitectura para clasificar correctamente los fallos mediante señales eléctricas. • Comprobación de la arquitectura seleccionada.	

Tabla 3.5: Historia de usuario Arquitectura a utilizar
Fuente: Autor

Historia de Usuario	
Código: H4	**Usuario:** Usuario Final
Iteración Asignada: 2	**Prioridad de Negocio:** Alta
Riesgo de Desarrollo: Alto	
Nombre de Historia: Desarrollo de la red neuronal difusa	
Descripción: Se desarrolla una red neuronal difusa en base al diseño y la arquitectura establecida que vaya en concordancia a una óptima clasificación y sobre todo se reduzca el tiempo de procesamiento, brindando así una clasificación optima y un funcionamiento ágil.	
Tareas: • Lenguaje de programación a utilizar • Establecimiento de datos • Utilización de librerías necesarias • Desarrollo de la red basada en el diseño y parámetros de la arquitectura. • Establecer los métodos necesarios para una óptima funcionalidad.	

Tabla 3.6: Historia de usuario Desarrollo de la red neuronal difusa
Fuente: Autor

Historia de Usuario	
Código: H5	**Usuario:** Usuario Final
Iteración Asignada: 3	**Prioridad de Negocio:** Media
Riesgo de Desarrollo: Bajo	
Nombre de Historia: Desarrollo de la Interfaz	
Descripción: Se desarrollará una interfaz simple, ergonómica y fácil de usar, en donde se contará con botones que se puedan elegir las muestras, además de un espacio donde se pueda graficar las muestras ingresadas para su respectivo análisis y obtención de resultados.	
Tareas: • Desarrollo de la Interfaz • Inclusión de botón para cargar las muestras • Desarrollo de métodos para la visualización de muestras • Obtención de resultado.	

Tabla 3.7: Historia de usuario Desarrollo de la Interfaz
Fuente: Autor

Actividades

- **Historia:** Dataset Entrenamiento

Tarea	
Código: T1	**Código de Historia:** H1
Nombre de Tarea: Diseño del Dataset	
Tipo de tarea: Estructuración Dataset	**Puntos de Estimación:** 2
Duración: 1 Semana	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Desarrollo de la estructura del Dataset que se implementara para el ingreso de las muestras.	

Tabla 3.8: Diseño del Dataset – Historia 1
Fuente: Autor

- **Historia:** Dataset Entrenamiento

Tarea	
Código: T2	**Código de Historia:** H1
Nombre de Tarea: Creación Dataset	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 1 Semana	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Se procede a la creación del Dataset	

Tabla 3.9: Creación del Dataset – Historia 1
Fuente: Autor

- **Historia:** Generación de muestras

Tarea	
Código: T3	**Código de Historia:** H2
Nombre de Tarea: Escoger algoritmos que realicen las simulaciones	
Tipo de tarea: Investigación	**Puntos de Estimación:** 1
Duración: 2 Días	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Se escogerá el algoritmo adecuado para la generación de muestras en base a los fallos propuestos.	

Tabla 3.10: Escoger Algoritmos que realicen las simulaciones – Historia 2
Fuente: Autor

- **Historia:** Generación de Muestras

Tarea	
Código: T4	**Código de Historia:** H2
Nombre de Tarea: Seleccionar algoritmos para añadir ruido Blanco	
Tipo de tarea: Investigación	**Puntos de Estimación:** 1
Duración: 2 Días	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Se escogerá el algoritmo óptimo para añadir ruido blanco.	

Tabla 3.11: Seleccionar algoritmos para añadir ruido Blanco – Historia 2
Fuente: Autor

- **Historia:** Generación de Muestras

Tarea	
Código: T5	**Código de Historia:** H2
Nombre de Tarea: Establecer métodos para generar muestras	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 6 Días	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Desarrollo de métodos útiles para la generación de muestras.	

Tabla 3.12: Establecer métodos para generar muestras – Historia 2
Fuente: Autor

- **Historia:** Generación de Muestras

Tarea	
Código: T6	**Código de Historia:** H2
Nombre de Tarea: Establecer métodos para ingresar los datos mediante el Dataset.	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 3 Días	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Establecer métodos útiles para ingresar los datos mediante el Dataset.	

Tabla 3.13: Establecer métodos para ingresar los datos mediante el Dataset- Historia 2
Fuente: Autor

- **Historia:** Arquitectura a utilizar

Tarea	
Código: T7	**Código de Historia:** H3
Nombre de Tarea: Establecimiento de una arquitectura para clasificar correctamente los fallos mediante señales eléctricas.	
Tipo de tarea: Investigación	**Puntos de Estimación:** 3
Duración: 1 Semana	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Selección de una arquitectura que pueda clasificar de forma óptima y correcta los fallos mediante señales eléctricas.	

Tabla 3.14: Establecimiento de una arquitectura – Historia 3
Fuente: Autor

- **Historia:** Arquitectura a utilizar

Tarea	
Código: T8	**Código de Historia:** H3
Nombre de Tarea: Comprobación de la arquitectura seleccionada.	
Tipo de tarea: Investigación	**Puntos de Estimación: 2**
Duración: 1 Semana	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Comprobar la arquitectura seleccionada en base a los estudios ya existentes.	

Tabla 3.15: Comprobación de la arquitectura seleccionada – Historia 3
Fuente: Autor

- **Historia:** Desarrollo de la red neuronal difusa.

Tarea	
Código: T9	**Código de Historia:** H4
Nombre de Tarea: Lenguaje de programación a utilizar	
Tipo de tarea: Investigación	**Puntos de Estimación: 2**
Duración: 2 Días	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Seleccionar el lenguaje de programación a utilizar de acuerdo con los requerimientos y el diseño de la arquitectura.	

Tabla 3.16: Lenguaje de Programación a utilizar – Historia 4
Fuente: Autor

- **Historia:** Desarrollo de la red neuronal difusa.

Tarea	
Código: T10	**Código de Historia:** H4
Nombre de Tarea: Establecimiento de Datos.	
Tipo de tarea: Investigación	**Puntos de Estimación: 2**
Duración: 1 Semana	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Definición de tipo de datos a utilizar dependiendo de los requerimientos establecidos.	

Tabla 3.17: Establecimiento de Datos – Historia 4
Fuente: Autor

- **Historia:** Desarrollo de la red neuronal difusa.

Tarea	
Código: T11	**Código de Historia:** H4
Nombre de Tarea: Utilización de librerías Necesarias	
Tipo de tarea: Investigación	**Puntos de Estimación: 2**
Duración: 1 Semana	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Definición de las librerías necesarias para agregar los métodos y su debida funcionalidad.	

Tabla 3.18: Utilización de librerías Necesarias – Historia 4
Fuente: Autor

- **Historia:** Desarrollo de la red neuronal difusa.

Tarea	
Código: T12	**Código de Historia:** H4
Nombre de Tarea: Desarrollo de la red basada en el diseño y parámetros de la arquitectura.	
Tipo de tarea: Desarrollo	**Puntos de Estimación: 2**
Duración: 2 Meses	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Desarrollo de la red neuronal difusa basada en los requerimientos, parámetros y diseño de la arquitectura.	

Tabla 3.19: Desarrollo de la red – Historia 4
Fuente: Autor

- **Historia:** Desarrollo de la red neuronal difusa.

Tarea	
Código: T13	**Código de Historia:** H4
Nombre de Tarea: Establecer los métodos necesarios para una óptima funcionalidad.	
Tipo de tarea: Desarrollo	**Puntos de Estimación: 2**
Duración: 2 Semanas	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Establecer los métodos necesarios para el correcto funcionamiento de la red, con un proceso ágil.	

Tabla 3.20: Establecer métodos necesarios para una óptima funcionalidad - Historia 4
Fuente: Autor

- **Historia:** Desarrollo de la Interfaz

Tarea	
Código: T14	**Código de Historia:** H5
Nombre de Tarea: Desarrollo de la Interfaz	
Tipo de tarea: Desarrollo	**Puntos de Estimación: 2**
Duración: 3 Semanas	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Desarrollo de una interfaz amigable para el usuario y ergonómica.	

Tabla 3.21: Desarrollo de la Interfaz - Historia 5
Fuente: Autor

- **Historia:** Desarrollo de la Interfaz

Tarea	
Código: T15	**Código de Historia:** H5
Nombre de Tarea: Inclusión de botón para cargar las muestras	
Tipo de tarea: Desarrollo	**Puntos de Estimación: 2**
Duración: 2 Semanas	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Inclusión de botones para selección de muestras con sus debidos métodos de lectura.	

Tabla 3.22: Inclusión de botón para cargar las muestras - Historia 5
Fuente: Autor

- **Historia:** Desarrollo de la Interfaz

Tarea	
Código: T16	**Código de Historia:** H5
Nombre de Tarea: Desarrollo de métodos para la visualización de muestras	
Tipo de tarea: Desarrollo	**Puntos de Estimación: 2**
Duración: 3 Semanas	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Desarrollo de métodos necesarios para la presentación de muestras mediante una gráfica.	

Tabla 3.23: Desarrollo de métodos para la visualización de muestras - Historia 5
Fuente: Autor

- **Historia:** Desarrollo de la Interfaz

Tarea	
Código: T17	**Código de Historia:** H5
Nombre de Tarea: Obtención de resultado	
Tipo de tarea: Desarrollo	**Puntos de Estimación: 2**
Duración: 3 Semanas	
Programador: Alexis Roberto Jarrín Núñez	
Descripción: Implementación de todos los métodos desarrollados para obtener el resultado.	

Tabla 3.24: Obtención de resultado - Historia 5
Fuente: Autor

Diseño

Diagrama de Secuencia

Para el correcto proceso de desarrollo y procesamiento de los datos se utiliza un diagrama de secuencia en donde se simboliza las iteraciones que se realizan de los datos para llegar a una óptima clasificación y muestra de resultados al usuario final.

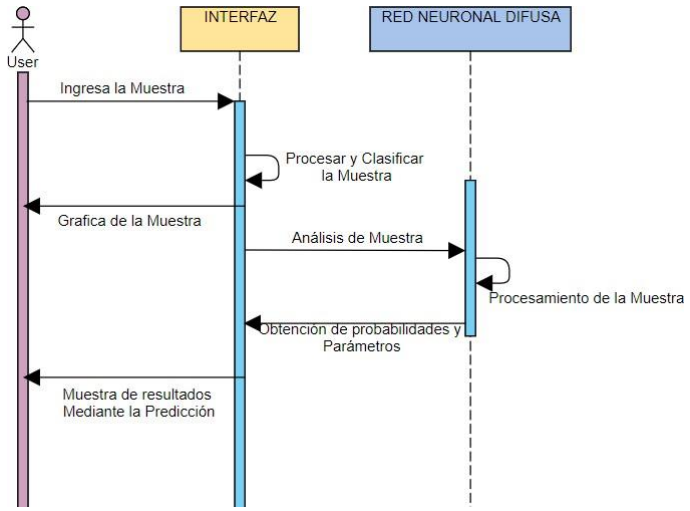


Fig3.5. Diagrama de secuencia del sistema
Fuente: Autor

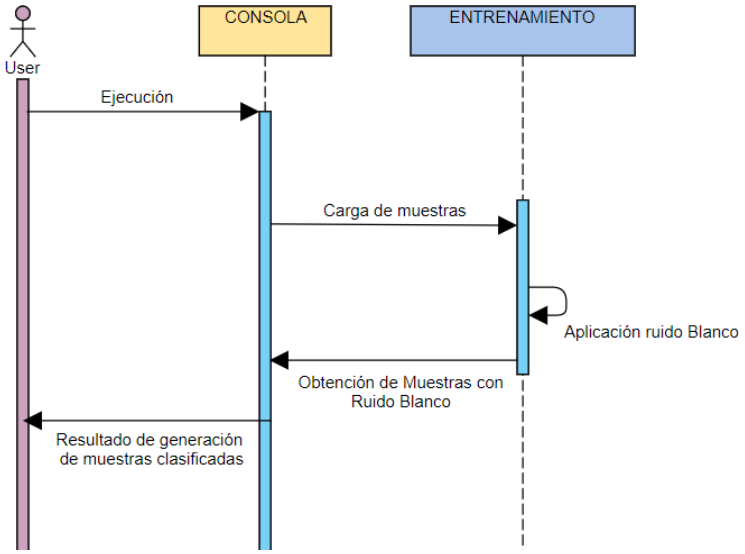


Fig3.6. Diagrama de secuencia obtención de muestras

Fuente: Autor

Tarjetas CRC (Clase – Responsabilidad – Colaboración)

