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 CONVOLUCIONAL PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL

Trabajo de Graduación. 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: Jonathan Javier Shulca Andrade

TUTOR: Ing. Mg. Edison Homero Álvarez Mayorga

Ambato – Ecuador

Agosto 2020

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: **"DESARROLLO DE UNA RED NEURONAL CONVOLUCIONAL PARA LA DETECCIÓN DE FALLOS EN MAQUINARIA ROTATIVA EN TIEMPO REAL",** desarrollado bajo la modalidad Proyecto de Investigación por el señor Jonathan Javier Shulca Andrade, 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, agosto 2020

EL TUTOR

Ing. Mg. Edison Homero Álvarez Mayorga

AUTORÍA

El presente proyecto de Investigación titulado: "DESARROLLO DE UNA RED NEURONAL CONVOLUCIONAL 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, agosto 2020

Jonathan Javier Shulca Andrade

C.C: 1803748647
AUTOR

APROBACIÓN TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Jonathan Javier Shulca Andrade, 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 CONVOLUCIONAL 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, agosto 2020

Ing. Elsa Pilar Urrutia Urrutia, Mg.

PRESIDENTA DELTRIBUNAL

Ing. Oscar Fernando Ibarra Torres

PROFESOR CALIFICADOR

Ing. Hernán Fabricio Naranjo Ávalos

PROFESOR 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 procesos 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, agosto de 2020

Jonathan Javier Shulca Andrade

C.C: 1803748647
AUTOR

DEDICATORIA

Este trabajo está dedicado principalmente a Dios por guiarme en cada paso que he dado en mi vida y por permitirme llegar hasta este punto en mi formación académica, a mis padres y hermanos los cuales han sido mi apoyo a lo largo de toda mi vida dándome todo su apoyo en los momentos difíciles, su amor y fortaleza para cumplir las metas que me he propuesto.

Jonathan Javier Shulca Andrade

AGRADECIMIENTO

Agradezco primeramente a Dios por permitirme terminar el camino que inicie de manera satisfactoria, a todos los docentes de la FISEI por sus enseñanzas a lo largo de toda mi carrera universitaria.

A mis compañeros de clase por compartir sus conocimientos los cuales me ayudaron a adquirir experiencia y un sinfín de enseñanzas.

Al Ing. Edison Álvarez por la paciencia brindada a lo largo del desarrollo de la presente investigación.

Jonathan Javier Shulca Andrade

ÍNDICE GENERAL DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

La importancia de esta investigación radica en que se trata de un tema que está en auge y con el cual se puede prever de soluciones a la comunidad productiva. La presente investigación pretende ser un aporte significativo a las compañías poseedoras de maquinarias con motores trifásicos de corriente alterna, debido a que brindará seguridad a los operadores sobre un funcionamiento de maquinaria libre de fallos.

El presente proyecto pretende ser de utilidad práctica en las áreas de Ingeniería en Sistemas e Ingeniería Industrial debido a que por medio de la utilización de herramientas informáticas se puede obtener información sobre el estado de los motores trifásicos de corriente alterna en tiempo real, mediante lo cual los operadores pueden anticipar fallos que se pueden presentar en motores trifásicos de corriente alterna y adelantarse a sus soluciones, lo cual reducirá las pérdidas ocasionadas por el paro imprevisto de las máquinas.

La solución propuesta está basada en la línea de investigación que trata la Inteligencia Artificial, la cual se basa en desarrollar sistemas inteligentes, en este caso por medio de la utilización de una red neuronal convolucional, los fallos en motores trifásicos serán detectados en sus primeras etapas para aplicar correcciones inmediatas.

Con este proyecto los principales beneficiarios serán las empresas que en su producción utilicen maquinarias rotativas, debido a que la producción nunca se vería detenida si los fallos fueran previstos.

Palabras clave: Redes neuronales, Inteligencia Artificial, convolución, fallos, motores de inducción.

ABSTRACT

The importance of this research lies in the that it is a topic that is booming and with which solutions can be foreseen to the productive community. This research is intended to be a significant contribution to companies possessing machinery with three-phase alternating current motors, because it will provide operators with safety on fault-free machinery operation.

This project is intended to be of practical use in the areas of Engineering in Systems and Industrial Engineering because through the use of computer tools information can be obtained on the status of three-phase ac motors in real time, whereby operators can anticipate failures that can occur in three-phase alternating current motors and anticipate their solutions , which will reduce losses caused by unforeseen machine shutdown.

The proposed solution is based on the line of research that deals with Artificial Intelligence, which is based on developing intelligent systems, in this case through the use of a convolutional neural network, failures in three-phase motors will be detected in their early stages to apply immediate corrections.

With this project, the main beneficiaries will be the companies that use rotating machinery in their production, because production would never be stopped if the failures were foreseen.

Descriptors: Neural Networks, Artificial Intelligence, Convolution, Faults, Induction Engines.

CAPÍTULO I

MARCO TEÓRICO

1.1 Antecedentes Investigativos

Después de una investigación en el repositorio de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, se encontró el trabajo desarrollado en el año 2018 titulado: "Control de la gesticulación de un robot social interactivo con aspecto humanoide". El cual fue elaborado por el señor Edisson Mauricio Arias Infante, en el cual se relata lo siguiente:

"…El control gesticular se basa en la identificación de los gestos faciales que genera el usuario, datos que son recopilados por medio de la visión artificial, luego estos datos son procesados por el submódulo de tratamiento de imágenes, posteriormente son enviados a un modelo de red neuronal convolucional, empleadas como clasificadores de patrones en el aprendizaje profundo (Deep Learning), para finalmente seleccionar y enviar la sentencia de movimiento al robot."[1].

Con este trabajo se puede entender la utilidad de las redes neuronales convolucionales aplicadas en un medio práctico, el cual en este caso fue identificar rasgos faciales. Aquí se puede comprender los beneficios que conlleva el utilizar este tipo de redes para el tratamiento e identificación de información en tiempo real.

En el artículo científico publicado el año 2017 bajo el título "Reconocimiento de imágenes en frames de video utilizando redes neuronales" desarrollado por el ingeniero Andrés Amores se manifiesta que "…En la actualidad existen varias aplicaciones para realizar el reconocimiento y clasificación de imágenes en tiempo real, para esto se requiere procesar el video de manera rápida. Uno de los métodos que se utiliza para realizar estas aplicaciones es el uso de redes neuronales. En el presente proyecto de investigación se realizó un clasificador de imágenes con una red neuronal convolucional (CNN) el cual se implementada para entrenar al clasificador cuyas imágenes a ser reconocidas son una secuencia de cuadros de video obtenido a través de una cámara…"[2].

Con este artículo se puede evidenciar la factibilidad de utilizar las redes neuronales convolucionales para el análisis de datos en tiempo real, cosa que produciría mejoras en el área en que sea aplicada.

Otro artículo científico de interés que fue publicado en el año 2018 bajo el título: "Desarrollo de un sistema de visión artificial para detectar automóviles estacionados en lugares no permitidos" el ingeniero Carlos Arteaga manifiesta que "…Las redes neuronales convolucionales previo a un entrenamiento y validación se encargan de extraer un mapa de características de las imágenes de entrada, para posteriormente a modo de mapa ser ingresado a la máquina de soporte vectorial SVM el cual se encargara de la clasificación, este funcionamiento es creado por un proceso previo de entrenamiento y validación, el cual genera una salida que determina si existe o no un automóvil…"[3].

Tomando como referencia este artículo se consiguió definir una metodología adecuada para agilitar el procesamiento de la red, la metodología sería dividir los datos por grupos para que sea más fácil su identificación.

Contextualización del problema

A nivel mundial el uso de redes neuronales convolucionales ha estado presente ya hace algunos años atrás, el mayor ejemplo de esta aplicación ha sido GOOGLE quien a finales del 2015 libero TensorFlow, Sundar Pichai, CEO de Alphanet menciona que "El uso de TensorFlow actualmente es para todo, desde el reconocimiento de voz en la aplicación de GOOGLE, hasta SmartReply en inbox y la búsqueda en Google Fotos". Entre las principales características que denotan de este recurso se encuentran la capacidad para generar etiquetas, categorías y reconocer textos en Google Imágenes. TensorFlow es la librería de redes neuronales libre más avanzada que existe, y ha sido desarrollada por una división llamada Google Brain. Destaca especialmente por su capacidad para generar sistemas de redes neuronales convolucionales, capaces de realizar tareas avanzadas de visión artificial, desde el reconocimiento, etiquetado y clasificación de imágenes a sistemas enormemente precisos de reconocimiento de escritura[4].

A nivel nacional las redes neuronales convolucionales todavía no tienen un uso considerable, pero uno de los casos existentes es el propuesto y desarrollado por la

Universidad Politécnica Salesiana de Cuenca, quienes como proyecto desarrollaron un sistema para el monitoreo y detección de incendios a través de un vehículo no tripulado. Este proyecto fue enfocado en el procesamiento de imágenes, es por esto que usaron redes neuronales convolucionales para determinar orígenes de incendios en base a imágenes tomadas a través de un dron[5].

Referente a la provincia de Tungurahua cantón Ambato, teniendo en cuenta los problemas que presentan las empresas poseedoras de maquinarias con motores trifásicos debido a defectos imprevistos, los cuales producen pérdidas debido al paro en la producción, se determinó que una red neuronal convolucional es la mejor opción para poder prevenir fallos en maquinaria rotativa ya que con su uso se puede anticipar los posibles defectos que se pueden ocasionar con el uso, haciendo así que el tiempo de reparación se vea reducido lo que evitará pérdidas en la producción de la empresa.

Fundamentación Teórica

Historia del mantenimiento

En la primera generación de la industrialización, cuando se empezaron a utilizar máquinas para las etapas de producción, las empresas poseían departamentos que se encargaban de los mantenimientos los cuales se dedicaban a esperar que las máquinas sufrieran de un desperfecto antes intervenirlas y realizar las respectivas reparaciones y así resolver los fallos, en la época que se empieza a conocer el concepto de fiabilidad se empieza a implementar técnicas para prevenir fallos. En la segunda generación se comenzó a implementar estrategias de mantenimiento preventivo iniciando con la capacitación del personal sobre técnicas para estudiar las tareas de mantenimiento más adecuadas que hay que realizar para evitar fallos. Dentro de la tercera generación se introduce el mantenimiento predictivo y la implementación de software para el monitoreo de fallos basando sus objetivos en los del departamento de mantenimiento. En la generación actual se implementa sistemas de mejora continua de los planes de mantenimiento preventivo, y en si la organización y ejecución del mantenimiento. Además, se establecen grupos de mejora y seguimiento en las acciones[6].

Objetivos de mantenimiento

Entre los principales objetivos de un mantenimiento está el de aumentar la disponibilidad de los equipos, reducir los costes al mínimo procurando evitar paros en

la producción, mejorar la fiabilidad de máquinas e instalaciones para evitar retrasos graves en las entregas y brindar asistencia al departamento de ingeniería en los nuevos proyectos para facilitar el mantenimiento de las nuevas instalaciones[7].

La gestión de mantenimiento debe tener en cuenta que hace parte de la realización de cada uno de estos objetivos, razón por la cual es fundamental trabajar de forma proactiva con cada uno de los departamentos que hacen parte de la compañía[7].

Para llevar a cabo un mantenimiento se requiere desarrollar un cronograma del mismo con anterioridad para no detener el normal funcionamiento de la compañía, de esta manera se consigue optimizar los recursos sin ocasionar paros en la producción y evitar posibles fallos en el futuro[7].

Mantenimiento predictivo

Este mantenimiento está basado en la predicción de errores antes de que se produzca y pueda provocar malestares. Se trata de prever defectos que se podrían presentar para así adelantarse a su posible solución o el momento en el cual un equipo dejará de trabajar en sus condiciones óptimas. Para conseguir prever fallos se diagnostica los equipos para así analizar parámetros como pueden ser de vibración, radiación infrarroja, tensiones y corrientes de alimentación, los cuales son emitidos por las máquinas sin que estas dejen de trabajar[7].

En general, el mantenimiento predictivo, determina el periodo de tiempo en el que el fallo va a tomar una relevante importancia, para poder planificar todas las intervenciones con tiempo suficiente, para que no llegue a tener consecuencias graves[7].

El mantenimiento predictivo consiste en el análisis del ambiente de trabajo para determinar el número de equipos, estudiar sus características fundamentales y sus modos potenciales de fallo. Una vez obtenido este análisis se procede a normalizar, es decir traducir los modos de fallos a parámetros predictivos y asignarles los límites de aceptación. Seguido de esto viene la sistematización lo cual es establecer las nuevas normativas para el comportamiento de la organización en la eventualidad de que se presente un fallo, es decir los pasos a seguir en caso de que se presente los cuales son confirmación del diagnóstico, evaluación, acción[7].

Este tipo de mantenimiento posee varias ventajas, entre las cuales están el reducir el tiempo de parada de las maquinarias debido al anticipado conocimiento sobre un fallo, historial de registro tanto de fallos como de reparaciones para optimización de los recursos humanos, lo cual permitirá conocer con exactitud el tiempo límite de actuación que no implique que el fallo se torne grave.

Las desventajas notorias para este tipo de mantenimiento es que la implantación de un sistema de este tipo requiere una inversión inicial muy alta, debido a que los equipos para detección de fallos tienen un costo elevado en el mercado, también se requiere destinar un personal capacitado para la lectura de los datos recolectados por los equipos utilizados.

Técnicas aplicadas en el mantenimiento predictivo

Análisis de vibraciones: se presentan en los mantenimientos del ámbito industrial tanto preventivo como predictivo, con el interés de alertar las consecuencias que conlleva un elemento vibrante en una máquina, así como la necesidad de prevenir las fallas que traen las vibraciones a medio plazo[8].

Análisis de lubricantes: estos se ejecutan dependiendo de la necesidad pueden ser tanto iniciales como rutinarios. Los iniciales son los que se dan a notar con irregularidades en un estudio de lubricación, las cuales se pueden corregir con cambios en las condiciones de operación, por otra parte, el análisis rutinario es aplicado en equipos de operatividad en la cual se realiza un estudio de la lubricación para tener una referencia de las condiciones del lubricante y de esta manera tener la seguridad de que la máquina se encuentra en óptimas condiciones de funcionamiento o si requiere un cambio[8].