Dataset Entrenamiento	
Responsabilidad	**Colaboradores**
Desarrollo de Dataset de entrenamiento con ingreso de Datos.	Tensorflow Numpy Matplotlib

Tabla 3.25: Tarjeta CRC Dataset Entrenamiento
Fuente: Autor

Entrenamiento de muestras	
Responsabilidad	**Colaboradores**
Ingreso de muestras al modelo propuesto para su clasificación.	Librería Boost Librería HDF5 Librería fstream Librería H5

Tabla 3.26: Tarjeta CRC Generación de muestras
Fuente: Autor

Arquitectura a utilizar	
Responsabilidad	**Colaboradores**
Utilización de la arquitectura óptima para el desarrollo de la red.	ANFIS SUGENO

Tabla 3.27: Tarjeta CRC Arquitectura a utilizar
Fuente: Autor

Desarrollo de la red neuronal difusa	
Responsabilidad	**Colaboradores**
Desarrollo de la Red Neuronal Difusa, de acuerdo a los requerimientos y parámetros establecidos.	Librería desarrollada en Python - TensorFlow Librería ANFIS Librería Argparse

Tabla 3.28: Tarjeta CRC Desarrollo de la red neuronal difusa
Fuente: Autor

Desarrollo de la Interfaz	
Responsabilidad	**Colaboradores**
Desarrollo de una interfaz ergonómica y sencilla, en la que se pueda seleccionar, y visualizar las ondas que se generan de las muestras.	QT creator Librerías estándar de C++ y Python.

Tabla 3.29: Tarjeta CRC Desarrollo de la Interfaz
Fuente: Autor

Codificación

La codificación de este proyecto se basa en librerías de c++ y librerías de Python debido a que para ANFIS en el lenguaje de programación c++ no hay librerías específicas para este tipo de red, dentro de las librerías a utilizar son las siguientes: tensorflow == 1. 7.0

numpy == 1. 14 .2

matplotlib == 2.2.2

tensorflow == 1. 15.2

numpy == 1. 15 .2

matplotlib == 3.0.0

Estructura de código

anfis.py: contiene la implementación de ANFIS.

mackey.py: contiene un ejemplo que utiliza ANFIS para la predicción de la serie Mackey Glass. Este ejemplo entrena el sistema en 1500 puntos de la serie y traza la serie real frente a la predicha, las curvas de aprendizaje y las funciones de pertenencia resultantes después del entrenamiento.

Una vez adaptadas las librerías de Python se procederá con el desarrollo de la red neuronal difusa en Linux – Ubuntu, así como para la obtención de resultados, desarrollo que se especifica a continuación.

3.2.2 Arquitectura y tecnología utilizada para el desarrollo de la red.

Arquitectura del Modelo

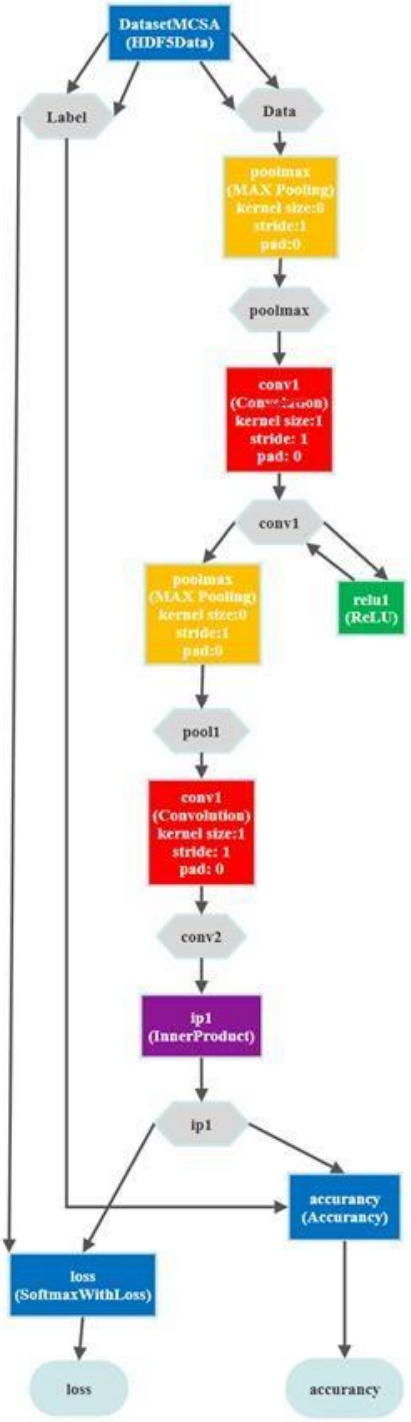


Fig. 3.7 Arquitectura del modelo desarrollado

Fuente: Autor

Explicación de la arquitectura

Para la arquitectura del modelo a utilizar se utiliza max-pooling, para facilitar el entrenamiento. El mismo que permite capturar una muestra completa de datos y dividirla en pequeñas fracciones para un posterior procesamiento, cual permite tomar una muestra completa y reducirla en pequeñas fracciones para posteriormente analizarlas. La capa conv1, son utilizadas para la obtención de características en las muestras, las cuales permitirán su clasificación.

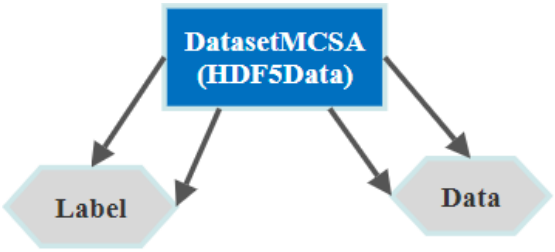


Fig. 3.8 Ingreso de los dataset a la red

Fuente: Autor

De acuerdo con la Fig. 3.7, al ingresar las muestras para cada análisis se carga el dataset de la muestra, así como el que posee los identificadores de las clases, pero el único que entra a la red para ser analizado es el que contiene la muestra.

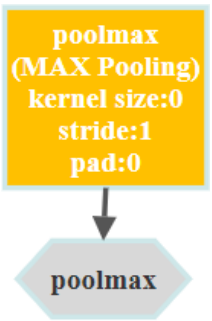


Fig. 3.9 Aplicación del max-pooling

Fuente: Autor

De acuerdo con la figura 3.8, se aplicará un max-pooling al inicio del análisis con el fin de reducir las dimensiones de las muestras de entrada en este caso a 1x5. Logrando así dividir la muestra para un procesamiento más rápido y exacto.

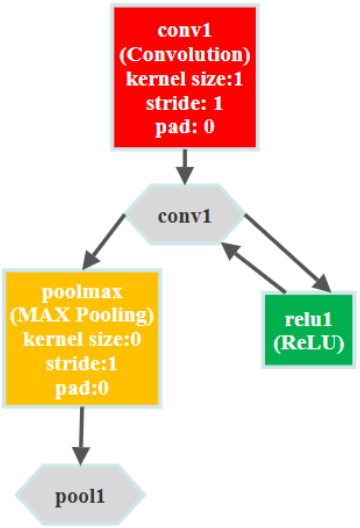


Fig. 3.10 Ingreso a la primera capa de convolución y reducción de la muestra Fuente: Autor

La muestra reducida ingresa en la capa conv1, para obtener las características que se necesita para la clasificación. Después de este proceso la muestra pasa por la función de activación ReLu la cual se encarga de transformar los valores negativos a 0 sin alterar los valores positivos. Pasado este proceso se aplica otro max-pooling para obtener otra muestra reducida y analizarla en la siguiente capa de convolución. En la Fig. 3.8 se encuentra la primera capa de convolución, la función de activación ReLU y la segunda capa de pooling.

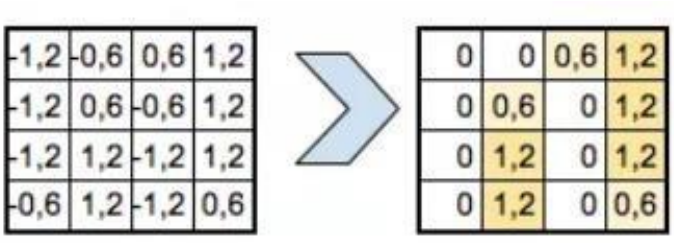


Fig. 3.11 Ejemplo de muestra después de la aplicación de ReLu

Fuente: Autor

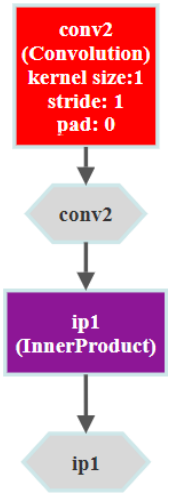


Fig. 3.12 Segunda capa de convolución

Fuente: Autor

Como datos de entrada para la segunda capa de convolución se toma la muestra normalizada y reducida de la capa anterior. Las muestras resultado obtenidas de la aplicación de la segunda convolución se van guardando en el espacio de producto interno para posteriormente ser clasificadas. La Fig. 3.10 muestra la segunda capa de convolución y la capa de InnerProduct.

Fig. 3.13 Etapa de clasificación de las muestras

Fuente: Autor

Una vez terminado todo el análisis de la muestra se procede con el análisis del conjunto de vectores para de esta manera clasificarlo en la clase correcta, aquí se vuelve a involucrar el dataset “label” que contiene las clases que se van a analizar. Esta sección se encarga de determinar la precisión de acierto que tendrá nuestra red al clasificar muestras. La Fig. 3.11 contiene la capa de SoftMax la cual se aplica antes de mostrar el resultado, esta capa posee la misma cantidad de nodos resultados, debido a que se encarga de calcular las probabilidades de todas las clases para que sumen 1.0.

Arquitectura Red Neuronal Difusa (ANFIS)

Una red neuronal Difusa por lo general consta de tres capas divididas de la siguiente manera capa de entrada, capas ocultas, capa de salida, dentro de las capas ocultas pueden aparecer una o más capaz como es para el caso para el modelo ANFIS que se utiliza para el desarrollo del presente proyecto.

La capa de entrada es la encargada de introducir los datos que se van a utilizar para su posterior procesamiento y entrenamiento.

Las capas ocultas son las encargas de realizar el procesamiento y entrenamiento de las capas anteriores para el caso de ANFIS se utilizan 3 capas ocultas.

La capa de salida es la encargada de presentar los resultados una vez procesados y entrenados los datos de muestra, para el caso de ANFIS en donde se obtiene una salida global.

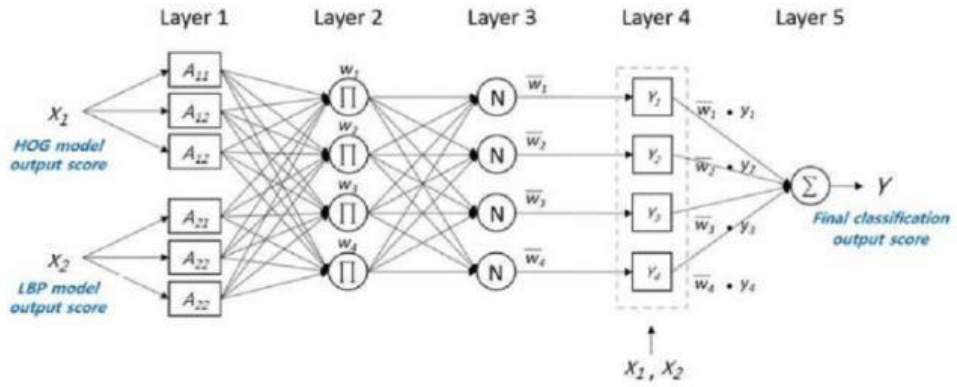


Fig. 3.14: Arquitectura ANFIS general desarrollada

Fuente: [15]

Proceso de la Arquitectura

Una vez implementado el modelo ANFIS en el desarrollo del proyecto, además, se utilizará parámetros estadísticos, al ingresar al modelo ANFIS se procesa mediante reglas, al utilizarse el método Sugeno las reglas se generan de forma automática en el entrenamiento.