Análisis por ultrasonido: es el cual analiza las ondas de sonido de baja frecuencia producidas por los equipos que no son perceptibles por el oído humano; permite detectar fricción en máquinas rotativas, fugas en válvulas, fugas en fluidos, perdidas de vacío y detección de arco eléctrico[8].

Análisis por termografía: es una técnica que permite medir y visualizar las temperaturas de superficies con precisión.

Análisis de firmas de corriente de un motor (MCSA), es un método que permite evaluar la condición del rotor en motores de inducción, se basa en capturar la forma de onda de la corriente de entrada del motor en funcionamiento, analizándola con la transformada rápida de Fourier y evaluando las amplitudes de los armónicos los cuales se amplifican ante la presencia de un fallo en el[8].

"…Según MCSA, el espectro de frecuencia de la corriente del motor se escanea con el objetivo de detectar componentes específicos vinculados a fallos. Basándose en la amplitud relativa de estos componentes, se puede inferir un defecto del espectro actual, se puede entonces clasificar a estos errores como de naturaleza eléctrica o mecánica con el siguiente principio…"[9].

Se puede conocer la naturaleza del fallo, analizando los componentes de la señal obtenida, para el caso de un defecto eléctrico, se presenta cuando los componentes de la amplitud de voltaje se ven reflejados en el espectro de voltaje, en el caso de un error de naturaleza mecánica los componentes no se ven reflejados en la señal de voltaje solo se presentan en el espectro de corriente[9].

Fallos en maquinarias rotativas

Los fallos se pueden provocar por diversas formas desde un mal diseño o error de cálculo, defectos de fábrica así como el mal uso de las instalaciones, máquinas o equipos, el desgaste natural por el uso, fenómenos naturales y otras causas[6].

Principales fallas en motores de inducción

Falla ocasionada por barras rotas (BRB). En su mayoría este tipo de fallos inicia con fisuras en una barra del rotor o en el anillo final los cuales pueden tornarse en fallos graves que pueden provocar cortes o rupturas en las barras[9]. Una vez producido un agrietamiento da inicio un ciclo degenerativo hasta que se produce una ruptura, cuando una barra se rompe sus vecinas empiezan a soportar más presión y corriente, debido a esto realizan mayor esfuerzo tanto térmico como mecánico lo que ocasionará que estas también empiecen un proceso de agrietamiento hasta que finalmente se produce una ruptura[10].

Falla ocasionada por excentricidad. Este fallo se ocasiona cuando no existe una perfecta alineación entre los centros del rotor y del estator, las principales causas de

este fallo son desgastes de cojinetes, torceduras del eje e irregularidades del rotor. Por otro lado las consecuencias que se provocan son vibraciones, así como aparición de armónicos de alta y baja en la frecuencia de corriente de un estator[10].

Inteligencia Artificial

La inteligencia artificial es un área de investigación la cual estudia el proceso de como aprende el cerebro humano con el propósito de recrearlo por medio de la utilización de programas de computación o prototipos no humanos de inteligencia, capaces de realizar acciones propias del hombre tales como: el autoaprendizaje, los gestos, el habla, la toma de decisiones y resoluciones de problemas de la vida cotidiana[11].

A la inteligencia artificial se le puede definir como aquella inteligencia presentada por artefactos construidos por humanos, por lo cual se puede asumir que un sistema posee inteligencia artificial cuando es capaz de llevar a cabo tareas que son usualmente realizadas por un humano. Dentro de las ciencias de la computación, la rama de la IA se basa en intentar recrear un comportamiento inteligente, similar al humano y al funcionamiento de las aplicaciones informáticas para el proceso de toma de decisiones[11].

Aprendizaje profundo

El aprendizaje profundo, es una técnica de aprendizaje automático implementada en ordenadores para asemejar lo que resulta natural para las personas y esto es aprender mediante ejemplos[12].

Con el aprendizaje profundo, un modelo informático puede aprender a realizar tareas de clasificación directamente a partir de imágenes, textos o sonido como lo haría una persona, pero de manera más rápida. Los modelos de aprendizaje profundo pueden obtener una gran precisión que, en ocasiones, supera el rendimiento humano. Los modelos se entrenan mediante un amplio conjunto de datos etiquetados los cuales se obtienen al ser recopilados usando herramientas de medición [12].

En la actualidad se ha incrementado el uso de redes neuronales artificiales en el desarrollo de algoritmos por parte de los desarrolladores, todo esto con el propósito de asimilar la estructura y función del cerebro. Teniendo en cuenta esto se diseñan módulos interconectados los cuales operan modelos matemáticos que son

continuamente afinados en base a los resultados obtenidos después del análisis de grandes cantidades de entradas, lo cual ayuda a mejorar el procesamiento. El aprendizaje profundo puede ser supervisado el cual requiere de intervención humana para entrenar la capacidad de los modelos o no supervisado el cual es autónomamente debido a que refina los modelos en base a una autoevaluación[13].

Enfoques de la Inteligencia Artificial

Algunos de los enfoques de la inteligencia artificial son los siguientes:

Inteligencia artificial fuerte: Es la que se enfoca en la programación de máquinas para que sean capaces de mantener estados cognitivos mentales como son el razonar y resolver problemas[14].

Inteligencia artificial débil: Se enfoca en las máquinas que son incapaces de realizar y resolver problemas, pero que programadas de manera correcta pueden simular la conciencia humana[14].

Inteligencia artificial aplicada: Se enfoca en el desarrollo de todo tipo de aplicaciones utilizadas en el ámbito comercial, en la actualidad existen varias aplicaciones exitosas y muy utilizadas[14].

Inteligencia artificial cognitiva: Se enfoca en desarrollar máquinas que estudien el comportamiento cognitivo del cerebro humano así como el estudio de teorías sobre el reconocimiento los objetos en el ámbito de la robótica, o la solución a los problemas abstractos[14].

Aplicaciones de la Inteligencia artificial

La inteligencia artificial se divide en varias ramas de aplicaciones entre los cuales se encuentran, la robótica, el análisis de imágenes o el tratamiento automático de textos. Con respecto a la robótica, una de las características más interesantes en las investigaciones es el aprendizaje adaptativo, en el cuál un sistema robotizado analiza las diferentes posibilidades con el objetivo de realizar movimientos ideales y de mejor resultado, por ejemplo: que un robot cuadrúpedo que se desplace de manera autónoma, debe realizar un proceso de exploración y aprendizaje similar al que realiza un recién nacido[15].

Una de las ramas principales en la inteligencia artificial se la domina como sistemas expertos, los cuales tienen por objetivo desarrollar sistemas capaces de analizar una muestra de datos y realizar tareas asociadas a un perfil profesional como podrían ser el diagnóstico y detección de fallos, planificación y toma de decisiones[15].

Redes neuronales artificiales

Las redes neuronales artificiales son diseñadas en base al funcionamiento de la corteza cerebral en los mamíferos. Sin embargo, estos modelos no se comparan al funcionamiento, la escala y la complejidad del cerebro humano. Los modelos de redes neuronales artificiales se pueden comprender como un conjunto de unidades de procesamiento básico, que se encuentran interconectadas y operan en función de las entradas dadas, para procesar la información y generar las salidas deseadas en base a su configuración[16].

Una red neuronal artificial está compuesta por un conjunto de neuronas conectadas entre sí, asemejando las conexiones sinápticas de un cerebro humano, cada neurona está compuesta por: conexiones de entrada, núcleo central de proceso y una salida[16].

Las redes neuronales pueden ser agrupadas en dos categorías genéricas basadas en la forma en que la información es propagada en la red.

Redes de conexiones hacia delante (Feed-forward networks): El flujo de información se da en una sola dirección. Si la red se considera como un diagrama de flujo en el cual las neuronas actúan como nodos, las conexiones entre los nodos se dan de manera que no se generan bucles o ciclos[16].

Redes de conexiones hacia atrás (Feed-back networks): Son redes que utilizan retroalimentación es decir poseen conexiones que forman ciclos dirigidos. La arquitectura les permite operar y generar secuencias de tamaños arbitrarios. Las redes de retroalimentación demuestran capacidad de aprendizaje y relaciones de secuencia en su memoria interna[16].

Redes Neuronales Convolucionales

Las redes neuronales convolucionales o CNN por sus siglas en inglés (Convolutional Neural Network), son un tipo de red de conexión hacia delante, tiene un funcionamiento similar al de las redes neuronales estándar. La diferencia radica en que

cada unidad en una capa de CNN es un filtro de dos o más dimensiones que se convoluciona con la entrada de esa capa lo que hace esencial para casos en los que se requiere identificar patrones de entradas de alta dimensión, como se encuentran en imágenes o videos. Los filtros CNN incorporan el contexto espacial al tener una forma espacial similar, pero más pequeña que los medios de entrada, y usan el uso compartido de parámetros para reducir significativamente el número de variables aptas para el aprendizaje, es decir analizan por partes lo que incrementa la velocidad del procesamiento[17].

Estructura de una red neuronal convolucional

La estructura básica de una red convolucional está formada por las siguientes capas:

Capa convolucional: Es el núcleo de las CNN y consiste en una serie de filtros de aprendizaje, estos filtros tienen un pequeño campo receptivo, por medio del cual siguiendo el flujo de aprendizaje hacia delante cada uno de estos filtros se convoluciona con todo el campo de visión produciendo así un mapa de características, el cual será tomado para agilizar los reconocimientos posteriores[17].

Capa de submuestreo: Es una parte fundamental de la red, estas capas se aplican después de las capas convolucionales. El propósito de esta capa es reducir el tamaño de la entrada, aún cuando genera pérdidas de información. También beneficia a la red al reducir la carga de cálculo en las siguientes capas y además de reducir el sobre ajuste de la red[17].

Capa completamente conectada: Por lo general esta capa se aplica al final de las combinaciones de las capas convolucionales y de submuestreo. Cada elemento corresponde a una neurona y el número total de neuronas en estas capas corresponderá el número de clases que se desean predecir. La última se emplea en tareas de clasificación[17].

Arquitectura de una red neuronal convolucional

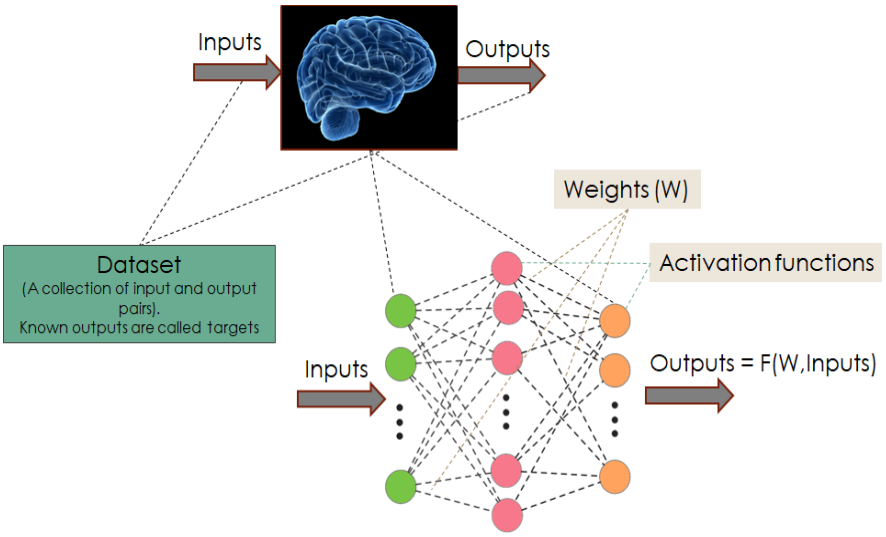

Fig. 1.1 Elementos de una CNN

Fuente: [17]

Para el desarrollo de una CNN principalmente se debe seleccionar un arquitectura. Una CNN puede poser varias arquitecturas las cuales pueden ser de utilidad para aplicaciones específicas. Para este caso un arquitectura comprobada con el tipo de análisis de MCSA sera la mas adecuada para utilizarce. [18].

El arquitectura elegida para el desarrollo de este proyecto es la del paper “Predicting detective engines using convolutional neuronal networks on temporal vibration signals”[18].

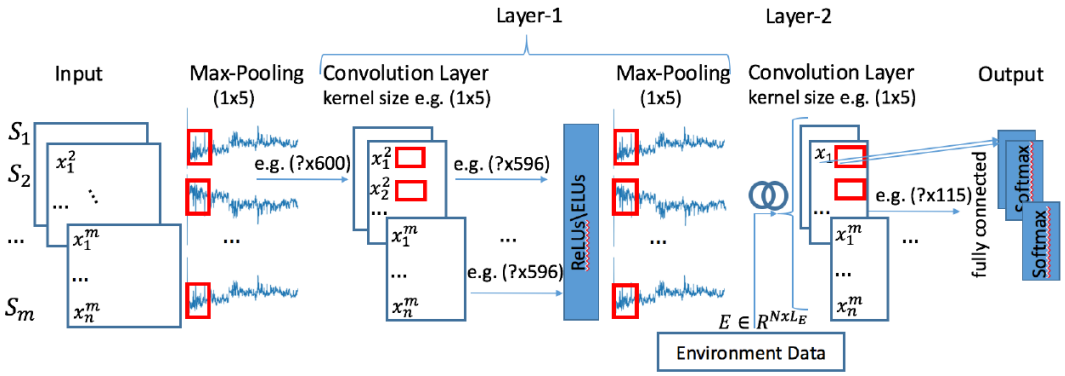

Fig. 1.2 Visión de las dos capas de arquitectura CNN

Fuente: [18]

Esta arquitectura está diseñada para facilitar el entrenamiento de las capas convolucionales, así como para mejorar la eficiencia, una vez pasada la etapa de

procesamiento de la señal se procede a reducir la muestra por medio de un max-pooling de (1x5) con esto se consigue disminuir progresivamente el tamaño espacial de las características y la complejidad de cálculo; las capas convolucionales están siendo usadas a manera de extractoras de características lineales no aprendibles en las señales temporales[18].

Una CNN puede tener varios tipos de entrada los cuales pueden ser datos en tres dimensiones, binarios o análogos; para este caso en particular se utiliza un tipo de entrada de datos de una dimensión con 512 elementos, los cuales conforman una gráfica de la transformada de Fourier (FFT) transformada a base de datos (DB). Los datos se almacenan de esta manera con el fin de poder graficarlos[18].

Las salidas en una CNN son conocidas comúnmente como clases, para este proyecto se trabajarán con tres clases, después de calcular los elementos de la red, la salida obtenida dirá si la muestra del MCSA será normal, con barras rotas o con asimetría de del eje.

Una de las partes fundamentales para el funcionamiento de una CNN es el dataset, el cual es una colección de datos debidamente etiquetados, tanto entradas, así como salidas, es decir le enseñaremos a la red a qué parámetro de salida pertenecerán los datos de entrada. La red sabrá clasificar las entradas para dar la salida correspondiente. El dataset creado para el proyecto contiene 512 elementos y la etiqueta de a qué clase corresponde, el número y calidad de muestras dentro del dataset son muy importantes debido que a mayor número de muestras la clasificación será más precisa, en las CNN se usan dataset con un número de muestras alrededor de 10000.

El entrenamiento de la red es fundamental para su uso, debido a que con el entrenamiento se encuentran los pesos de cada neurona los cuales servirán para poder clasificar de manera correcta las entradas de la red, dependiendo del volumen del dataset el entrenamiento puede tomar desde minutos hasta días.

Generación de dataset

La parte más importante del proyecto es el dataset, basado en su cantidad y calidad el proyecto tendrá éxito; el dataset será creado utilizando la librería caffe por lo cuál se deberán cumplir con ciertas características.

Las capas de las librerías de caffe están definidas como blobs, las cuales son contenedores de datos reales que caffe puede procesar. También proporcionan la capacidad de sincronización entre la CPU y GPU. Matemáticamente hablando, es una matriz de dimensión N que se encuentra almacenada de forma c-contigua. Los detalles que debe poseer un blob son como se almacena y comunica la información ya sea tanto en las capas, como entre las capas y redes, en el proyecto cada capa de entrada de la red aceptará los datos de la onda por esta razón el blob debe seguir los lineamientos de programación de la librería con un orden de definición en la capa de entrada[19].

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

$$N: Representa\ el\ número\ de\ muestras$$
$$K: Representa\ el\ número\ de\ canales\ de\ muestra$$
$$H: Representa\ la\ altura\ del\ dato\ de\ entrada$$
$$W: Representa\ el\ ancho\ del\ dato\ de\ entrada$$

Ecuación 1 Estructura de un Blob

Fuente: [20]

Teniendo en cuenta el orden de definición de los blobs se distingue la singularidad de esta investigación, ya que a diferencia de los tres canales RGB que poseen las imágenes, así como dos datos en H y W, para el presente proyecto, el N representará el número de muestras en este caso trabajaremos con 60000, las cuales son tanto simuladas como obtenidos por medio de mediciones reales facilitadas por la Universidad Salesiana de Quito. Como el valor de la transformada de Fourier tiene una sola dimensión, es decir no se necesita especificar dos componentes para definir la muestra, se puede asumir el valor de H como 1 y el valor de W será el contenido del vector después de las transformaciones, teniendo en cuenta que caffe acepta datos en formato database se usa HDF5 debido a su compatibilidad con C++.

Onda de corriente motor AC

La onda de corriente de un motor viene representada por la función coseno en los motores que son AC, la onda en función del tiempo tiene la siguiente forma:

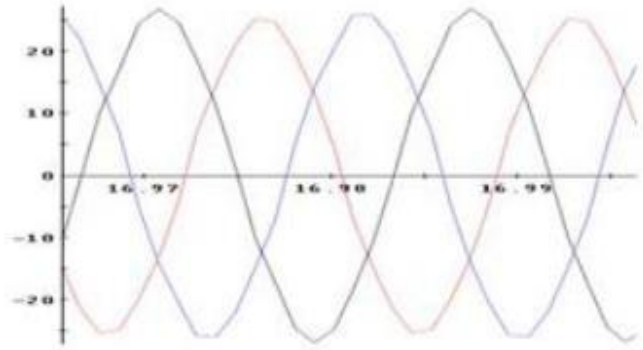

Fig. 1.3 Onda de motor AC en función del tiempo

Fuente: [19]

En el Artículo Científico escrito por V. Ferñao Pires bajo el nombre de "Motor square current signature analysis for induction motor rotor diagnstic" se menciona que la onda fundamental de la señal de corriente está representada por la siguiente ecuación:

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

$Donde$:

$i_a(t)$ $Representa\ la\ señál\ en\ el\ dominio\ del\ tiempo$
I_{max} $Representa\ la\ amplitud\ de\ onda\ de\ a\ corriente$
ω $Representa\ la\ frecuencia\ de\ la\ onda$

Ecuación 2 Ecuación de onda fundamental de señal de corriente

Fuente: [20]

Sin embargo, en otro Artículo Científico escrito por el mismo autor bajo el nombre de "Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics" se menciona que es recomendable añadir ruido blanco para realizar un mejor entrenamiento de la red, debido a que el ruido blanco es el ruido que se produce en los instrumentos de medición, puesto que en la obtención de la señal de una onda en un ambiente real poseerá ciertas variaciones debido a la presencia de este ruido[20].

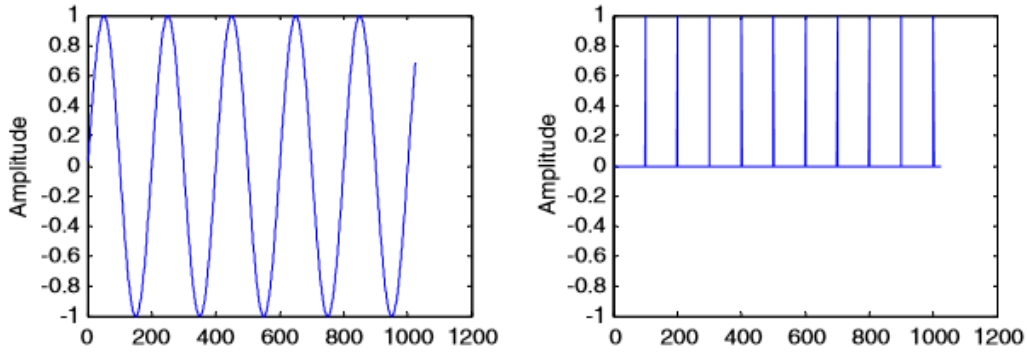

Fig. 1.4 Gráficas de señales simuladas onda sinusoidal pura y serie de impulsos

Fuente: [20]

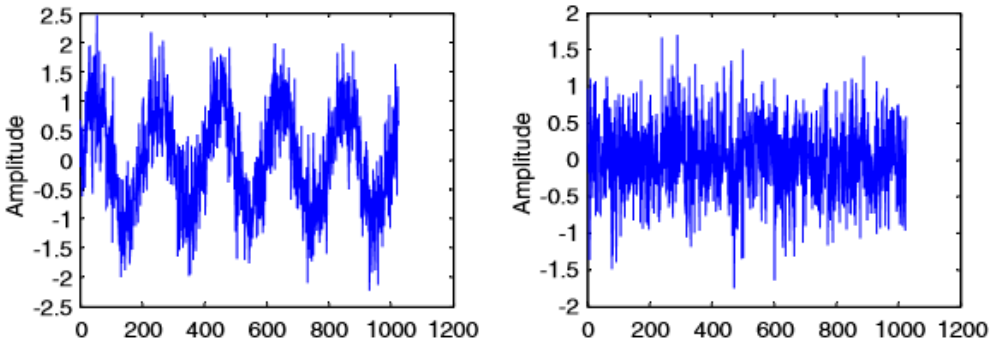

Fig. 1.5 Gráficas de señales simuladas con ruido blanco añadido

Fuente: [20]

Comparando las gráficas simuladas queda claro la importancia de agregar el ruido blanco para la obtención de muestras, esto es para que al generarlas se puedan acercar en gran parte a una señal de onda real.

Para el caso de la onda de corriente de motor con el fallo de barras rotas, se toma el caso anterior como base, debido a que la señal característica se mantiene, pero para simular este fallo se requiere la adición de otros términos los cuales están explicados a continuación[20].

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

I_{lsb} *Máximo valor de la amplitud del lado más bajo de la banda de corriente*

I_{usb}*Máximo valor de la amplitud del lado más alta de la banda de corriente*

Ecuación 3 Ecuación para simulación de onda con fallo de barras rotas

Fuente: [20]

Para obtener ondas similares a las que se podrían obtener en la vida real, estas muestras también requieren de la añadidura de ruido blanco.

Para el caso de la onda de corriente de motor con presencia del fallo de asimetría del eje es similar, es necesario añadir otros términos en la simulación los cuales serán explicados a continuación[20].

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

ωr *Frecuencia de la banda del espectro de la onda, en función del tiempo*

Ecuación 4 Ecuación para simulación de onda con fallo de asimetría del eje

Fuente: [20]

La onda obtenida por medio de esta ecuación también requiere de la adición de ruido blanco para que la simulación sea lo más parecida a una obtenida en la vida real.

1.2 Objetivos

Objetivo General

Desarrollar una red neuronal convolucional para la detección de fallos en maquinaria rotativa en tiempo real.

Objetivos Específicos

- Analizar el estado actual de la información sobre las redes neuronales convolucionales y sus aplicaciones.
- Analizar los fallos más comunes que se pueden presentar en maquinaria rotativa.
- Diseñar una red neuronal convolucional con la capacidad de predecir fallos por medio de la alimentación de datos.
- Elaborar pruebas o correcciones necesarias a la red desarrollada para demostrar su correcto funcionamiento.

CAPÍTULO II
METODOLOGÍA

2.1 Materiales

La información que se necesita para la elaboración de este proyecto se obtendrá mediante la investigación y la recolección de requerimientos del departamento de investigación de la Universidad Técnica de Ambato de la Facultad de Ingenierías en Sistemas Electrónica e Industrial.

2.2 Métodos

Modalidad de investigación

Una de las modalidades a utilizarse será la de tipo bibliográfica dado que se utilizará fuentes como libros, documentos y artículos de carácter científico para recopilar información sobre las redes neuronales convolucionales.

La investigación tendrá una modalidad aplicada porque para el desarrollo de la red para detección de fallos se aplicará los conocimientos adquiridos durante todo el largo de la carrera.

Recolección de información

La información necesaria para el desarrollo de este proyecto será obtenida mediante las bibliotecas y repositorios virtuales de las universidades del país, para el estado actual de las redes neuronales convolucionales, así como de sus arquitecturas se obtendrá información avalada de repositorios virtuales de organizaciones internacionales, debido a las características de este proyecto no se requiere población y muestra.

Procesamiento y análisis de la información

La información obtenida requerirá un proceso de limpieza y filtrado para eliminar lo defectuoso o no necesario para la investigación, luego será catalogada en base a su beneficio para el proyecto. De esta forma se obtendrá información útil para el desarrollo del marco teórico, así como para la especificación de las clases con las cuales va a trabajar la red.

Desarrollo del proyecto

Para el desarrollo del proyecto se consideraron las siguientes fases:

a. Descripción de metodología ágil a utilizarse para el desarrollo del proyecto

b. Descripción del arquitectura y tecnologías usadas para el desarrollo y funcionamiento de la red neuronal convolucional.

c. Descripción de la estructura y tipos de datos a ser manejados por la red.

d. Desarrollo y descripción de métodos para generación de muestras simuladas tanto para funcionamiento normal como para las dos clases de fallos.

e. Desarrollo y descripción de métodos necesarios para la creación del dataset.

f. Desarrollo y descripción de código necesario para la creación de la red.

g. Descripción de los entrenamientos de la red con diversas muestras.

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

Proceso

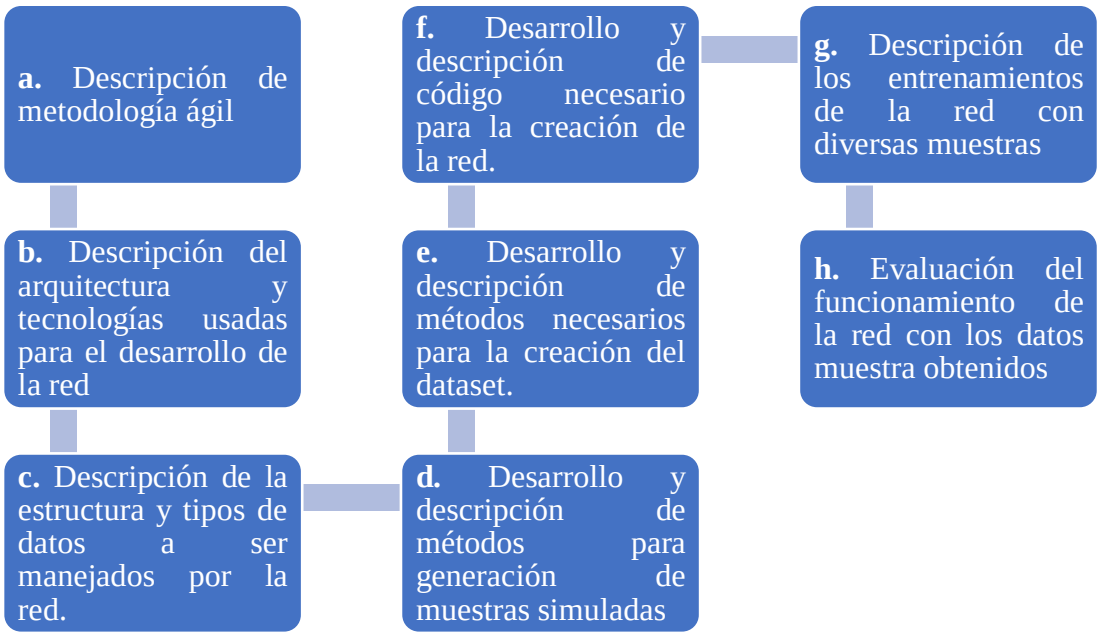

Fig. 2.1 Proceso para el desarrollo del proyecto

Fuente: Elaborado por el autor

CAPÍTULO III

RESULTADOS Y DISCUCIÓN

3.1 Análisis y discusión de resultados

Para el análisis del funcionamiento se tomó como referencia los estudios realizados con muestras en motores reales, de esta manera se consiguió demostrar la efectividad de la utilización de este tipo de redes para la detección de fallos en maquinaria rotativa. Se consiguieron resultados favorables debido al entrenamiento realizado con una gran cantidad de muestras, estos resultados se encuentran representados a continuación.