En la arquitectura ANFIS se utilizan 5 Capas como se muestra en la Fig. 3.12. y al no existir una librería propia de c++, se utilizará el método TensorFlow en Python, de la siguiente manera.

fis = anfis(trainingData)

fis = anfis(trainingData,options)

[fis,trainError] = anfis(___)

[fis,trainError,stepSize] = anfis(___)

[fis,trainError,stepSize,chkFIS,chkError] = anfis(trainingData,options)

fis = anfis(trainingData) genera un sistema de inferencia difusa (FIS) de Sugeno de salida única y ajusta los parámetros del sistema utilizando los datos de entrenamiento de entrada/salida especificados. El objeto FIS se genera automáticamente utilizando particiones de cuadrícula.

El algoritmo de entrenamiento utiliza una combinación de los métodos de descenso de gradiente de mínimos cuadrados y de propagación hacia atrás para modelar el conjunto de datos de entrenamiento.

fis = anfis(trainingData,options) sintoniza un FIS utilizando los datos y las opciones de entrenamiento especificados. Con esta sintaxis, puede especificar:

- Un objeto inicial de FIS para sintonizar.
- Datos de validación para evitar el sobreajuste a los datos de entrenamiento.
- Opciones de algoritmos de entrenamiento.
- Si se debe mostrar información sobre el progreso del entrenamiento.

[fis,trainError] = anfis(__) devuelve la raíz del error cuadrático medio de entrenamiento para cada época de entrenamiento.

[fis,trainError,stepSize] = anfis(_) devuelve el tamaño del paso de entrenamiento en cada época de entrenamiento.

[fis,trainError,stepSize,chkFIS,chkError] = anfis(trainingData,options) devuelve el error de validación de datos para cada época de entrenamiento, chkErrory el objeto de esquí alpino en sintonía para la que el error de validación es mínimo, chkFIS. Para usar esta sintaxis, debe especificar los datos de validación usando options.ValidationData.

Función de Activación

La función de activación es la que permite obtener un resultado global lineal partiendo de las muestras de entrada**.** Al utilizar el modelo ANFIS posee diversas funciones de activación, la seleccionada y que mejor se adapta al proyecto es la función Gaussiana, esta función calcula los valores de membresía difusos utilizando una función de membresía gaussiana, al utilizar esta membresía siempre tiene un valor máximo de 1 ajustando los parámetros dentro del rango establecido.

Parámetros de la arquitectura de red

La red neuronal difusa al tener una arquitectura no tan definida utiliza los siguientes parámetros estadísticos:

- Media
- Desviación Estándar
- Varianza
- Curtosis
- Asimetría estadística
- Media cuadrática

Al utilizar la Arquitectura del modelo ANFIS se utiliza menos parámetros, es decir no se procede con la utilización total de la FFT, sino se utilizan los parámetros ya descritos.

3.2.3 Tecnologías utilizadas y funcionamiento de la red neuronal difusa.

El desarrollo del presente proyecto fue realizado en Linux – Ubuntu debido a las ventajas que este posee, el lenguaje de programación a utilizar es c++, con algunas librerías y métodos de Python debido al uso del modelo ANFIS mediante Sugeno, es decir las reglas se automatizan de acuerdo con los parámetros que utiliza el sistema. Una de las ventajas de la utilización del lenguaje de programación c++ es la manipulación de objetos y un lenguaje más interactivo además de ser muy dinámico.

Para poder desarrollar el proyecto se utilizaron librerías desarrolladas en Python debido a la arquitectura ANFIS, además de la librería Boost para realizar los cálculos estadísticos mediante los acumuladores, la librería fstream para ingresar los datos mediante el formato .bat y la librería H5 que nos permitirá leer todos los dataset generados con anterioridad. También se utiliza la librería Eigen que nos permite el uso del algoritmo FFT. Como ultimo paso se utiliza la librería QT creator para la elaboración y diseño de interfaces gráficas.

QT Creator

Fig. 3.15: QT Creator

Fuente: Autor

Al utilizar Linux la herramienta más didáctica que se utilizar es QT Creator que se utiliza mediante la instalación de la librería, es un lenguaje multiplataforma fácil y que cuenta con características que facilitan el diseño y aumentan la productividad, al decir que es un lenguaje multiplataforma nos referimos soporta ambientes de desarrollo como c++. QML y JavaScript.

Al hablar de QT Creator hablamos de una herramienta útil para c++, que se puede presentar de dos formas, la primera es mediante líneas de código que resulta un poco más complejo el diseño, para la segunda forma se puede utilizar una herramienta gráfica, la cual se implementó en el presente proyecto Qt Designer, ayudándonos a crear las interfaces de forma ágil, rápida y sobre todo amigable para el usuario.

Librerías para la creación del Dataset

Dentro del desarrollo del dataset se obtiene el código a partir de los parámetros utilizados mediante una ruta establecida, en este caso para el desarrollo de la arquitectura ANFIS no existe una librería propia de c++, es por eso que se procede a utilizar la librería por Tiago Cuervo en la que se utiliza el TensorAnfis, esta es una implantación simple de un sistema de inferencia difuso basado en nuero adaptativo(ANFIS) en TensorFlow, los principales requisitos para este tensor en Python, Tensorflow, Numpy, Matplotlib desarrollado en Python, una vez importa esta librería el requisito para que el modelo acepte los datasets debe tener un formato de tipo .bat, para que su generación y lectura sean más agiles. A continuación, se presenta el importe de librerías estadísticas.

```
#include <iostream>
#include <string>
#include <boost/math/distributions.hpp>
#include <boost/accumulators/accumulators.hpp>
#include <boost/accumulators/statistics/stats.hpp>
#include <boost/accumulators/statistics/mean.hpp>
#include <boost/accumulators/statistics/variance.hpp>
#include <boost/accumulators/statistics/kurtosis.hpp>
#include <boost/accumulators/statistics/skewness.hpp>
#include <boost/accumulators/statistics/min.hpp>
#include <boost/accumulators/statistics/max.hpp>
#include <boost/accumulators/statistics/sum.hpp>
#include <boost/accumulators/statistics/count.hpp>
#include <fstream>
#include <boost/filesystem.hpp>
#include <H5Cpp.h>
#include <H5File.h>
#include <cstdlib>
```

Fig. 3.16: Librerías para el procesamiento de muestras mediante el Dataset.

Fuente: Autor

Adicional a las librerías especificadas se utilizan también fstream para que la lectura del dataset en formato .bat, la librería boost para poder procesar la parte estadística a través de sus acumuladores y la librería H5 que nos permitirá leer los dataset generados con anterioridad.

Métodos para la creación de muestras simuladas en condiciones Normales y para los tipos de fallos.

Para la generación de muestras simuladas se desarrolló un programa en C++ que genera datos con la estructura que soporta la librería Caffe. Como el programa es dinámico se puede generar un número personalizado de ondas, para lo cual es necesaria la modificación de los parámetros dados.

Los parámetros del generador de ondas requieren valores específicos los cuales serán especificados a continuación:

Parámetros	Descripción
--a	Especificar el valor de la amplitud de la onda
--fm	Especificar el valor de la frecuencia de muestreo Hz
--t	Especificar el número total de muestras, maximo1000
--n	Especificar el número de puntos por muestra
--f	Especificar el nombre de archivo de salida
--e	Especificar el tipo de error
-sincomprobacion	Ejecutar programa sin comprobar datos
-sincsv	Para la no generación de archivos csv

Tabla 3.30 Parámetros para generar muestras.

Fuente: Autor

Para compilar este programa se debe acceder mediante el terminal a su ubicación y ahí se debe ejecutar el comando con los parámetros que deseamos como en el siguiente ejemplo:

./GenerarOndas --a 3 --t 400 --n 512 --f prueba1 --sincsv

Con este comando se generará una onda de 3 de amplitud la cual tendrá 400 muestras en total, 512 números de puntos por muestra, el nombre del documento será prueba1 y no se generará ningún archivo csv.

El código para la generación de muestras está desarrollado en base a un vector de vectores de tipo flotante bajo el nombre de muestras. También posee un generador al azar llamado “dist” el cual será utilizado para la simulación del ruido blanco. Así como una sección para el cálculo de cada uno de los datos de la onda.

Se utilizan dos ciclos “for” para el ingreso de datos en los dos vectores uno de ellos utilizado para llenarlo con los datos de la onda en función de tiempo en esta sección se añade el ruido blanco a la onda, en el otro por medio de la función “push_back” se le llenara el vector con las muestras generadas.

```
154     //Formato, primera columna el label, las demás los datos
155     //Primera etapa, obtener los puntos
156     std::vector<std::vector<float>> muestras;
157     std::vector<float> muestra;
158     const double mean = 0.0;
159     const double stddev = amplitud/10;
160     std::default_random_engine generator;
161     std::normal_distribution<double> dist(mean, stddev); //Generador de ruido blanco
162     cout << "Generando muestras" << endl;
163     float tiempomuestreo = 1/frecuenciamuestreo;
164     cout << "tiempo de muestreo: " << tiempomuestreo << endl;
165     float argumento  = 0;
166     //Generacion Dataset
167     const H5std_string FILE_NAME(nombrearchivo + ".h5");
168     const H5std_string DATASET_NAME1("data");
169     const H5std_string DATASET_NAME2("label");
170
171     for(int i = 0; i < totalnumeromuestras; i++){
172         argumento = (rand() % 200 - 100)/100.0;
173         muestra.clear();
174         for(int j = 0; j < numerodepuntospormuestra; j++){
175             muestra.push_back(amplitud*cos(frecuencia*argumento) + dist(generator));//Calculo
176             argumento += tiempomuestreo;
177         }
178         muestras.push_back(muestra);
179     }
180     cout << "Finalizo de generar muestras" << endl;
```

Fig. 3.17 Generación de muestras de funcionamiento normal

Fuente: Autor

Usando el mismo procedimiento se generan las muestras para barras rotas.

```
200     cout << "Introduciendo error por barras rotas" << endl;
201     //introducir falla por barra rota
202     for(size_t i = 0; i < tamanhovector; i++){
203         argumento = (rand() % 200 - 100)/100.0;
204         muestra.clear();
205         for(size_t j = 0; j < muestras.at(i).size(); j++){
206             muestra.push_back(muestras.at(i).at(j) + Ilsb*cos((1-2*s)*frecuencia*tiempomuestreo)+Iusb*cos((1+2*s)*frecuencia*tiempomuestreo));
207             argumento += tiempomuestreo;
208         }
209         muestras.push_back(muestra);
210     }
211     cout << "Finalizo de generar muestras" << endl;
212     if(generarcsv){
213         archivofunciontiempo.open("muestrabarrasrotas" + nombrearchivo +".csv");
214         for(size_t i = 0; i < muestras.at(muestras.size() - 1).size(); i++){
215             time (&rawtime);
216             timeinfo = localtime (&rawtime);
217             tiempo = asctime(timeinfo);
218             archivofunciontiempo  << muestras.at(muestras.size() - 1).at(i) << "," << tiempo;
219             usleep(tiempomuestreo*1000000);
220
221         }
222         archivofunciontiempo.close();
223     }
```

Fig. 3.18 Generación de muestras de barras rotas

Fuente: Autor

Para el fallo de asimetría de eje se utiliza el mismo procedimiento para generar y guardar las muestras.

```
//introducir falla por asinotria del eje
cout << "Generando muestras de asinetria del eje" << endl;
for(size_t i = 0; i < tananhovector; i++){
    argumento = (rand() % 200 - 100)/100.0;
    muestra.clear();
    for(size_t j = 0; j < muestras.at(i).size(); j++){
        muestra.push_back(muestras.at(i).at(j) + Ilsb*cos((frecuencia - wr)*tiemponuestreo)+Iusb*cos((frecuencia + wr)*frecuencia*tiempomuestreo));
        argumento += tiemponuestreo;
    }
    muestras.push_back(muestra);
}
if(generarcsv){
    archivofunciontiempo.open("muestrasasimetria" + nombrearchivo + ".csv");
    for(size_t i = 0; i < muestras.at(muestras.size() - 1).size(); i++){
        time (&rawtime);
        timeinfo = localtime (&rawtime);
        tiempo = asctime(timeinfo);
        archivofunciontiempo  << muestras.at(muestras.size() - 1).at(i) << "," << tiempo;
        usleep(tiempomuestreo*1000000);

    }
    archivofunciontiempo.close();
}
```

Fig. 3.19 Generación de muestras de asimetría de eje

Fuente: Autor

3.2.4 Desarrollo y descripción de obtención de los parámetros utilizados para la red.

Como se describió los parámetros a utilizar dentro de la red neuronal difusa, se utilizan parámetros estadísticos que se obtienen de la siguiente manera:

El primer paso es ejecutar una línea de comando en donde se especifique el nombre del archivo que contiene las muestras de entrada, para el desarrollo de este proyecto se ha utilizado diferentes archivos, en donde están clasificadas las muestras, en consecuencia, es necesario ordenarlas en una lista, que fácilmente se las puede generar mediante el comando ls > nombre.lst dentro del directorio creado llamado muestras y el nombre de salida del dataset. Una vez realizado el proceso descrito, procedemos a escribir el siguiente comando:

./ObtenerParametros --f datosentrenamiento.hi5list --o dataset.dat

Después de haber ejecutado el comando tendremos como resultado un archivo que contiene los nombres de los dataset existentes con el siguiente formato.

Fig. 3.20: Nombres de los dataset de entrenamiento

Fuente: Autor

Cada uno de estos archivos de texto consta con los parámetros estadísticos predeterminados de los archivos HDF5, después de esto se los guarda en un acumulador para su posterior procesamiento mediante las operaciones estadísticas y su obtención de parámetros. También se calculan máximos y mínimos para saber qué tipo de parámetro nos podemos encontrar y poderlos clasificar por clases.

```
108     vector<vector<ParametrosEstadisticos>> caracteristicas(clases, vector<ParametrosEstadisticos>()); //clases, matematicas
109     vector<vector<ParametrosEstadisticos>> caracteristicasTestSet(clases, vector<ParametrosEstadisticos>()); //clases, matematicas
110     ParametrosEstadisticos parametrosindividual;
111     while (std::getline(archivoentrada, linea))
112     {
113         if(linea.length()==0) break;
114         cout <<  "Leyendo rutar " << canonical(rutaRelativa).string() + "/" + linea << endl;
115         const H5std_string  nombreArchivoDataset(canonical(rutaRelativa).string() + "/" + linea);
116         try{
117             H5File archivo(nombreArchivoDataset, H5F_ACC_RDWR);
118             DataSet datasetData = archivo.openDataSet(NombreDatasetData);
119             DataSet datasetLabel = archivo.openDataSet(NombreDatasetLabel);
120             DataSpace filespaceData = datasetData.getSpace();
121             DataSpace filespaceLabel = datasetLabel.getSpace();
122             int rankData = filespaceData.getSimpleExtentNdims();
123             int rankLabels = filespaceLabel.getSimpleExtentNdims();
124             hsize_t dimslabels[RANK], dimsdata[RANK];    // dataset dimensions
125             rankLabels = filespaceLabel.getSimpleExtentDims( dimslabels );
126             rankData = filespaceData.getSimpleExtentDims( dimsdata );
127             DataSpace mSpaceData(RANK, dimsdata);
128             DataSpace mSpaceLabels(RANK, dimslabels);
129             double datosLabel[dimslabels[0]][dimslabels[1]][dimslabels[2]][dimslabels[3]];
130             double datosData[dimsdata[0]][dimsdata[1]][dimsdata[2]][dimsdata[3]];
131
132             datasetData.read( datosData, PredType::NATIVE_DOUBLE, mSpaceData, filespaceData );
133             datasetLabel.read( datosLabel, PredType::NATIVE_DOUBLE, mSpaceLabels, filespaceLabel );
134             accumulator_set<float,stats<tag::min, tag::max, tag::sum, tag::count, tag::mean, tag::kurtosis, tag::variance, tag::skewness>> acumulador; //Sirve para obtener las ca
135             for(size_t i = 0; i < dimsdata[0]-2; i++){
136                 acumulador = {};
137                 for(size_t j = 0; j < dimsdata[3]; j++){
138                     //cout << "Linea: " << linea << "i: " << i  << " - " << dimsdata[0] << " j: " << j << "-" << dimsdata[3]  << " Datos:[   " << i << "][" << dimsdata[1] << "]["
139                     acumulador(datosData[i][dimsdata[1]][dimsdata[2]][j]);
140                 }
141                 parametrosindividual.media = mean(acumulador);
142                 parametrosindividual.kurtois = kurtosis(acumulador);
143                 parametrosindividual.desviacionestandar = sqrt(varianza(acumulador));
144                 parametrosindividual.varianza = variance(acumulador);
145                 parametrosindividual.skewness = skewness(acumulador);
146                 parametrosindividual.rms = sqrt(abs(sum(acumulador))/boost::accumulators::count(acumulador));
147                 caracteristicas.at(int(datosLabel[i][dimslabels[1]][dimslabels[2]][dimslabels[3]])).push_back(parametrosindividual);
148             }
149         }  catch(FileIException error)
150         {
151             error.printErrorStack();
152             return -1;
153         }
154
155     }
```

Fig. 3.21: Parámetros estadísticos que se obtienen

Fuente: Autor

Después se procede a desarrollar un vector de salida para obtener los parámetros estadísticos con los que se va a trabajar, el nombre del vector se lo asigna de acuerdo a como el desarrollador lo considere necesario como se ve a continuación.

```
216     for(size_t i = 0; i < caracteristicas.size(); i++){
217         int totalrandom = 0;
218         int totalmuestras = caracteristicas.at(i).size();
219         vector<int> vectorandoms;
220         while(totalrandom < caracteristicas.at(i).size()/5){
221             vectorandoms.push_back(rand() % totalmuestras);
222             totalrandom+=1;
223             totalmuestras-=1;
224         }
225         for(size_t j = 0; j < vectorandoms.size(); j++){
226             caracteristicasTestSet.at(i).push_back(caracteristicas.at(i).at(vectorandoms.at(j)));
227             caracteristicas.at(i).erase(caracteristicas.at(i).begin() + vectorandoms.at(j));
228         }
229     }
230     obtenerDatasets("train_" + nombresalida, caracteristicas);
231     obtenerDatasets("test" + nombresalida, caracteristicasTestSet);
232     return 0;
233 }
234
```

Fig. 3.22: Vector de Salida

Fuente: Autor

Una vez definidos los parámetros y el vector de salida procedemos con la ejecución, obteniendo los resultados de la siguiente manera:

```
********** Clase 0**********
(Media) Maximo: 1.43359 Media: -0.273742 Minimo: -0.460392
(Desviacion Estandar) Maximo: 0.495571 Media: 0.199284 Minimo: 0.146014
(Varianza) Maximo: 0.24559 Media: 0.040202 Minimo: 0.0213201
(Kurtois) Maximo: 22.9354 Media: 7.52436 Minimo: -1.92818
(Skewness) Maximo: 2.48522 Media: 0.639355 Minimo: -0.685338
(RMS) Maximo: 1.19733 Media: 0.52022 Minimo: 0.139586
**********
********** Clase 1**********
(Media) Maximo: -0.19962 Media: -0.491672 Minimo: -0.637408
(Desviacion Estandar) Maximo: 0.178864 Media: 0.153337 Minimo: 0.126709
(Varianza) Maximo: 0.0319924 Media: 0.0235606 Minimo: 0.0160551
(Kurtois) Maximo: 32.7017 Media: 21.6671 Minimo: 9.48624
(Skewness) Maximo: 3.56398 Media: 2.22575 Minimo: 0.590819
(RMS) Maximo: 0.798378 Media: 0.698481 Minimo: 0.446788
**********
********** Clase 2**********
(Media) Maximo: -0.173011 Media: -0.471299 Minimo: -0.620661
(Desviacion Estandar) Maximo: 0.178556 Media: 0.158123 Minimo: 0.129736
(Varianza) Maximo: 0.0318823 Media: 0.0250561 Minimo: 0.0168315
(Kurtois) Maximo: 28.7698 Media: 19.0524 Minimo: 11.7317
(Skewness) Maximo: 3.29872 Media: 2.01505 Minimo: 0.421534
(RMS) Maximo: 0.78782 Media: 0.683543 Minimo: 0.415946
**********
```

Fig. 3.23: Nombres de las clases de dataset de entrenamiento

Fuente: Autor

Una vez ejecutada la sentencia obtenemos el archivo test.dat en el cual se representa mediante cada columna el parámetro y el séptimo digito es la clase (0, 1 o 2).

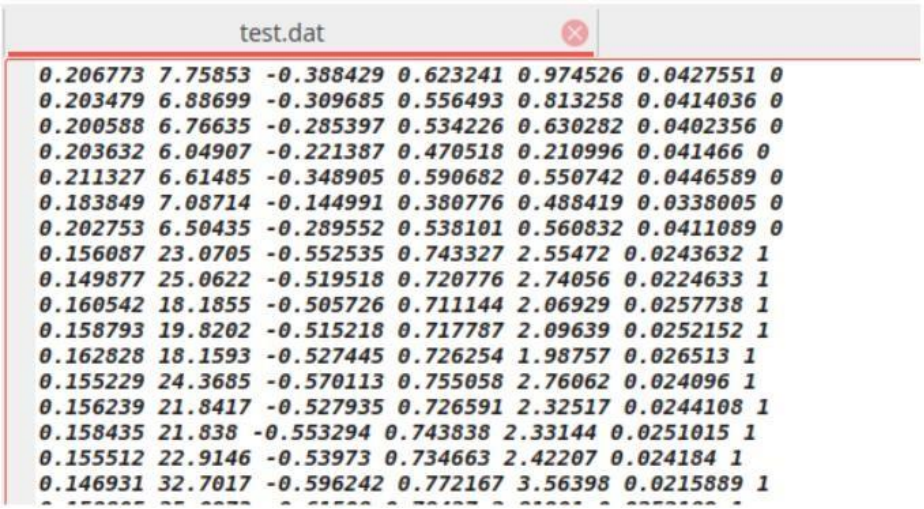


Fig. 3.24: Archivo de salida test.dat de entrenamiento

Fuente: Autor

3.2.5. Entrenamiento de la red neuronal difusa

Como está establecido para el desarrollo y ejecución de la red neuronal difusa se utiliza el método ANFIS con Sugeno, para poder implementar el modelo ANFIS se utilizan librerías de Python mediante TensorFlow, debido a que en c++ no hay una implementación fiable para este método, entonces como primer paso se establecen los parámetros de la Arquitectura ANFIS:

Número de reglas: 96

Número de entradas: 6

Ratio de entrenamiento alfa: 0.01

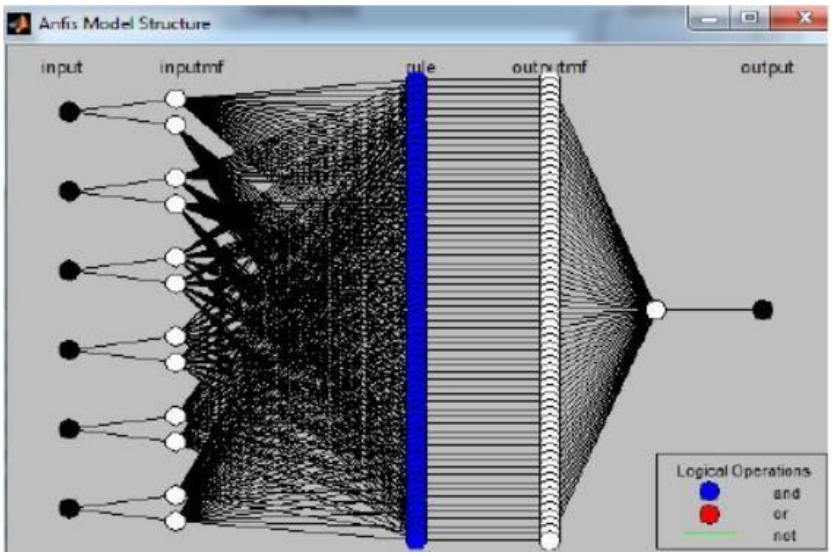


Fig. 3.25: Estructura del modelo ANFIS

Fuente: [20]

Descripción de reglas a utilizar generadas automáticamente mediante el método de Sugeno y en referencia al artículo ¨Vibration Based Fault Detection of Centrifugal Pump by Fast Fourier Transform and Adaptive Nuero-Fuzzy Inference System¨ [20] en el que se propone clasificar los datos de muestras en los diferentes tipos de fallo a analizar, en el caso de este proyecto se clasifica en normal, barras rotas y asimetría de eje, además de los datos de entrenamiento del modelo ANFIS se utiliza estas muestras para verificar la precisión y efectividad del modelo para los dos tipos de falla y el funcionamiento normal de la red neuronal difusa.