Clases	Normal	Asimetría Eje	Barras Rotas
Resultados obtenidos con varias pruebas	99.9999	99.9211	99.2042
	99.9999	99.9211	99.2042
	99.9999	99.9250	99.2042
	99.9999	99.9211	99.2035
	99.9999	99.9210	99.2042
	99.9999	99.9211	99.2042

Tabla 3.1 Pruebas de la red con diversas muestras

Fuente: Elaborado por el Autor

Los resultados de la probabilidad de una muestra normal nunca llegan al cien por ciento de acierto, debido a que existen diversos tipos de fallos que se pueden presentar, y al ser analizado por semejanzas con grandes cantidades de muestras. Las probabilidades obtenidas no llegan a alcanzar el máximo.

3.2 Desarrollo de la propuesta

3.2.1 Descripción de metodología ágil a utilizarse para el desarrollo del proyecto

Metodología XP (Extreme Programming)

La programación extrema nace como nueva disciplina de desarrollo de software hace aproximadamente seis años, y ha causado un gran impacto entre el colectivo de programadores del mundo. Kent Beck, su autor, es un programador que ha trabajado en múltiples empresas y que actualmente lo hace como programador en la conocida empresa automovilística DaimlerChrysler. Con sus teorías ha conseguido el respaldo de gran parte de la industria del software y el rechazo de otra parte. La programación

extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica[21].

Esta metodología ágil se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico[22].

Roles XP

Los roles de acuerdo con la propuesta original de Beck son:

Programador: El programador escribe las pruebas unitarias y produce el código del sistema[22].

Cliente: Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio[22].

Encargado de pruebas (Tester): Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas. [22].

Encargado de seguimiento (Tracker): Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración[22].

Entrenador (Coach): Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente[22].

Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas[22].

Gestor (Big boss): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación[22].

Ventajas

- Apropiado para entornos volátiles
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para los clientes, conocen las fechas de entrega de funcionalidades. Vital para los negocios
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente
- Permite tener realimentación de los usuarios muy útil.
- La presión está a lo largo de todo el proyecto y no en una entrega final.[21]

Proceso XP

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.

2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.

4. El programador construye ese valor de negocio.

5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración[22].

Metodología SCRUM

El SCRUM se asume como metodología extremadamente ágil y flexible, diseñado para definir un proceso de desarrollo interactivo e incremental se puede aplicar a cualquier

producto o en la gestión de cualquier actividad complejo. Esta metodología se basa en el desarrollo de aplicaciones incrementales centrado en el equipo con ciclos de iteración corta[23].

SCRUM se aplica a proyectos tanto pequeños como grandes luchando por liberar el proceso de cualquier barrera, el objetivo principal es conseguir una evaluación correcta del entorno en evolución, constantemente adaptándose al caos de intereses y necesidades, indicados y utilizados para el desarrollo de software en entornos complejos donde los requisitos cambian con cierta frecuencia y el camino utilizado para aumentar la productividad en este tipo de sistemas[23].

Roles [23]

Los involucrados directos: Son las personas que están comprometidas con el proyecto y el proceso de Scrum.

Product Owner: Es la persona que toma decisiones y es la que realmente conoce el negocio del cliente y su visión del producto. Se encarga de escribir las ideas del cliente, las ordena por prioridad y las coloca en el Product Backlog.

ScrumMaster: Es el encargado de comprobar que el modelo y la metodología funciona. Eliminará todos los inconvenientes que hagan que el proceso no fluya e interactuará con el cliente y con los gestores.

Equipo de desarrollo: Suele ser un equipo pequeño de 5-9 personas y tienen autoridad para organizar y tomar decisiones para conseguir su objetivo. Está involucrado en la estimación del esfuerzo de las tareas del Backlog.

Los involucrados indirectos: Aunque no son parte del proceso de Scrum, es necesario que parte de la retroalimentación de la salida del proceso y así poder revisar y planear cada sprint.

Usuarios: Es el destinatario final del producto.

Stakeholders: Las personas a las que el proyecto les producirá un beneficio. Participan durante las revisiones del Sprint.

Managers: Toma las decisiones finales participando en la selección de los objetos y los requisitos.

Características SCRUM: [23]

- Es un proceso ágil para gestionar y control de desarrollo de proyectos.
- Scrum resalta e impulsa el trabajo en equipo, el aprendizaje constante y una estructura que es flexible a los cambios que van sucediendo en la fase de desarrollo.
- Es un proceso que controla el caos resultante de necesidades e intereses en conflicto.
- Es una forma de aumentar la comunicación y maximizar la cooperación.
- Es una forma de detectar y eliminar cualquier impedimento que perturbe el desarrollo de producto.
- Escalable desde pequeños proyectos incluso grandes proyectos en toda la empresa.

Beneficios

- Reducción de riesgos.
- Mayor integración entre los miembros de los equipos.
- Resolución rápida de problemas.
- Progreso medido continuamente.
- Los clientes se convierten en parte del equipo de desarrollo.
- Entregas frecuentes de funcionalidades de trabajo.
- Discusiones diarias sobre el estado con el personal
- Profesionales de negocios y Las tecnologías trabajan juntas.

	XP	**SCRUM**
Tamaño del proyecto	Pequeños - Medianos	Pequeños -Medianos - Grandes
Tamaño del equipo	<10	<10 (Múltiples equipos)
Estilo del desarrollo	Iterativo y rápido	Iterativo y rápido
Estilo de código	Limpio y sencillo	No especificado
Entorno físico	Equipos en un mismo lugar	No especificado
Cultura de negocio	Colaborativo y operativo	No especificado

Tabla 3.2 Comparación de metodologías

Fuente: Elaborado por el Autor

Mediante el uso de un cuadro comparativo de las posibles metodologías para la elección de cual se adapte más a nuestro proyecto, se optó por la metodología XP, debido a sus beneficios entre los cuales están el permitir un análisis, diseño, desarrollo y pruebas de manera rápida, según [22], la metodología consta de varias fases las cuales ayudaran con el desarrollo del proyecto.

- Planificación de proyecto
- Diseño
- Codificación
- Pruebas

A continuación, se muestra el diagrama del proceso, detallado de un mantenimiento a un equipo en una empresa:

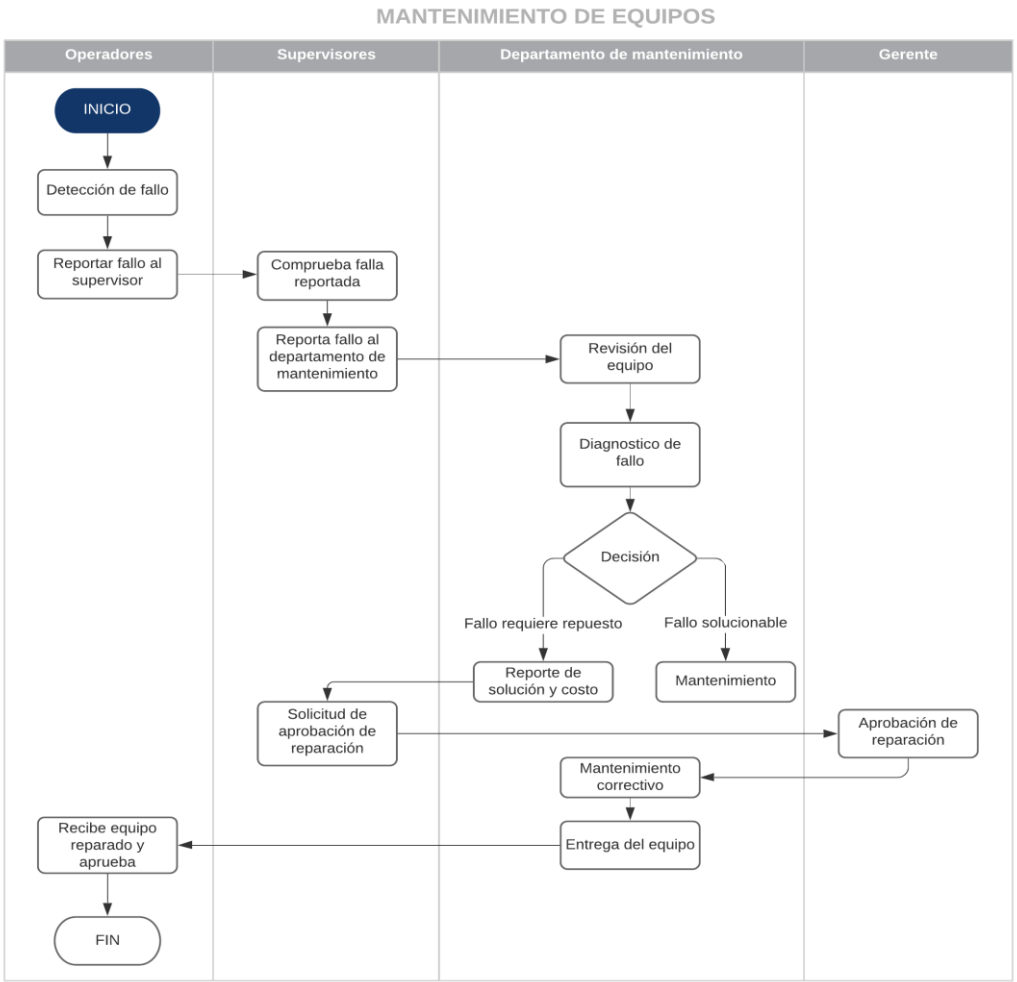

Fig. 3.1 Diagrama del proceso llevado a cabo en un mantenimiento

Fuente: Elaborado por el autor

Planificación del proyecto

Para la planificación se utilizan historias de usuario las cuales permitirán definir los requerimientos que tendrá el presente proyecto. Los requerimientos del proyecto fueron establecidos por un docente del departamento de investigación de la FISEI debido a que este es un proyecto de investigación.

Rol	Descripción
Programador	Autor del presente trabajo de investigación
Usuario final	Departamento de investigación de la FISEI
Tester	Autor del presente trabajo de investigación
Tracker	Autor del presente trabajo de investigación
Entrenador	Autor del presente trabajo de investigación
Consultor	Docente departamento de investigación de la FISEI
Gestor	Autor del presente trabajo de investigación

Tabla 3.3 Descripción de roles

Fuente: Elaborado por el Autor

Historias de usuario en base a los requerimientos establecidos.

Historia de Usuario	
Código: H1	**Usuario:** Usuario Final
Iteración Asignada: 1	**Prioridad de Negocio:** Alta
Riesgo de desarrollo: Alto	
Nombre de historia: Dataset para entrenamiento	
Descripción: La red neuronal convolucional requiere de gran cantidad de muestras para su entrenamiento razón por la cual se desea generar un dataset con muestras de funcionamientos de un motor en normalidad, así como bajo la influencia de fallos para alcanzar un nivel alto de acierto en la predicción.	
Tareas: • Diseño de arquitectura del dataset • Creación de dataset	

Tabla 3.4 Historia de usuario Dataset para entrenamiento

Fuente: Elaborado por el Autor

Historia de Usuario	
Código: H2	**Usuario:** Usuario Final
Iteración Asignada: 1	**Prioridad de Negocio:** Alta
Riesgo de desarrollo: Alto	
Nombre de historia: Generador de muestras	
Descripción: Teniendo en cuenta la necesidad de una gran cantidad de muestras y la dificultad para conseguirlas, se requiere un pequeño sistema capaz de generar muestras tanto de funcionamiento normal como con presencia de fallos, esto con el uso de algoritmos, de esta manera se logrará conseguir muestras similares a las reales.	
Tareas: • Selección de algoritmos necesarios para las simulaciones • Selección de algoritmos para añadir ruido blanco • Desarrollo de métodos para generación de muestras • Desarrollo de métodos para carga de muestras en el dataset	

Tabla 3.5 Historia de usuario Generador de muestras

Fuente: Elaborado por el Autor

Historia de Usuario	
Código: H3	**Usuario:** Usuario Final
Iteración Asignada: 2	**Prioridad de Negocio:** Alta
Riesgo de desarrollo: Alto	
Nombre de historia: Selección de la arquitectura	
Descripción: Se requiere una arquitectura con las características para agilitar la clasificación de fallos, debido a que por medio una buena elección el clasificador será optimo, para la selección se requiere de una arquitectura probada.	
Tareas: • Selección de una arquitectura para la clasificación de señales eléctricas • Comprobación de la funcionalidad teórica	

Tabla 3.6 Historia de usuario Selección de la arquitectura

Fuente: Elaborado por el 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 convolucional	
Descripción: El desarrollo de la red debe seguir la arquitectura elegida, debido a que de esta manera se conseguirá reducir el procesamiento en la clasificación y brindará velocidad al análisis, para lograr un funcionamiento rápido y ligero del programa.	
Tareas: • Selección del lenguaje a utilizar • Determinar las librerías necesarias para el desarrollo • Desarrollo de la red siguiendo los parámetros de la arquitectura • Desarrollo de los métodos necesarios para el funcionamiento de la red	

Tabla 3.7 Historia de usuario Desarrollo de la red neuronal convolucional

Fuente: Elaborado por el 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 interfaz	
Descripción: El proyecto requiere de un interfaz simple la cual permitirá elegir una muestra para analizarse y a su vez permitirá visualizar la muestra graficada y la sección que se tomara para el análisis, debido a que lo importante del proyecto es el análisis de los datos para la obtención de una pronta predicción.	
Tareas: • Diseño del interfaz • Desarrollo del método para selección de muestras • Desarrollo de los métodos para graficar las muestras • Desarrollo de código para muestras de resultado	

Tabla 3.8 Historia de usuario Desarrollo de interfaz

Fuente: Elaborado por el Autor

Actividades

- **Historia:** Dataset para entrenamiento

Tarea	
Código: T1	**Código de historia: H1**
Nombre de tarea: Diseño de arquitectura del dataset	
Tipo de tarea: Modelado dataset	**Puntos de Estimación:** 2
Duración: 1 semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Diseñó de la estructura que poseerá el dataset a utilizarse, esto en base a la arquitectura que admite la librería caffe.	

Tabla 3.9 Diseño de arquitectura del dataset – Historia 1

Fuente: Elaborado por el Autor

- **Historia:** Dataset para entrenamiento

Tarea	
Código: T2	**Código de historia: H1**
Nombre de tarea: Creación de dataset	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 1 semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Creación del dataset	

Tabla 3.10 Creación del dataset – Historia 1

Fuente: Elaborado por el Autor

- **Historia:** Generador de muestras

Tarea	
Código: T3	**Código de historia: H2**
Nombre de tarea: Selección de algoritmos necesarios para las simulaciones	
Tipo de tarea: Investigación	**Puntos de Estimación:** 1
Duración: 3 días	
Programador: Jonathan Javier Shulca Andrade	

Descripción: Selección de algoritmos necesarios para las simulaciones en base a los fallos que se van a estudiar.