Como se describe en la Fig. 3.19 se utiliza 6 conjuntos de entrada, procesando los parámetros de las muestras a utilizar con un paro a las 150 iteraciones.

Al final de las 150 iteraciones la curva de convergencia del error de la red se obtiene de acuerdo como se muestra en la Fig.3.20, además de reglas se obtuvieron de la siguiente manera:

Rule 1. If (input1 is in1mf1) and (input2 is in2mf1) and (input3 is in3mf1) and (input4 is in4mf1) and (input5 is in5mf1) and (input6 is in6mf1) then (output is out1mf1) (1)

Rule 2. If (input1 is in1mf1) and (input2 is in2mf1) and (input3 is in3mf1) and (input4 is in4mf1) and (input5 is in5mf1) and (input6 is in6mf2) then (output is out1mf2) (1)

· ……

· ……

· **Rule 96**. If (input1 is in1mf1) and (input2 is in2mf1) and (input3 is in3mf1) and (input4 is in4mf1) and (input5 is in5mf1) and (input6 is in6mf2) then (output is out1mf2) (1)

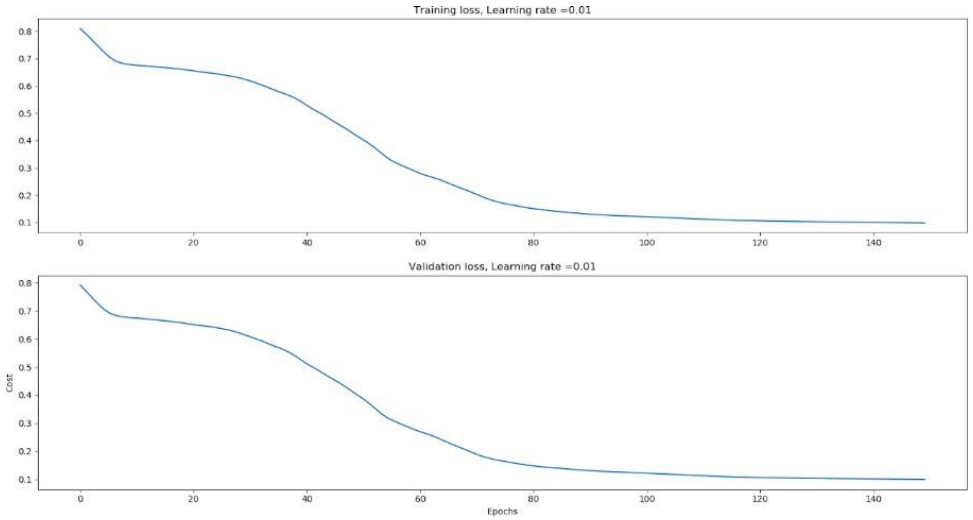
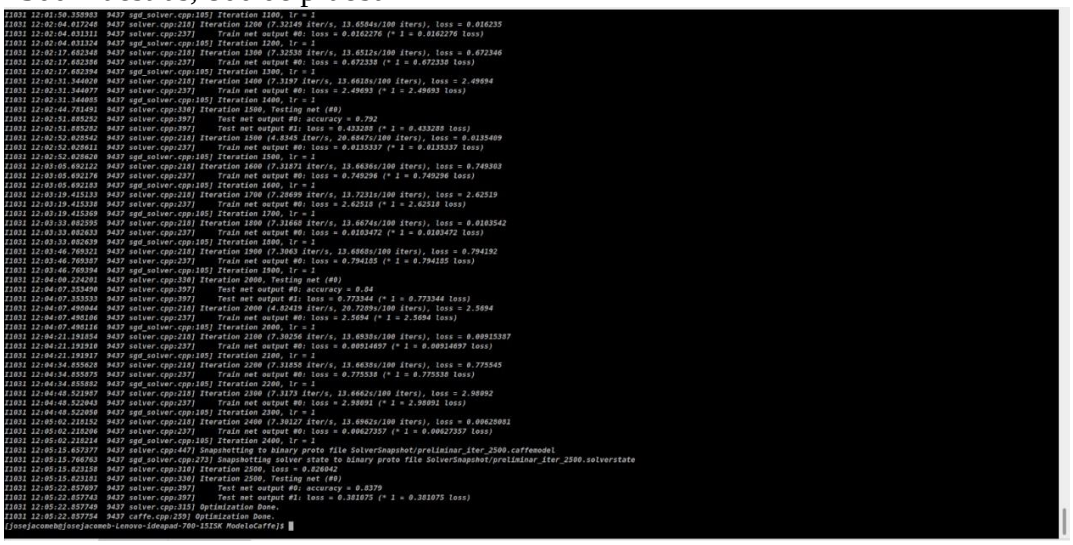


Fig. 3.26: Entrenamiento y Validación de las iteraciones

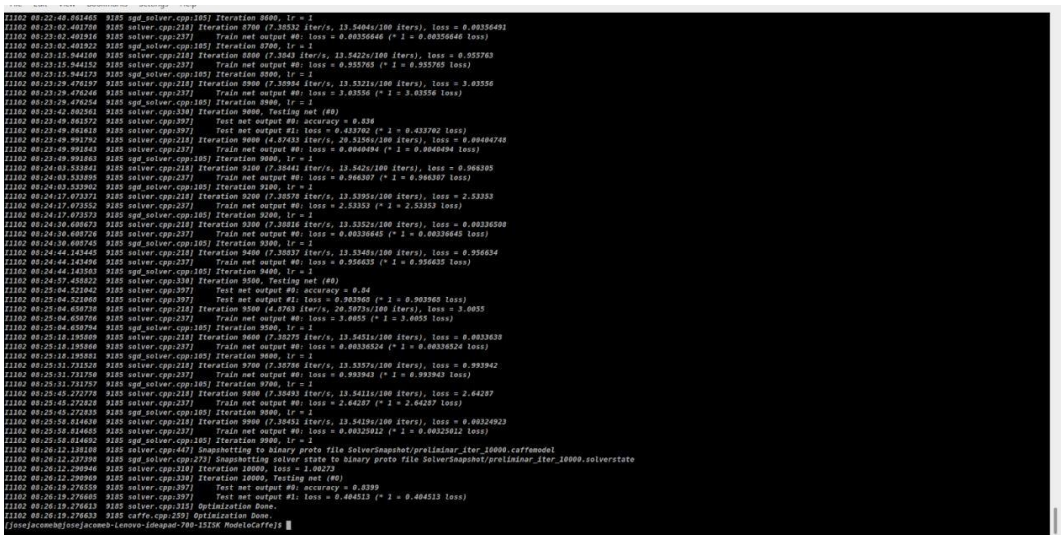
Fuente: Autor

Resultados de entrenamiento

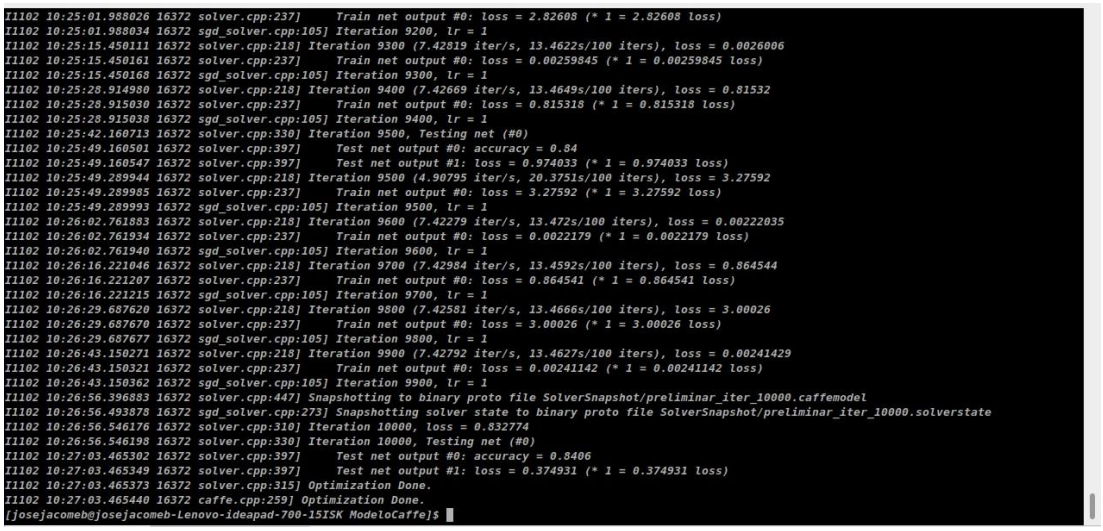
2500 muestras, 800 de prueba



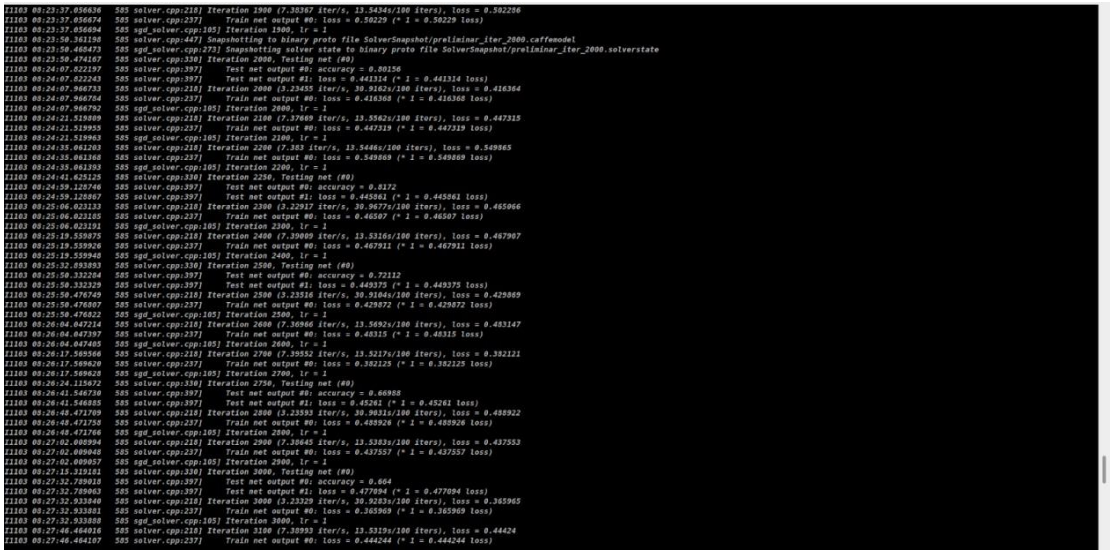
40000 muestras, 8000 de prueba



80% de acierto



64 000 muestras 94% de acierto



3.2.6. Desarrollo y descripción del cogido para la creación de la red

En el desarrollo de la red se utilizan librerías que no son propias de c++, es decir se importan de Python como la librería Argparse, descrita de la siguiente manera:

El código de la fuente de la librería es la siguiente argparse.py, argparse es un módulo Python que facilita la escritura de interfaces de línea de comandos, el sistema define que parámetros requiere y este los analiza a través de sys.argv, una de las ventajas de este módulo es que permite al usuario visualizar mensajes de error cuando a este se le asignan parámetros no válidos.

```
import argparse

parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('integers', metavar='N', type=int, nargs='+',
                    help='an integer for the accumulator')
parser.add_argument('--sum', dest='accumulate', action='store_const',
                    const=sum, default=max,
                    help='sum the integers (default: find the max)')

args = parser.parse_args()
print(args.accumulate(args.integers))
```

Fig. 3.27: Modulo argparse

Fuente: Autor

En la Fig. 3.12 se visualiza la lectura de numero enteros para realiza una suma o el máximo valor asignado.

Una vez realizo este proceso se lo guarda en un archivo llamado prop.py que se lo ejecuta dentro de una línea de comandos, dándonos mensajes de ayuda muy útiles.

```
$ python prog.py -h
usage: prog.py [-h] [--sum] N [N ...]

Process some integers.

positional arguments:
 N           an integer for the accumulator

optional arguments:
 -h, --help  show this help message and exit
 --sum       sum the integers (default: find the max)
```

Fig. 3.28: Archivo porp.py del módulo argparse

Fuente: Autor

Al ejecutarse el archivo prop.py se obtiene la suma o el máximo de parámetros asignados siempre y cuando estos sean los correctos.

```
$ python prog.py 1 2 3 4
4

$ python prog.py 1 2 3 4 --sum
10
```

Fig. 3.29: Ejecución Modulo argparse

Fuente: Autor

Una vez ejecutado la línea de comando se obtiene el resultado y al no mandar los parámetros adecuados obtenemos un mensaje de error.

```
$ python prog.py a b c
usage: prog.py [-h] [--sum] N [N ...]
prog.py: error: argument N: invalid int value: 'a'
```

Fig. 3.30: Ejecución Modulo argparse

Fuente: Autor

Una vez conocido los parámetros a utilizar, además de haber realizado el entrenamiento se procede con el llamado al método denominado por el archivo MCSAAnfis.py en el cual se especifican cada uno de los pasos:

Para poder llamar al archivo MCSAAnfis.py es necesario llamar una línea de código dentro del mismo directorio donde se encuentra el script:

./MCSAAnfis.py --train ./../train.dat --test ./../test.dat --entradas 6 --reglas 96 --alfa 0.01

Una vez ejecutado se carga el archivo de entrenamiento para realizar la inferencia y el de comprobación para saber el acierto de la red. Para realizar este proceso es necesario establecer la versión del Python en la cabecera que este caso será Python3, se importan las librerías de Tensorflow y Argparse, la primera establece el modelo y la segunda para obtener los parámetros de entrada establecidos por el usuario.

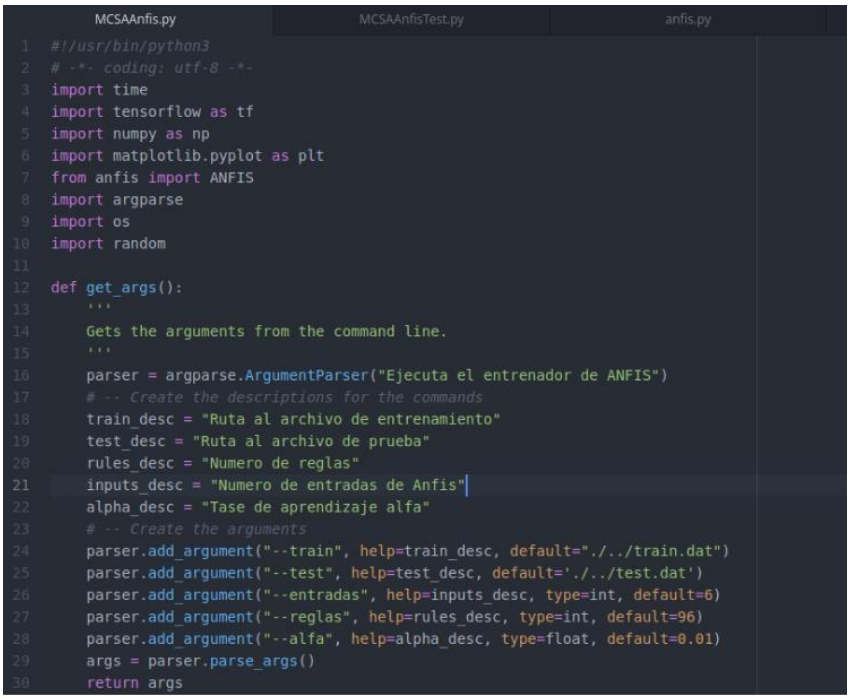


Fig. 3.31: Obtención de argumentos

Fuente: Autor

En la Fig. 3.26 se definen las librerías a utilizar, después se crea el método get_args que nos ayudara a obtener los argumentos del entrenador ANFIS, en donde se establece la ruta del archivo de entrenamiento, la ruta del archivo de prueba, el número de reglas establecidas, las entradas del modelo ANFIS y por último la tasa de aprendizaje.

Una vez obtenidos los argumentos dentro del programa main, se leen los datos de los archivos .bat y se randomiza las entradas, para que el entrenamiento sea más preciso y no tenga fallos de overflow.

```python
def main():
    args = get_args()
    readtrnData = []
    readtrnLbls = []
    trnData = []
    trnLbls = []
    chkData = []
    chkLbls = []
    f = open(args.train, "r")
    for x in f:
        lista = x.rstrip().split(" ")
        lista = [float(i) for i in lista]
        readtrnData.append(lista[:args.entradas])
        readtrnLbls.append(int(lista[args.entradas]))
    f.close()
    print("Empieza a randomizar")
    while len(readtrnData) > 0:
        numero_aleatorio = int(random.random()*len(readtrnData))
        trnData.append(readtrnData[numero_aleatorio])
        trnLbls.append(readtrnLbls[numero_aleatorio])
        readtrnData.pop(numero_aleatorio)
        readtrnLbls.pop(numero_aleatorio)
    print("Salio")
    f = open(args.test, "r")
    for x in f:
        lista = x.rstrip().split(" ")
        lista = [float(i) for i in lista]
        chkData.append(lista[:args.entradas])
        chkLbls.append(lista[args.entradas])
    f.close()
```

Fig. 3.32: Metodo main

Fuente: Autor

Una vez establecidos todos los parámetros y llamados y randomizado en el método main se procede a ejecutar el ciclo for para realizar el entrenamiento, este está configurado para detenerse cuando se haya realizado 150 iteraciones.

```python
nombrearchivo = 'MCSA_ANFIS'
num_epochs = 150
fis = ANFIS(n_inputs=args.entradas, n_rules=args.reglas, learning_rate=args.alfa)
with tf.compat.v1.Session() as sess:
    sess.run(fis.init_variables)
    saver = tf.train.Saver()#, v3: "y/Assign", v4: "beta1_power/Assign"})
    trn_costs = []
    val_costs = []
    time_start = time.time()
    for epoch in range(num_epochs):
        #  Run an update step
        trn_loss, trn_pred = fis.train(sess, trnData, trnLbls)
        # Evaluate on validation set
        val_pred, val_loss = fis.infer(sess, chkData, chkLbls)
        if epoch % 10 == 0:
            print("Train cost after epoch %i: %f" % (epoch, trn_loss))
            #print("Weight matrix: {1}".format(sess.run(variables[1])))
            #print("Weight matrix: {2}".format(sess.run(variables[2])))
        if epoch == num_epochs - 1:
            time_end = time.time()
            print("Elapsed time: %f" % (time_end - time_start))
            print("Validation loss: %f" % val_loss)
            # Plot real vs. predicted
            pred = np.vstack((np.expand_dims(trn_pred, 1), np.expand_dims(val_pred, 1)))
            plt.figure(1)
            plt.plot(pred)
        trn_costs.append(trn_loss)
        val_costs.append(val_loss)
    saver.save(sess, nombrearchivo)
```

Fig. 3.33: ANFIS y ciclo for para realizar iteraciones.

Fuente: Autor

Después de haber realizado todo este proceso se guarda el archivo con el nombre de MCSA_ANFIS, que a su vez este contiene 4 archivos más:

Fig. 3.34: Archivos de salida de MCSA_ANFIS

Fuente: Autor

Una vez que el archivo MCSAAnfis.py se ejecuta, al final sale el modelo, que tiene los pesos y tiene el siguiente nombre MCSA_ANFIS, el archivo checkpoint guarda los datos de la iteración donde se quedó el entrenamiento, el archivo index guarda la arquitectura de la red y el MCSA_ANFIS guarda los pesos. Depende de cada entrenamiento el acierto final de la red, pero en este caso fue del 94%.

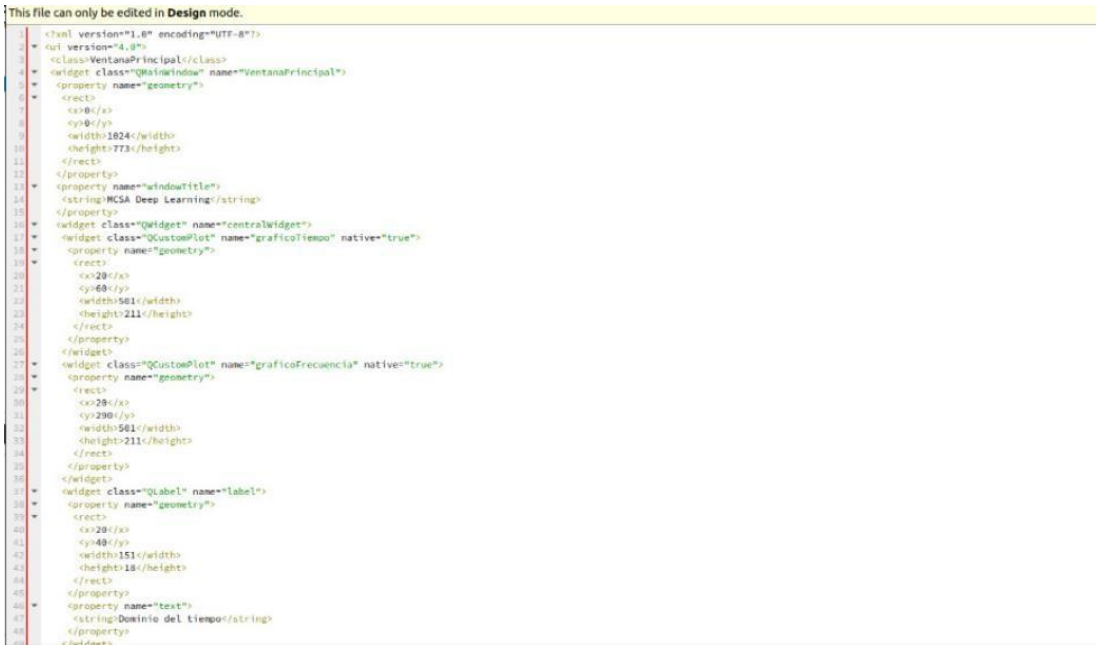


Fig. 3.35: Código de Creación de Interfaz

```
This file can only be edited in Design mode.
      <string/>
     </property>
    </widget>
   </widget>
   <widget class="QMenuBar" name="menuBar">
    <property name="geometry">
     <rect>
      <x>0</x>
      <y>0</y>
      <width>1024</width>
      <height>22</height>
     </rect>
    </property>
   </widget>
   <widget class="QToolBar" name="mainToolBar">
    <attribute name="toolBarArea">
     <enum>TopToolBarArea</enum>
    </attribute>
    <attribute name="toolBarBreak">
     <bool>false</bool>
    </attribute>
   </widget>
   <widget class="QStatusBar" name="statusBar"/>
   <widget class="QToolBar" name="toolBar">
    <property name="windowTitle">
     <string>toolBar</string>
    </property>
    <attribute name="toolBarArea">
     <enum>TopToolBarArea</enum>
    </attribute>
    <attribute name="toolBarBreak">
     <bool>false</bool>
    </attribute>
   </widget>
   <layoutdefault spacing="6" margin="11"/>
   <customwidgets>
    <customwidget>
     <class>QCustomPlot</class>
     <extends>QWidget</extends>
     <header>qcustomplot.h</header>
     <container>1</container>
    </customwidget>
   </customwidgets>
   <resources/>
   <connections/>
  </ui>
```

Fig. 3.36: Código de Creación de Interfaz

3.2.7. Evaluación del funcionamiento de la red con los datos muestra obtenidos.

Para realizar la evaluación y ejecución del programa para visualizar los diferentes tipos de errores se utilizó las muestras de entrenamiento de cada fallo y con el funcionamiento normal.