Tabla 3.11 Selección de algoritmos necesarios para las simulaciones – Historia 2

Fuente: Elaborado por el Autor

- **Historia:** Generador de muestras

Tarea	
Código: T4	**Código de historia: H2**
Nombre de tarea: Selección de algoritmos para añadir ruido blanco	
Tipo de tarea: Investigación	**Puntos de Estimación:** 1
Duración: 2 días	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Selección de algoritmos para añadir ruido blanco	

Tabla 3.12 Selección de algoritmos para añadir ruido blanco – Historia 2

Fuente: Elaborado por el Autor

- **Historia:** Generador de muestras

Tarea	
Código: T5	**Código de historia: H2**
Nombre de tarea: Desarrollo de métodos para generación de muestras	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 1 semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Desarrollo de métodos para generación de muestras	

Tabla 3.13 Desarrollo de métodos para generación de muestras – Historia 2

Fuente: Elaborado por el Autor

- **Historia:** Generador de muestras

Tarea	
Código: T6	**Código de historia: H2**
Nombre de tarea: Desarrollo de métodos para carga de muestras en el dataset	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 4 días	
Programador: Jonathan Javier Shulca Andrade	

Descripción: Desarrollo de métodos para carga de muestras en el dataset

Tabla 3.14 Desarrollo de métodos para carga de datos en el dataset – Historia 2

Fuente: Elaborado por el Autor

- **Historia:** Selección de la arquitectura

Tarea	
Código: T7	**Código de historia: H3**
Nombre de tarea: Selección de una arquitectura para la clasificación de señales eléctricas	
Tipo de tarea: Investigación	**Puntos de Estimación:** 3
Duración: 1 semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Selección de una arquitectura con características que agilicen la clasificación de señales eléctricas	

Tabla 3.15 Desarrollo de métodos para carga de datos en el dataset – Historia 3

Fuente: Elaborado por el Autor

- **Historia:** Selección de la arquitectura

Tarea	
Código: T8	**Código de historia: H3**
Nombre de tarea: Comprobación de la funcionalidad teórica	
Tipo de tarea: Investigación	**Puntos de Estimación:** 2
Duración: 1 Semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Comprobación de la funcionalidad de la arquitectura seleccionada en base a estudios y comprobaciones desarrolladas en artículos científicos	

Tabla 3.16 Comprobación de la funcionalidad teórica – Historia 3

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de la red neuronal convolucional

Tarea	
Código: T9	**Código de historia: H4**
Nombre de tarea: Selección del lenguaje a utilizar	
Tipo de tarea: Investigación	**Puntos de Estimación:** 2

Duración: 3 días
Programador: Jonathan Javier Shulca Andrade
Descripción: Seleccionar un lenguaje de programación en base los requerimientos para facilitar el desarrollo

Tabla 3.17 Selección del lenguaje a utilizar – Historia 4

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de la red neuronal convolucional

Tarea	
Código: T10	**Código de historia: H4**
Nombre de tarea: Determinar las librerías necesarias para el desarrollo	
Tipo de tarea: Investigación	**Puntos de Estimación:** 2
Duración: 1 semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Determinar las librerías necesarias a utilizarse con el propósito de agilitar el desarrollo	

Tabla 3.18 Determinar las librerías necesarias para el desarrollo – Historia 4

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de la red neuronal convolucional

Tarea	
Código: T11	**Código de historia: H4**
Nombre de tarea: Desarrollo de la red siguiendo los parámetros de la arquitectura	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 1 mes	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Desarrollo de la red siguiendo los parámetros de la arquitectura utilizando clases y métodos para tener un programa ágil	

Tabla 3.19 Desarrollo de la red siguiendo los parámetros de la arquitectura – Historia 4

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de la red neuronal convolucional

Tarea	
Código: T12	**Código de historia: H4**
Nombre de tarea: Desarrollo de los métodos necesarios para el funcionamiento de la red	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 3 semanas	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Desarrollo de métodos complementarios para el correcto funcionamiento de la red.	

Tabla 3.20 Desarrollo de los métodos necesarios para el funcionamiento de la red – Historia 4

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de interfaz

Tarea	
Código: T13	**Código de historia: H5**
Nombre de tarea: Diseño del interfaz	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 1 semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Diseño de una interfaz sencilla.	

Tabla 3.21 Diseño del interfaz red – Historia 5

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de interfaz

Tarea	
Código: T14	**Código de historia: H5**
Nombre de tarea: Desarrollo del método para selección de muestras	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 1 semana	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Desarrollo de método para lectura de archivos de tipo db	

Tabla 3.22 Desarrollo del método para selección de muestras – Historia 5

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de interfaz

Tarea	
Código: T15	**Código de historia: H5**
Nombre de tarea: Desarrollo de los métodos para graficar las muestras	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 2 semanas	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Desarrollo de método para la graficar las muestras que serán analizadas	

Tabla 3.23 Desarrollo de los métodos para graficar las muestras – Historia 5

Fuente: Elaborado por el Autor

- **Historia:** Desarrollo de interfaz

Tarea	
Código: T16	**Código de historia: H5**
Nombre de tarea: Desarrollo de código para muestras de resultado	
Tipo de tarea: Desarrollo	**Puntos de Estimación:** 2
Duración: 2 semanas	
Programador: Jonathan Javier Shulca Andrade	
Descripción: Desarrollo de código para mostrar resultado	

Tabla 3.24 Desarrollo de código para muestras de resultado – Historia 5

Fuente: Elaborado por el Autor

Diseño

Diagrama de secuencia

El diagrama de secuencia simboliza las interacciones generales que ocurren a través del proyecto, se incluye la figura de cómo se transportan los datos para ser analizados y clasificados hasta llegar a ser presentados a modo de información al usuario final.

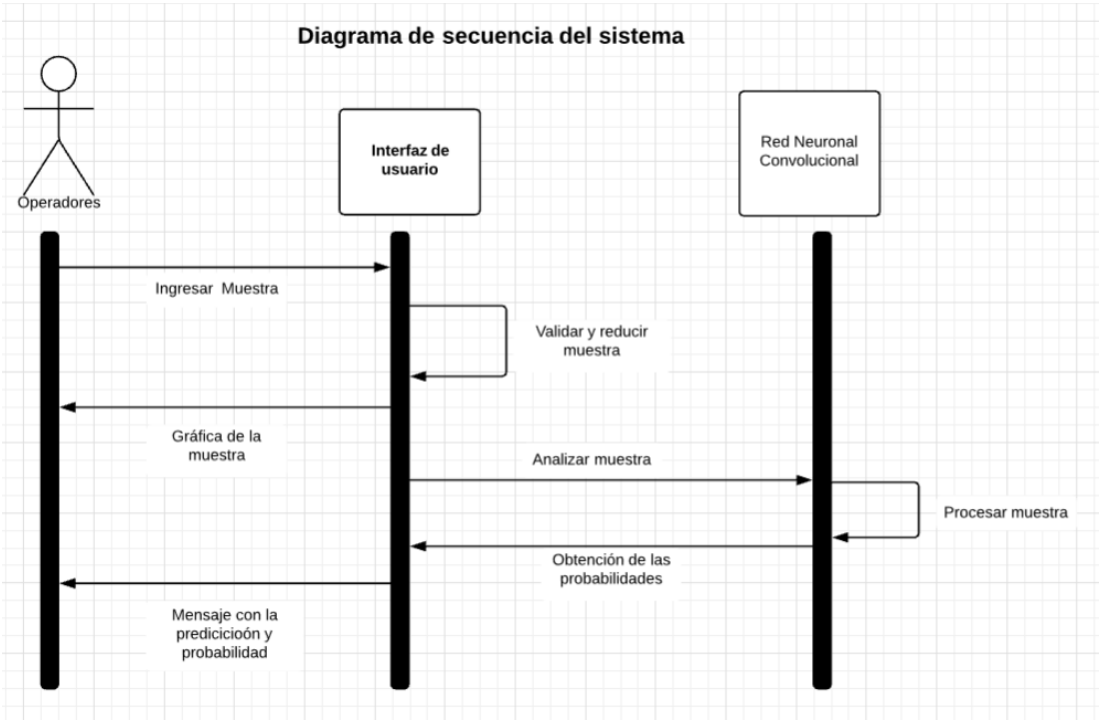

Fig. 3.2 Diagrama de la secuencia del sistema

Fuente: Elaborado por el autor

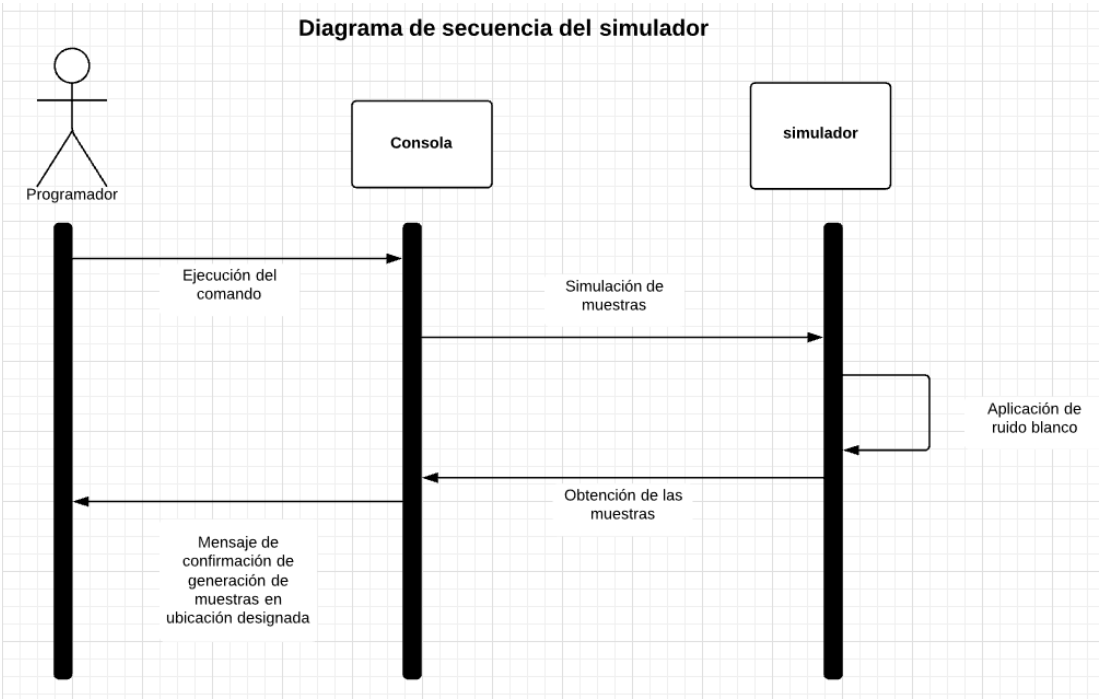

Fig. 3.3 Diagrama de la secuencia del simulador

Fuente: Elaborado por el autor

Para la fase de diseño se utilizó lo recomendado por la metodología XP, lo cual es el uso de tarjetas CRC (Clase – Responsabilidad - Colaboración).

Dataset para entrenamiento	
Responsabilidad	**Colaboradores**
Diseñar un dataset con las muestras necesarias para el entrenamiento de la red.	Definición de blob para caffe

Tabla 3.25 Tarjeta CRC Dataset de entrenamiento

Fuente: Elaborado por el autor

Generador de muestras	
Responsabilidad	**Colaboradores**
Simulación de muestras de funcionamiento normal y con fallos	Librerías HDF5 QT creator Librería Eigen Librería Matemáticas Librería H5

Tabla 3.26 Tarjeta CRC Generador de muestras

Fuente: Elaborado por el autor

Selección de la arquitectura	
Responsabilidad	**Colaboradores**
Búsqueda de arquitectura para desarrollo de la red	Fuentes bibliográficas internacionales

Tabla 3.27 Tarjeta CRC Selección de la arquitectura

Fuente: Elaborado por el autor

Desarrollo de la red neuronal convolucional	
Responsabilidad	**Colaboradores**
Diseño y desarrollo de la red neuronal basada en la arquitectura elegida, estableciendo los respectivos controles y características para su funcionamiento	Librerías caffe Librería Eigen QT creator

Tabla 3.28 Tarjeta CRC Desarrollo de la red neuronal convolucional

Fuente: Elaborado por el autor

Desarrollo de la interfaz	
Responsabilidad	**Colaboradores**
Diseño y desarrollo de una interfaz para el manejo del software, el cual permitirá buscar y abrir la muestra que se quiera analizar, así como visualizar la onda de la muestra y mostrar la clase a la que pertenece.	QT creator Librerías estándar de C++

Tabla 3.29 Tarjeta CRC Desarrollo de la interfaz

Fuente: Elaborado por el autor

Codificación

Para este proyecto se utilizó el lenguaje de programación C++ tanto para el desarrollo del simulador de muestras, así como para el desarrollo de la red neuronal convolucional y fue desarrollado en Linux – Ubuntu, para una mejor comprensión la explicación sobre lo utilizado para el desarrollo, así como la explicación del código se encuentra a continuación.

3.1.2 Descripción del arquitectura y tecnologías usadas para el desarrollo de la red neuronal convolucional.

Arquitectura de red neuronal convolucional

Generalmente la arquitectura de una CNN está compuesta por tres partes las cuales son una capa convolucional, capa de submuestreo y capa completamente conectada. una CNN pude poseer varias arquitecturas, pero en este caso utilizara únicamente una.

La capa convolucional es la encargada de obtener el mapa de características de la imagen que se analiza.

La capa de submuestreo es la encargada de reducir las muestras antes de analizarlas, con el propósito de agilitar la clasificación.

La capa completamente conectada es la ultima capa y se encarga de la clasificación de las muestras.