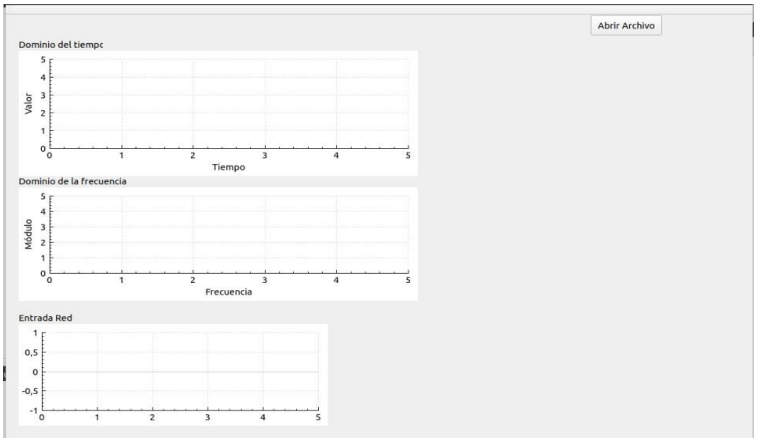


Fig. 3.37: Interfaz del proyecto

Fuente: Autor

La interfaz del proyecto es simple debido a que lo importante es la gráfica y la obtención de los resultados ya sea en cualquiera de los dos fallos o su funcionamiento

normal. Al dar clic en el botón Abrir Archivo se cargará los datos de muestra ya clasificados en base a la arquitectura que se estableció.

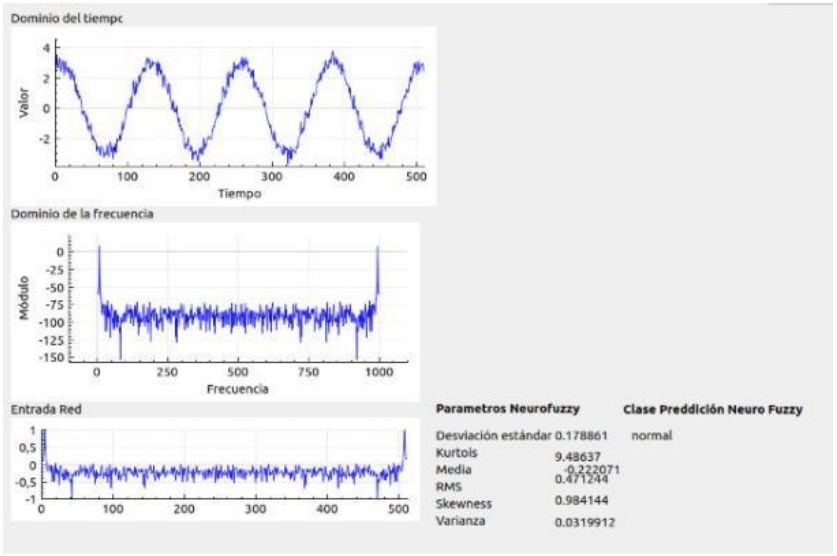


Fig. 3.38: Archivo de salida Normal

Fuente: Autor

La salida del proyecto como se ve en la Fig. 3.21 se presentan los parámetros estadísticos propuestos al desarrollar la arquitectura se denota una cierta semejanza para determinar el tipo de funcionamiento o los dos fallos, en este caso el tipo de resultado obtenido es un resultado normal, pero para los distintos fallos las gráficas son similares debido al tamaño de la gráfica a analizar.

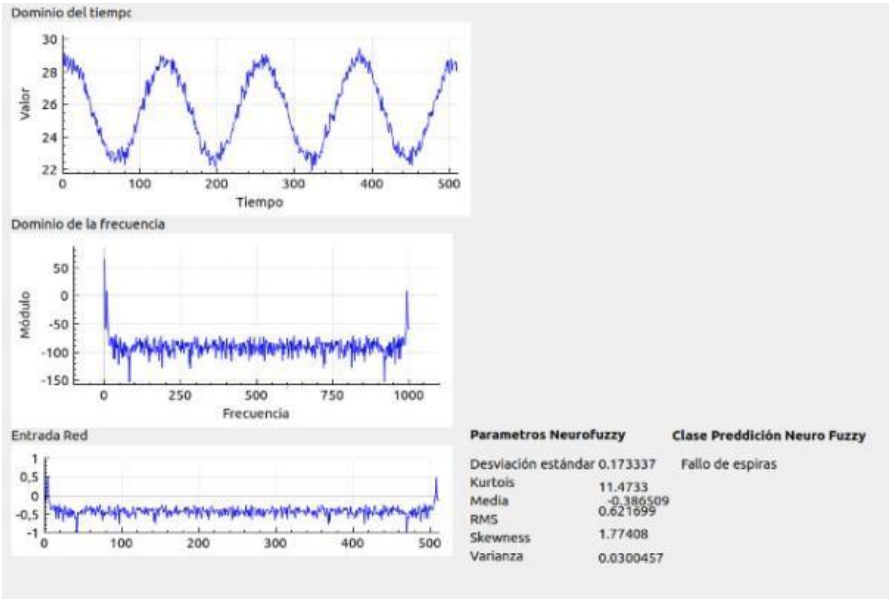


Fig. 3.39: Archivo de salida Asimetría de Eje

Fuente: Autor

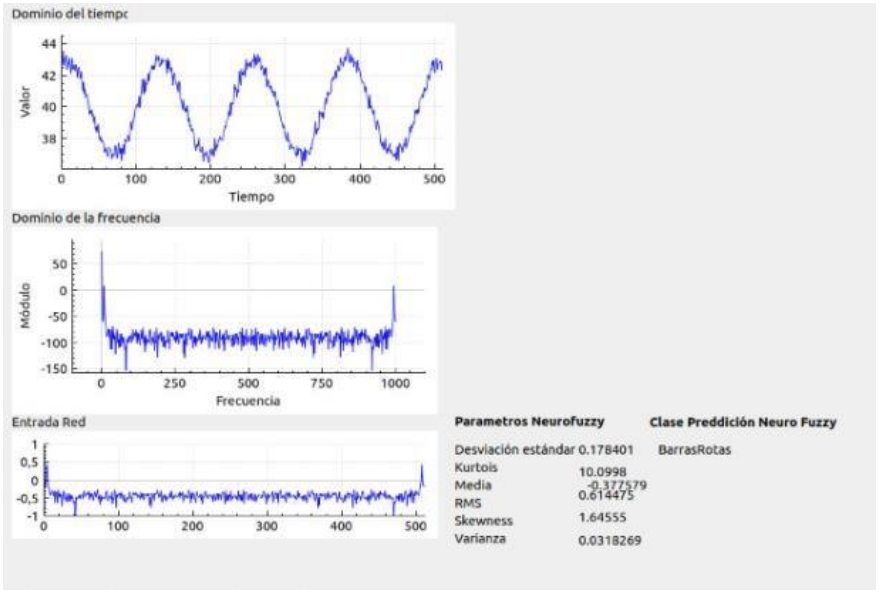


Fig. 3.40: Archivo de salida Barras Rotas

Fuente: Autor

Para graficar la evolución de los datos durante el entrenamiento. Se incluyó una herramienta que permite obtener la gráfica, para obtener la gráfica se requiere de la ejecución del siguiente código:

tools/plot_training_log.py.example 2 netloss.png resultadoentrenamiento.log

Para obtener la gráfica de la función Loss en base a las iteraciones. Para la obtención de la gráfica de la precisión en base a las iteraciones el parámetro ingresado debe ser 0, como resultado se obtienen los siguientes resultados.

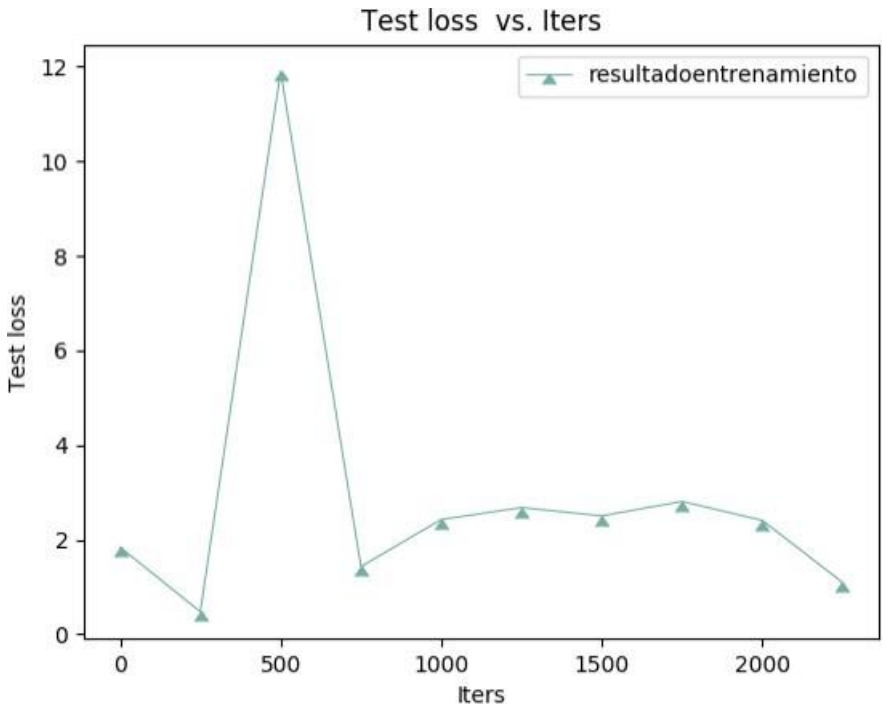


Fig. 3.41 Función Loos en base a las iteraciones

Fuente: Elaborado por el autor

La gráfica de la función Loss, permite analizar la diferencia entre el resultado calculado por la red y el resultado esperado. Esta función es aplicada en cada iteración de los dataset de entrenamiento, de esta manera se ajustan los pesos para conseguir que la perdida calculada por la función sea lo menor posible. Con los entrenamientos realizados se consiguió una gran reducción de perdida la cual a mayor entrenamiento será menor.