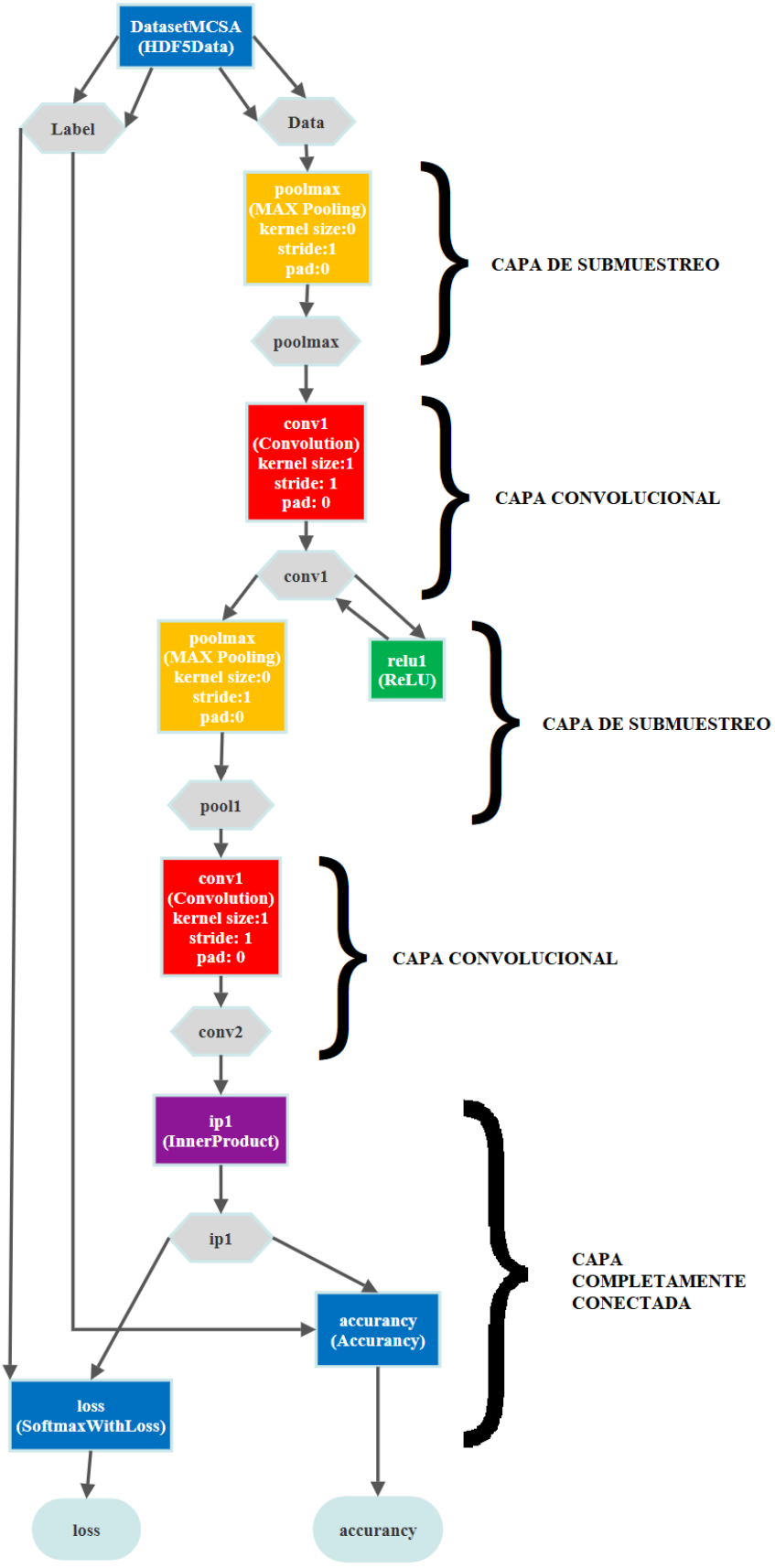

Fig. 3.4 Arquitectura CNN desarrollada

Fuente: Elaborado por el autor

Explicación de la arquitectura

Esta arquitectura posee características para facilitar el entrenamiento, usando un max-pooling. El cual permite tomar una muestra completa y reducirla en pequeñas fracciones para posteriormente analizarlas. Las capas convolucionales, son utilizadas para la obtención de características en las muestras, las cuales permitirán su clasificación.

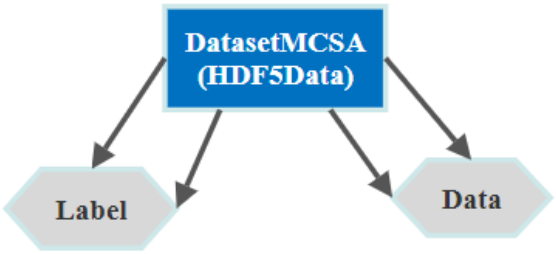

Fig. 3.5 Ingreso de los dataset a la red

Fuente: Elaborado por el autor

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. La fig. 3.5 en la capa de entrada.

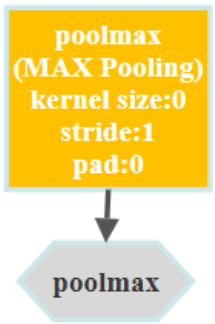

Fig. 3.6 Aplicación del max-pooling para análisis de muestra

Fuente: Elaborado por el autor

Se aplica un max-pooling al inicio del análisis con el fin de reducir las dimensiones de las muestras de entrada en este caso a 1x5. De esta manera el análisis de la muestra completa se lo realiza por partes con el fin de conseguir que el procesamiento sea más rápido. La Fig. 3.6 es la capa de pooling.

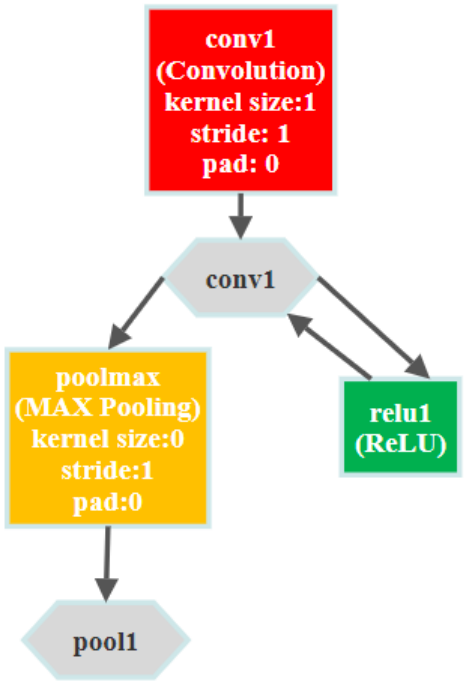

Fig. 3.7 Ingreso a la primera capa de convolución y reducción de la muestra

Fuente: Elaborado por el autor

La muestra reducida ingresa en la capa convolucional, 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. 3.7 se encuentra la primera capa de convolución, la función de activación ReLU y la segunda capa de pooling.

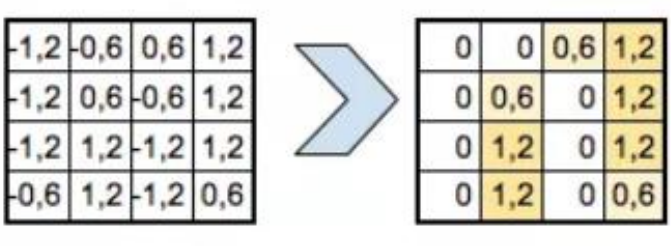

Fig. 3.8 Ejemplo de muestra después de la aplicación de ReLu

Fuente: Elaborado por el autor

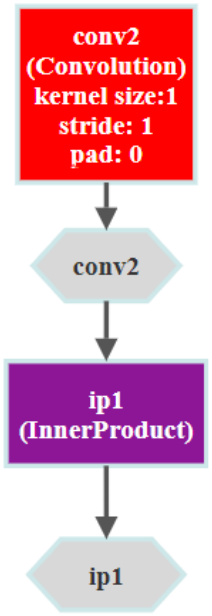

Fig. 3.9 Segunda capa de convolución

Fuente: Elaborado por el 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. 3.9 muestra la segunda capa de convolución y la capa de InnerProduct.

Fig. 3.10 Etapa de clasificación de las muestras

Fuente: Elaborado por el 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. 3.10 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.

Función de activación

La función de activación es la encargada de generar una salida partiendo de un valor de entrada. Existen varias funciones y cada una de ellas posee diversas características. Para las redes neuronales convolucionales existe la función RelU, la cual se caracteriza por su buen desempeño en redes convolucionales. Al poseer un comportamiento lineal para las entradas positivas evita el estancamiento provocado la saturación, debido a que los valores positivos son mantenidos y los valores negativos los anula.

Parámetros de la red

La red neuronal convolucional utiliza varios parámetros los cuales se conocen como:

Número de entradas: 512

Número de neuronas de entrada: 1

Número de neuronas intermedias: 711

Número de neuronas de salida: 3

El numero de entradas se estable por el tamaño de la muestra que se va a manejar. El número de neuronas de entrada es el total de muestras que va a ser capaz de analizar a la vez. El número de neuronas intermedias es conocido también como la capa oculta y es el total de características conseguidas por medio de los entrenamientos. El total se incrementa proporcionalmente con el numero de entrenamientos. Las neuronas de salida son el numero total de clases con el cual se va a trabajar.

3.2.3 Tecnologías utilizadas para el desarrollo y funcionamiento de la red neuronal convolucional

Principalmente este proyecto fue desarrollado para su funcionamiento sobre Linux, utilizando el lenguaje de programación C++. A pesar de la existencia de varios programas que facilitan el desarrollo de lo que son las redes neuronales se decidió optar por la que se asumió es de mejor utilidad para el desarrollo de redes convolucionales, basado en que posee librerías específicas para su implementación, así como para optimizar su funcionamiento, el manejo de los datos y la creación de las

interfaces. Entre las librerías usadas están Caffe, la cual es una librería usada extensivamente por la industria del software para el desarrollo de redes neuronales convoluciones. La librería Eigen perteneciente a C++, comúnmente utilizada para álgebra lineal fue de uso para la aplicación del algoritmo de FFT. HDF5, una librería de C++ la cual permite guardar grandes cantidades de datos en archivos de manera jerárquica fue utilizada para generar una mayor cantidad de datos para el entrenamiento de la red neuronal. Por último, la librería usada para crear las interfaces fue QT creator el cual es un framework usado para la creación de interfaces gráficas.

Qt Creator

Fig. 3.11 Qt Creator

Fuente: Elaborado por el autor

Es un entorno para el desarrollo integrado multiplataforma en C++, el cual también permite la creación de interfaces y se centra en proporcionar características de ayuda para los nuevos usuarios que desean aprender, posee un avanzado editor de código el cual tiene soporte para C++, QML y ECMAscript.

Las interfaces pueden ser diseñadas directamente en C++ o a su vez se puede generar automáticamente por medio de la utilización de herramientas gráficas. En este proyecto se utilizó Qt Desinger, el cual se encuentra integrado para ayudar a los usuarios en el diseño.

Descripción de las librerías para la creación de dataset.

Para la generación del dataset el cual tendrá muestras de todas las clases que analizaremos, se desarrolló un programa el cual es un proyecto de QT creator. Este

programa es configurable y posee ciertas opciones, pero siempre se debe tener en cuenta que los parámetros se deben adaptar a la entrada del Blob de Caffe, debido a que su estructura se asemeja a la de un vector grande. Los datos generados serán guardados siguiendo dicha estructura, este programa está desarrollado en el lenguaje C++ por lo cual requiere de varias librerías para su funcionamiento entre las cuales están las siguientes:

```
#include <iostream>
#include <math.h>
#include <chrono>
#include <ctime>
#include <unistd.h>
#include <fstream>
#include <random>
#include <algorithm>
#include <eigen3/unsupported/Eigen/FFT>
#include <H5Cpp.h>
#include <H5File.h>
#include <stdlib.h>
```

Fig. 3.12 Librerías del programa generador de muestras

Fuente: Elaborado por el autor

Entre las librerías declaradas están las de Eigen, Matemáticas, estándar y las de H5 todas estas librerías nos permitirán guardar las muestras en formato base de datos para poder ser usadas en caffe.

3.2.4 Desarrollo y descripción de métodos para generación de muestras simuladas tanto para funcionamiento normal como para las dos clases 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 Valores aceptados en los parámetros del generador de muestras

Fuente: Elaborado por el 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 esta 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. 3.13 Código donde se generan las muestras de funcionamiento normal

Fuente: Elaborado por el 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. 3.14 Código donde se genera las muestras de fallo de barras rotas

Fuente: Elaborado por el autor

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

```
//introducir falla por asimetria del eje
cout << "Generando muestras de asimetria del eje" << endl;
for(size_t i = 0; i < tamanhovector; 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*tiemponuestreo));
        argumento += tiemponuestreo;
    }
    muestras.push_back(muestra);
}
if(generarcsv){
    archivofunciontiempo.open("muestraasimetria" + 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. 3.15 Código donde se genera las muestras de fallo de asimetría de eje

Fuente: Elaborado por el autor

Para añadir más tipos de fallos se requiere agregar un número igual de muestras al vector de vectores, es decir si asumimos M como el número de clases y N el número de muestras el tamaño del vector de vectores tendría un tamaño de 3*N el cual si se agregase más clases solo debería cambiar a modo de (3+M)*N conservando así el mismo número de muestras en cada clase.

Una vez obtenido las muestras es necesario realizar la conversión a FFT para obtener el espectro de frecuencias, para esto se utiliza la librería Eigen, como esta operación se debe realizar a cada muestra es necesario iterar en el vector de vectores nuestras.

```
248     cout << "Convertir a FFT" << endl;
249     //Convertir a FFT
250     Eigen::FFT<float> fft;
251     std::vector<std::vector<std::complex<float>>> muestrasFFT;
252     std::vector<std::complex<float> > muestraFFT;
253
254     for(size_t i = 0; i < muestras.size(); i++){
255         muestra.clear();
256         muestra = muestras.at(i);
257         fft.fwd(muestraFFT, muestra);
258         muestrasFFT.push_back(muestraFFT);
259     }
```

Fig. 3.16 Código para la conversión de muestras a FFT

Fuente: Elaborado por el autor

El resultado obtenido será un número complejo el cual será añadido al vector de vectores muestrasFFT, para proceder con la conversión del espectro a escala de deciBeles (dB).

```
257     //Convertir a dB
258     cout << "Convertir a dB" << endl;
259     std::vector<std::vector<float>> muestrasdB;
260     std::vector<float> muestradB;
261     for(size_t i = 0; i < muestrasFFT.size(); i++){
262         muestradB.clear();
263         for(size_t j = 0; j < muestrasFFT.at(i).size(); j++){
264             muestradB.push_back(20*log(std::abs(muestrasFFT.at(i).at(j))/muestrasFFT.at(i).size()));
265         }
266         muestrasdB.push_back(muestradB);
267     }
```

Fig. 3.17 Código para la conversión a dB

Fuente: Elaborado por el autor

Debido a que la mayoría de las veces los espectros de amplitud se ven representados en decibelios, usar esta unidad de medida facilita la apreciación de pequeños componentes que se encuentren en la señal. Para la conversión a dB, se tomó como referencia los cálculos descritos en el paper de Michael Cerna y Audrey F. Harvey con el nombre de “The fundamentals of FFT-Based signal análisis and measurement” en el cual menciona que con la utilización de una ecuación se puede calcular la relación en decibelios a partir de los valores de amplitud. En este caso también se utiliza la función “abs” para obtener los valores absolutos tanto de la componente real, así como de la imaginaria. Estos valores obtenidos son guardados en un vector de vectores “muestradB”. Obtenidos los datos de esta manera requieren de una normalización, la cual es la conversión de los parámetros a la entrada de la red, en este caso los límites para la entrada de números a la red deben ser de [-1,1]. Para conseguir una mejor visualización de los datos se debe realizar una re escalación para que así todos los números se encuentren tanto normalizados como estandarizados[24].

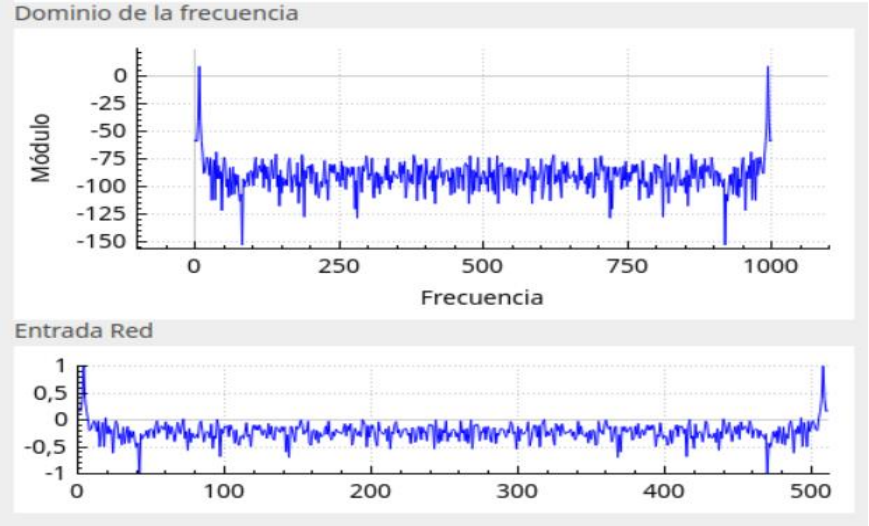

Fig. 3.18 Gráfica de onda antes y después de ser normalizada

Fuente: Elaborado por el autor

3.2.5 Desarrollo y descripción de métodos necesarios para la creación del dataset.

Con nuestra muestra normalizada y guardada en un vector se puede pasar a la creación del dataset. Para esto serán necesarias las librerías HDF5 debido a que con ellas se pueden crear archivos para almacenar datos basados en modelos que permiten organizar y acceder lógicamente a los datos. Se define dos nombres de dataset de los cuales uno será “label” en el cual se guardará el número que corresponde a cada clase, para definirlos se usará 0 para funcionamiento normal, 1 para falla por barras rotas y 2 para asimetría de eje. El segundo dataset adoptara el nombre de “data” en el cual se guardarán los datos del vector.

```
cout << "Generando archivo final de muestras" << endl;
// Open an existing file and dataset.
//Generacion Dataset
const H5std_string FILE_NAME(nombrearchivo + ".h5");
const H5std_string DATASET_NAME1("data");
const H5std_string DATASET_NAME2("label");
const int   RANK = 4;
H5File file(FILE_NAME, H5F_ACC_TRUNC);
hsize_t dims[RANK];                    // dataset dimensions
dims[0] = muestrasNormalizada.size();
dims[1] = 1;
dims[2] = 1;
dims[3] = muestrasNormalizada.at(0).size();

cout << "dims[0]" << muestrasNormalizada.size() << endl;
cout << "dims[1]" << muestrasNormalizada.at(0).size() << endl;
DataSpace dataspace(RANK, dims);
```

Fig. 3.19 Código para la creación de los dataset

Fuente: Elaborado por el autor

Es necesario crear un vector de tipo double que posea la misma estructura de un blob, con los parámetros de n muestras, 1 canal, 1 de ancho y el número de puntos de la onda para este caso será de 512, el cual será llenado por medio de un “for” con los datos del vector que guarda los datos normalizados, después se escribe en el dataset datos.

```
309     DataType datatype(H5::PredType::NATIVE_DOUBLE);
310     DataSet dataset = file.createDataSet(DATASET_NAME1, datatype, dataspace);
311     cout << "Generando HF5" << endl;
312     double arrayMuestrasnormalizada[dims[0]][dims[1]][dims[2]][dims[3]];
313     for(unsigned long i = 0; i < muestrasNormalizada.size(); i++){
314         for(size_t j = 0; j < muestrasNormalizada.at(i).size(); j++){
315             arrayMuestrasnormalizada[i][0][0][j] = muestrasNormalizada.at(i).at(j);
316         }
317     }
318     dataset.write(arrayMuestrasnormalizada, PredType::NATIVE_DOUBLE);
319     dataset.close();
320     dataspace.close();
```

Fig. 3.20 Código de carga de datos en dataset datos

Fuente: Elaborado por el autor

Para la carga del dataset “label” el ultimo parámetro tendrá los valores 0,1,2, por lo que las dimensiones cambian, se hace un nuevo proceso para el “for” teniendo en cuenta que las muestras hasta N son de funcionamientos correctos, hasta 2*N son de fallo de barras rotas y los 3*N son los fallos de asimetría de eje basado en esto se asigna el número en cada vector.

```
const int RANKcategorias = 4;
hsize_t dimsCat[RANKcategorias];
dimsCat[0] = muestrasNormalizada.size();
dimsCat[1] = 1;
dimsCat[2] = 1;
dimsCat[3] = 1;
DataSpace dataspaceCat(RANKcategorias, dimsCat);
DataType datatypeCat(H5::PredType::NATIVE_DOUBLE);
DataSet datasetCat = file.createDataSet(DATASET_NAME2, datatypeCat, dataspaceCat);
double categoriasMuestrasnormalizada[dimsCat[0]][dimsCat[1]][dimsCat[2]][dimsCat[3]];
for(unsigned long i = 0; i < muestrasNormalizada.size(); i++){
    if(i < tamanhovector) tipoerror = 0;
    else if((i >= tamanhovector) && (i < 2*tamanhovector)) tipoerror = 1;
    else tipoerror = 2;
    categoriasMuestrasnormalizada[i][0][0][0] = int(tipoerror);
    std::cout << "Tipo error: " <<  tipoerror << " Tipo error vector: " << categoriasMuestrasnormalizada[i][0][0][0]  <<  std::endl;
}

datasetCat.write(categoriasMuestrasnormalizada, PredType::NATIVE_DOUBLE);

datasetCat.close();
dataspaceCat.close();
file.close();
```

Fig. 3.21 Código de clasificación y carga de datos en dataset label

Fuente: Elaborado por el autor

Los archivos generados se pueden revisar con la utilización del software HDF5 Compass, por medio de este se pueden revisar los dataset creados, también se puede visualizar la estructura que poseen los dataset, así como se puede verificar el número de muestras.

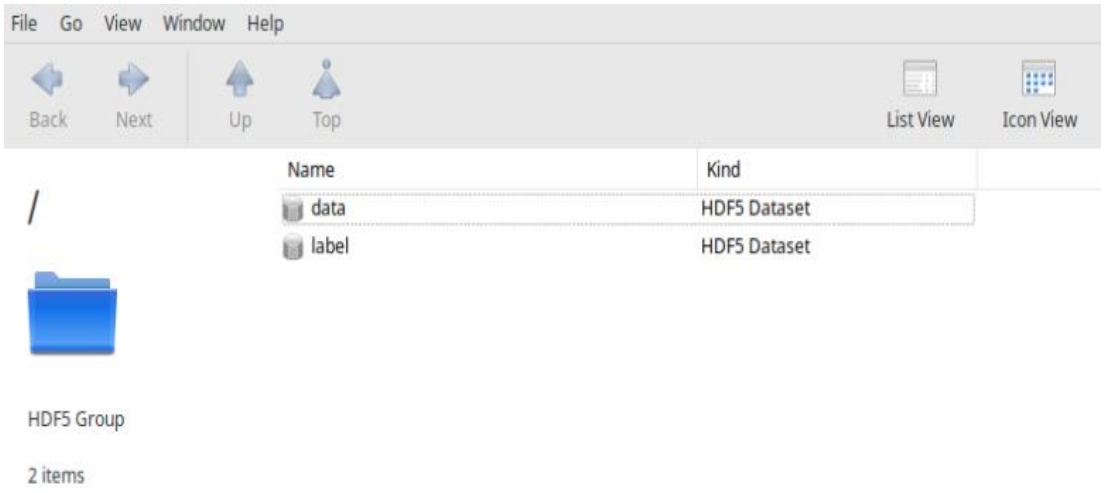

Fig. 3.22 Dataset creados para entrenamiento

Fuente: Elaborado por el autor

En el dataset data se pueden visualizar los 512 datos generados, así como también se puede visualizar la estructura que posee la base de datos compatible con el blob de entrada de caffe.

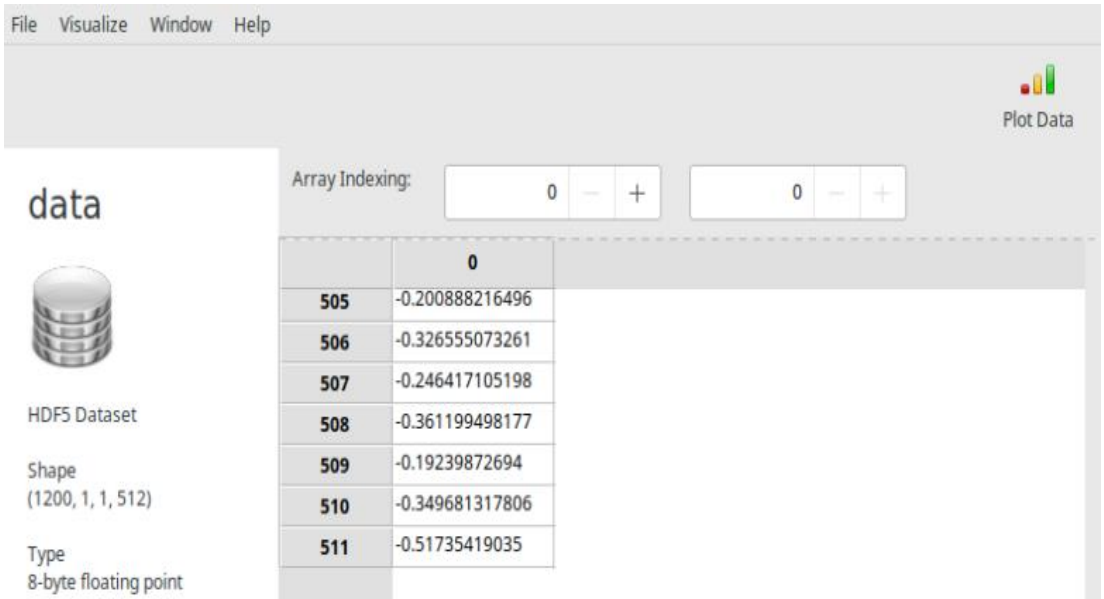

Fig. 3.23 Dataset data con datos muestra

Fuente: Elaborado por el autor

La estructura del dataset “label” también es compatible con el blob de entrada de caffe, pero en este caso solo posee un dato.

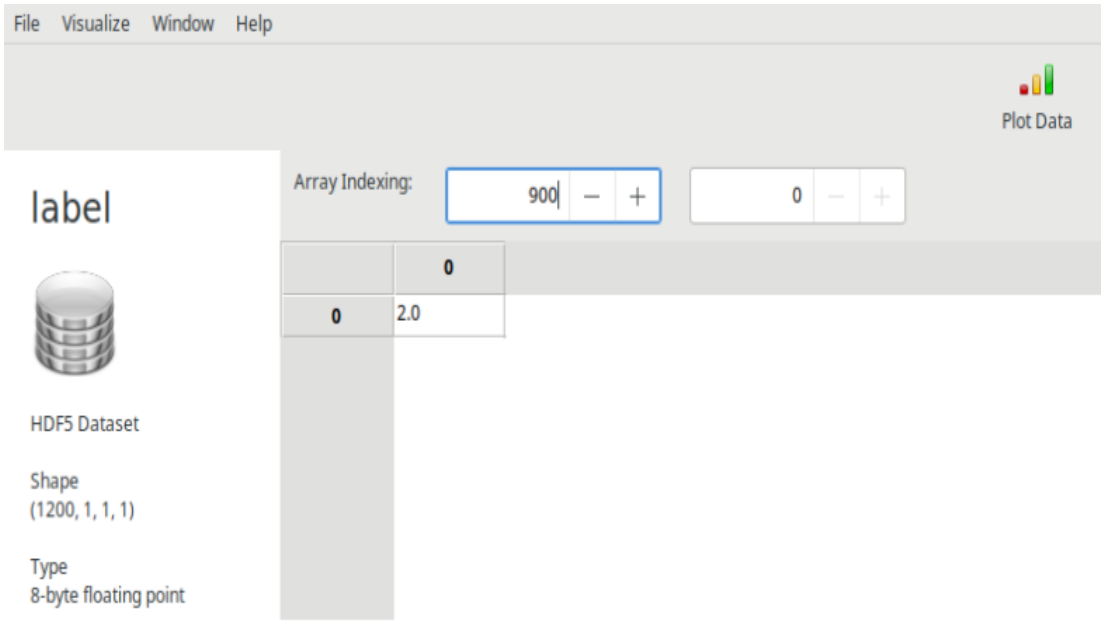

Fig. 3.24 Dataset label con datos muestra

Fuente: Elaborado por el autor

La razón por la cual solo se generan 400 muestras a la vez es debido a que caffe puede apilar los datos uno tras otro. La librería de HDF5, es utilizada para que los archivos del dataset se ordenen continuamente y así poder utilizarlos en el modelo de caffe.

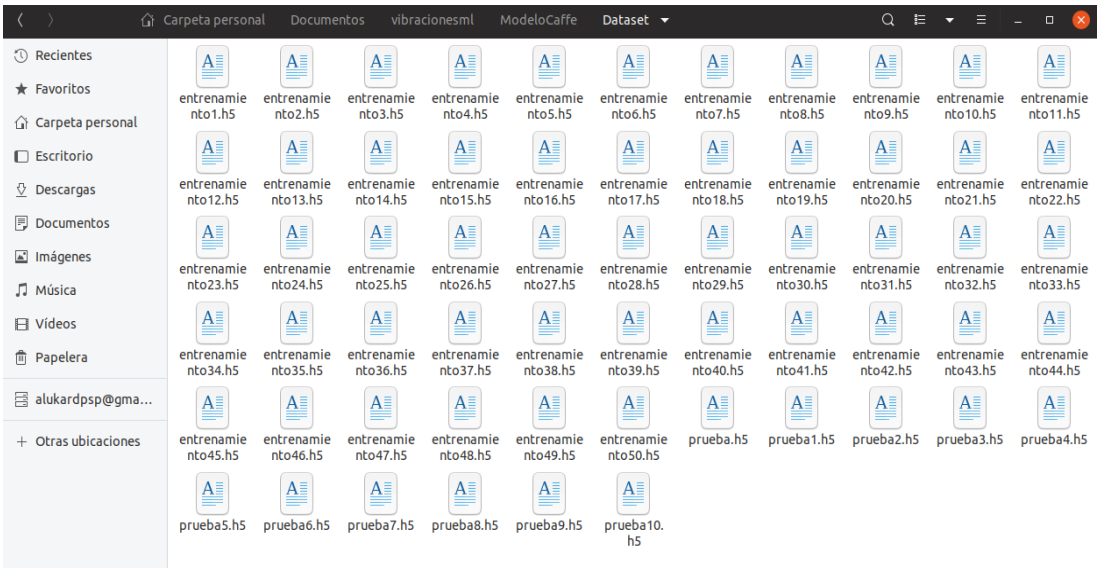

Fig. 3.25 Archivos del dataset ordenados en el modelo de caffe

Fuente: Elaborado por el autor

3.2.6 Desarrollo y descripción de código necesario para la creación de la red.

Para la creación de la red es necesario diversos tipos de archivos, los cuales guardarán tanto los parámetros como la estructura de la red, estos archivos serán necesarios para la utilización.

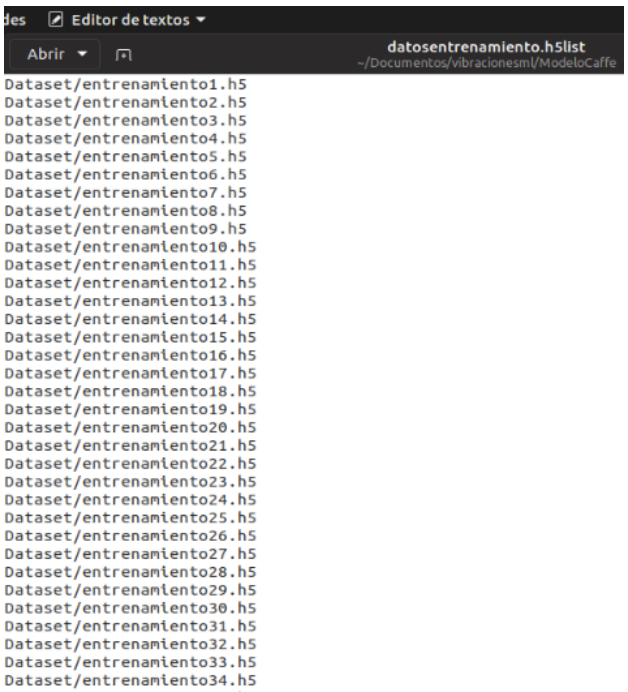

Fig. 3.26 Directorio de archivos para entrenamientos automáticos

Fuente: Elaborado por el autor

Teniendo en cuenta que las muestras generadas son en grupos de cuatrocientas cada una, para conseguir un total de 60000 muestras se generaron diversos archivos los cuales son utilizados por medio de un archivo de direcciones que posee la ubicación de todas las muestras que se usaran para el entrenamiento, este será utilizado desde el archivo que carga los parámetros de la red.

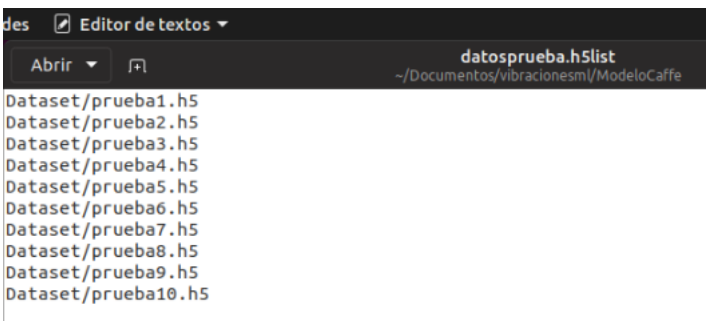

Fig. 3.27 Directorio de archivos para pruebas automáticas

Fuente: Elaborado por el autor

De la misma manera se probó la red con algunos datos para aseverar el correcto entrenamiento de esta, haciendo un archivo con direcciones de las muestras de prueba, de esta manera se comprobó el acierto que se consiguió con la red.

Abrir ▾ redMCSAparametros.prototxt ~/Documentos/vibracionesml/ModeloCaffe