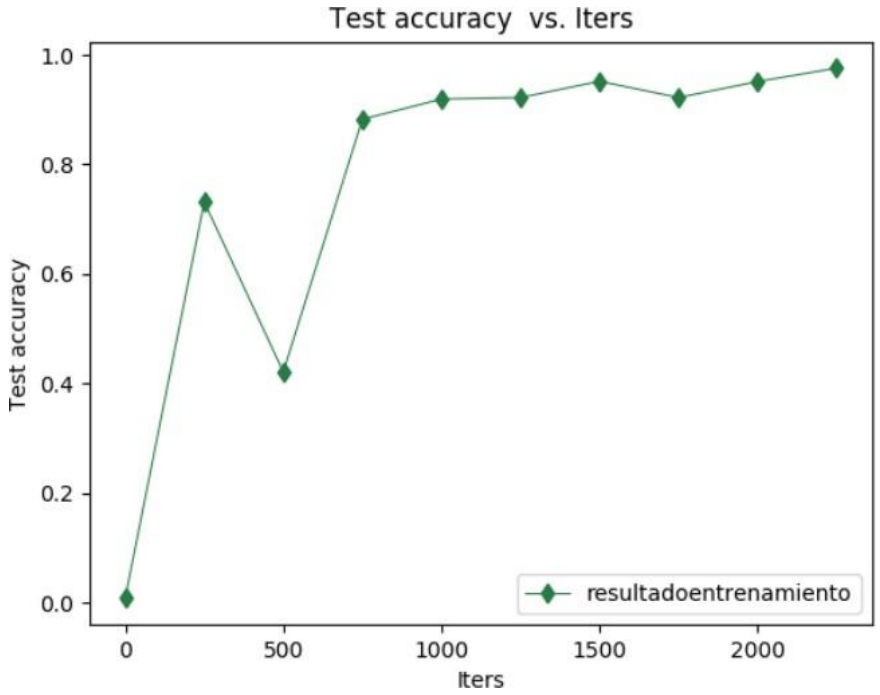


Fig. 3.42 Precisión en base a las iteraciones
Fuente: Elaborado por el autor

La gráfica demuestra la precisión que obtuvo la red con el entrenamiento. Con cada iteración de entrenamiento se obtiene una mejor precisión, debido a que el mapa de características se va incrementando lo cual facilita la clasificación, esto demuestra que a mayor entrenamiento la precisión incrementa de igual manera.

La precisión alcanzada por la red neuronal difusa fue del 94%, esto se consiguió con 150 iteraciones.

3.2.8 Pruebas

Pruebas de aceptación

Prueba de aceptación	
Numero: 1	**Historia de Usuario:** Dataset Entrenamiento
Nombre: Desarrollo de Dataset para entrenamiento	
Descripción: Al utilizar una red neuronal difusa se requiere una gran cantidad de muestras que nos permitan llegar a un resultado, por lo que necesita desarrollar un dataset de entrenamiento de datos para su correcto funcionamiento y para los dos tipos de fallos.	
Condiciones de Ejecución: Debe ser ejecutado por el investigador	
Entrada: -	
Resultado esperado: Al utilizar una red neuronal difusa y un modelo ANFIS es necesario implementar algunas librerías desarrolladas en Python que nos permita procesar los datos y clasificarlos de acuerdo con los tipos de fallos que se producen.	
Evaluación de la Prueba: Prueba Satisfactoria	

Tabla 3.31: Prueba de Aceptación 1 – Historia 1
Fuente: Autor

Prueba de aceptación	
Numero: 2	**Historia de Usuario:** Generación de muestras
Nombre: Desarrollo de generación de muestras.	
Descripción: Mediante el modelo de dataset implantado dentro de nuestra red se necesita una gran cantidad de muestras para poder procesarlas, debido a esto se requiere un pequeño algoritmo que logre clasificar las muestras en los distintos tipos de fallos para lograr obtener muestras similares a las reales.	
Condiciones de Ejecución: Desarrollar el aplicativo para desarrollar muestras.	
Entrada: Se necesita conocer la ruta en donde se alojan las muestras para poder procesarlas.	
Resultado esperado: Se obtiene las muestras deseadas para el correcto funcionamiento de nuestro proyecto.	
Evaluación de la Prueba: Prueba Satisfactoria.	

Tabla 3.32: Prueba de Aceptación 2 – Historia 2
Fuente: Autor

Prueba de aceptación	
Numero: 3	**Historia de Usuario:** Arquitectura a utilizar
Nombre: Investigación de una arquitectura a utilizar.	
Descripción: Se necesita una arquitectura de red que contemple el modelo ANFIS para poder realizar un proceso ágil y efectivo a la hora de analizar los datos y clasificarlos, al no contar con una estructura definida se utiliza el TensorAnfis.	
Condiciones de Ejecución: Debe ser ejecutado por el investigador	
Entrada:	
Resultado esperado: Al no contar con una arquitectura Definida se utiliza diferentes librerías de Python y sobre todo el TensorAnfis que nos ayudara a agilitar el proceso, el tensor se basa en investigaciones realizadas con anterioridad.	
Evaluación de la Prueba: Prueba Satisfactoria	

Tabla 3.33: Prueba de Aceptación 3 – Historia 3
Fuente: Autor

Prueba de aceptación	
Numero: 4	**Historia de Usuario:** Desarrollo de la red neuronal difusa
Nombre: Desarrollo de la red neuronal difusa	
Descripción: En base a la arquitectura, a los parámetros especificados, al modelo ANFIS se desarrollará la red que permita clasificar los datos de manera óptima y acelerar el procesamiento sin consumir muchos recursos, logrando los resultados deseados.	
Condiciones de Ejecución: Debe ejecutarse al desarrollar la red neuronal difusa.	
Entrada: Ingreso de una muestra.	
Resultado esperado: Con el desarrollo de la arquitectura en base al modelo ANFIS se logra desarrollar la red neuronal difusa iniciando por la carga de la muestra, su procesamiento y posterior clasificación, presentando el resultado deseado ya sea en funcionamiento normal o los fallos propuestos.	
Evaluación de la Prueba: Prueba Satisfactoria	

Tabla 3.34: Prueba de Aceptación 4 – Historia 4
Fuente: Autor

Prueba de aceptación	
Numero: 5	**Historia de Usuario:** Desarrollo de la Interfaz
Nombre: Desarrollo de una Interfaz	
Descripción: Se requiere desarrollar una interfaz amigable en donde se pueda seleccionar la muestra a analizar mediante un botón, además de visualizar los gráficos de la muestra, los parámetros estadísticos y el resultado deseado, ya sea su funcionamiento normal o los tipos de fallos propuestos.	
Condiciones de Ejecución: Contar con la Interfaz de Usuario.	
Entrada: Abrir el proyecto y seleccionar la muestra mediante el botón.	
Resultado esperado: Después haber culminado el diseño de la Interfaz se procederá con la selección de la muestra, posteriormente se visualizará los gráficos y se presentará los resultados en base a la muestra seleccionada.	
Evaluación de la Prueba: Prueba Satisfactoria.	

Tabla 3.35: Prueba de Aceptación 5 – Historia 5
Fuente: Autor

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se investigó el funcionamiento de una red neuronal difusa mediante ANFIS que permiten identificar los factores, y parámetros estadísticos que intervienen para su desarrollo.
- Se identificó los fallos a analizar en motores trifásicos de corriente alterna, de acuerdo a la matriz de datos se identificó los parámetros que influyen en este tipo de fallos.
- Se desarrolló una red neuronal difusa capaz de aprender e identificar las características en funcionamiento normal y los fallos propuestos.
- Se implementó la red neuronal difusa con las pruebas de funcionamiento que permiten identificar los fallos que pueden presentar los motores trifásicos de corriente altera.

4.2 Recomendaciones

- En el desarrollo de la red neuronal difusa se recomienda analizar las arquitecturas utilizadas en trabajos de investigación certificados, que permitan la implementación de la lógica difusa en redes neuronales artificiales, logrando así reducir tiempos de procesamiento, agilidad en la clasificación de muestras y sobre todo la obtención de un resultado confiable.
- La maquinaria industrial es una parte fundamental dentro de la industria, razón por la cual se debe tener un monitoreo constante de estas, para evitar todo tipo de fallo, avería ocasionando grandes pérdidas económicas y de producción.
- Se recomienda el uso de este tipo de sistemas que ayuden a predecir los diferentes tipos de errores que pueden presentarse en motores, evitando así, pérdidas de producción, daños drásticos de partes y piezas, permitiendo planificar mantenimientos preventivos.
- Se recomienda añadir ruido blanco a las gráficas de las muestras para que estas sean más claras al momento de presentar resultados, obteniendo una clasificación de las muestras a procesar para optimizar y agilizar su procesamiento.
- Es recomendable utilizar datos de prueba para el entrenamiento de la red, al igual que una gran cantidad de datos que permitan procesados mediante algoritmos que se asemejen a los datos reales.

CAPITULO V
MATERIALES DE REFERENCIA

BIBLIOGRAFIA

[1] J. A. Zapata Sinaluisa, “Diseño y simulación de un controlador neuro-difuso ANFIS para un convertidor dc-dc Zeta.” SANGOLQUI, p. 158, 2018.

[2] C. J. Celi Sánchez, “Diseño, desarrollo e implementación de un sistema adaptativo neuro-difuso aplicado al brazo robot Mitsubishi RV-2AJ con visión artificial, utilizando un controlador basado en procesador ARM.” p. 100, 2017.

[3] J. A. Mina Beltran and W. A. Diaz Gonzalez, “Prototipo de un Sistema Web inteligente basado en redes neuronales difusas para medir el estado cognitivo del Estudiante en la asignatura Simulación de Sistemas por parte del Docente de la Cátedra.” p. 131, 2019.

[4] J. E. Monzón and M. I. Pisarello, “Identificación de Latidos Cardíacos Anómalos con Redes Neuronales Difusas.” p. 4, 2015.

[5] E. V. Alvarado Guichay, “Algoritmo neuro-difuso para la detección y clasificación de fallas en líneas de transmisión eléctrica del sistema ecuatoriano usando simulaciones y datos de registradores de fallas.” p. 148, 2015.

[6] M. A. Ma. Belén, “Mantenimiento Industrial,” vol. 1, pp. 3–47, 2015.

[7] H. Niwa, “Técnicas De Mantenimiento Predictivo. Metodologia De Aplicación En Las Organizaciones,” *Development*, vol. 134, no. 4, pp. 635–646, 2017.

[8] J. J. Saucedo Dorantes, “Metodologías para mejorar la confiabilidad del diagnóstico de fallas en cadenas cinemáticas basado en algoritmos inteligentes y fusión de datos,” p. 110, 2017.

[9] A. J. Besa Gonzálvez and J. Carballeira Morado, “Diagnóstico y corrección de fallos de componentes mecánicos (2a. ed.),” p. 270, 2018, [Online]. Available: http://ebookcentral.proquest.com/lib/utnortesp/detail.action?docID=5426104.

[10] H. A. Banda Gamboa, *Inteligencia Artificial: Principios y Aplicaciones*, vol. I, no. May. Quito, 2014.

[11] C. Lepage Chumpitza, "Aplicaciones actuales de la inteligencia artificial y su uso con la tecnología IBM Watson." p. 54, 2016.

[12] A. F. Morocho Caiza, "Desarrollo de un Regulador Neurodifuso Adaptativo para el Control de una Planta de Vuelo Vertical de National Instrument," p. 83, 2018.

[13] L. Pujol, P. Arevalo, and E. Ortiz, "La modelación con Redes Neuronales para la previsión de caudales en cuencas del ámbito Mediterráneo haciendo uso de los datos SAIH. Aplicación a cuencas del Júcar y el Segura," *Hidrol. y Gestión del Agua 4*, p. 10, 2015.

[14] J. D. Martin, "Implementación de Redes Neuro-Difusas Para Per Aplicadas en Problemas de," p. 106.

[15] C. García García and N. J. Meléndez Acosta, "Clasificación de género utilizando medidas antropométricas, análisis de texturas y la arquitectura ANFIS," *Res. Comput. Sci.*, vol. 136, no. 1, pp. 33–46, 2017, doi: 10.13053/rcs-136-1-3.

[16] V. Vaidhehi, "The role of Dataset in training ANFIS System for Course Advisor," vol. 1, no. 6, pp. 249–253, 2014.

[17] C. F. N. José Antonio Aquino Robles, Leonel Corona Ramírez, "MODELADO Y SIMULACIÓN DE LA OPERACIÓN DEL MOTOR SERIE CON DIFERENTES TENSIONES DE MODELING AND SIMULATION OF SERIES MOTOR OPERATION Resumen," vol. 41, no. 134, pp. 861–876, 2019.

[18] H. V. Cevallos, "Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software," 2018.

[19] J. C. Salazar *et al.*, "Scrum versus XP: similitudes y diferencias Scrum vs XP: Similarities and Differences," 2018.

[20] S. Farokhzad, “Detección de fallas basada en la vibración de la bomba centrífuga mediante transformada rápida de Fourier y adaptativa Sistema de inferencia neuro-difusa,” 2017.