```
name: "CNNMSCA"
layer {
  name: "DatasetMCSA"
  type: "HDF5Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  hdf5_data_param {
    source: "datosentrenamiento.h5list"
    batch_size: 64
    shuffle: true
  }
}
layer {
  name: "DatasetMCSA"
  type: "HDF5Data"
  top: "data"
  top: "label"
  include {
    phase: TEST
  }
  hdf5_data_param {
    source: "datosprueba.h5list"
    batch_size: 100
  }
}
layer {
  name: "poolmax"
  type: "Pooling"
  bottom: "data"
  top: "poolmax"
  pooling_param {
    pool: MAX
    kernel_h: 1
    kernel_w: 5
  }
}
```

Abrir ▾ redMCSAparametros.prototxt ~/Documentos/vibracionesml/ModeloCaffe

```
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "poolmax"
  top: "conv1"
  convolution_param {
    num_output: 596
    kernel_h: 1
    kernel_w: 5
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_h: 1
    kernel_w: 5
    stride: 2
  }
}
```

Abrir ▾ redMCSAparametros.prototxt ~/Documentos/vibracionesml/ModeloCaffe

```
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "pool1"
  top: "conv2"
  convolution_param {
    num_output: 115
    kernel_h: 1
    kernel_w: 5
    stride: 1
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "conv2"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 3
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}
```

Abrir ▾ redMCSAparametros.prototxt ~/Documentos/vibracionesml/ModeloCaffe

```
  name: "ip1"
  type: "InnerProduct"
  bottom: "conv2"
  top: "ip1"
  param {
    lr_mult: 1
  }
  param {
    lr_mult: 2
  }
  inner_product_param {
    num_output: 3
    weight_filler {
      type: "xavier"
    }
    bias_filler {
      type: "constant"
    }
  }
}

layer {
  name: "accuracy"
  type: "Accuracy"
  bottom: "ip1"
  bottom: "label"
  top: "accuracy"
  include {
    phase: TEST
  }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip1"
  bottom: "label"
  top: "loss"
}
```

Fig. 3.28 Archivo contenedor de los parámetros que tomará la red

Fuente: Elaborado por el autor

Para la carga de las muestras fue necesario un archivo de tipo prototxt el cual contendrá la arquitectura de la red, así como se establecerán comandos necesarios para la lectura de archivos h5list y se cargarán los archivos contenedores de las direcciones de las muestras.

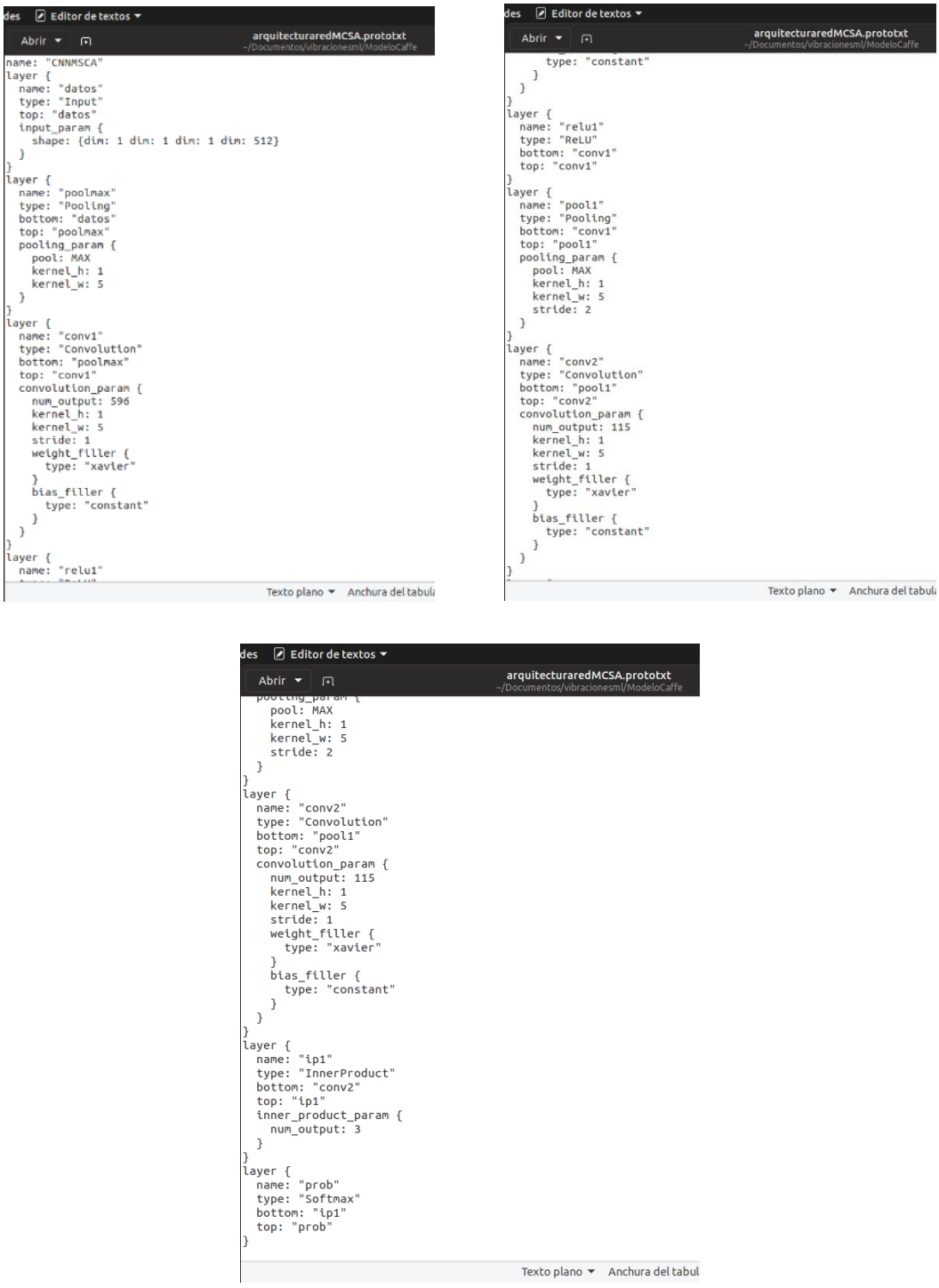

Fig. 3.29 Archivo contenedor de la arquitectura de la red

Fuente: Elaborado por el autor

La arquitectura elegida para la red es creada en un archivo aparte de tipo prototxt este archivo contendrá las partes de la arquitectura con sus características desde las capas de reducción hasta las capas de convolución, este archivo con la arquitectura será cargado para su utilización en cada ejecución.

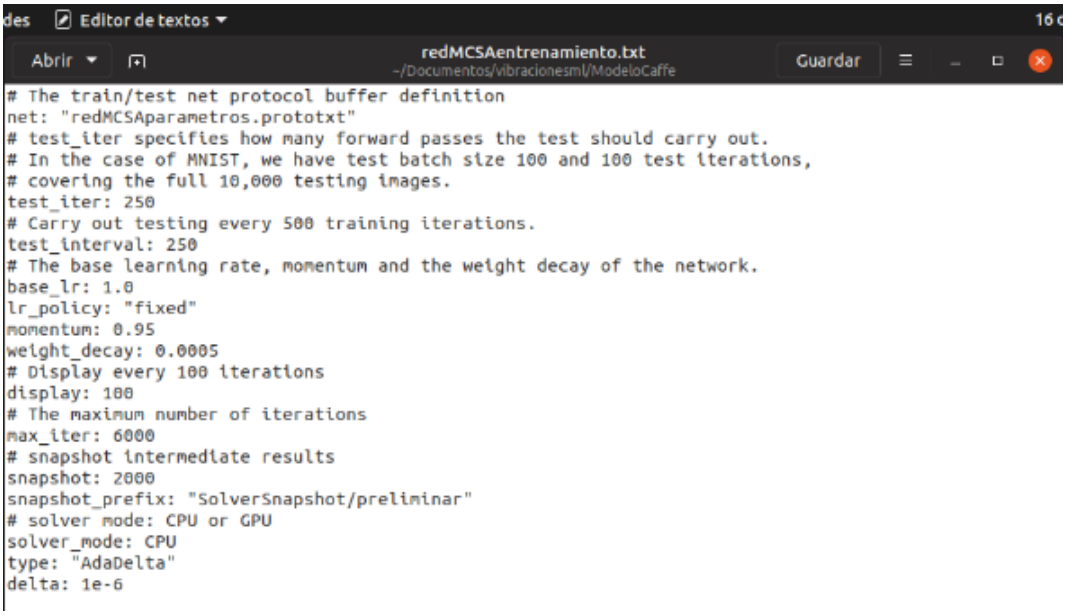

Fig. 3.30 Archivo generador del entrenamiento

Fuente: Elaborado por el autor

Para el entrenamiento de la red se crea un archivo en el cual se especifica cual es el archivo que debe leer para empezar el aprendizaje, en este caso es el contenedor de las direcciones de las muestras, de la misma manera el entrenamiento es guardado en la carpeta para ser utilizado posteriormente.

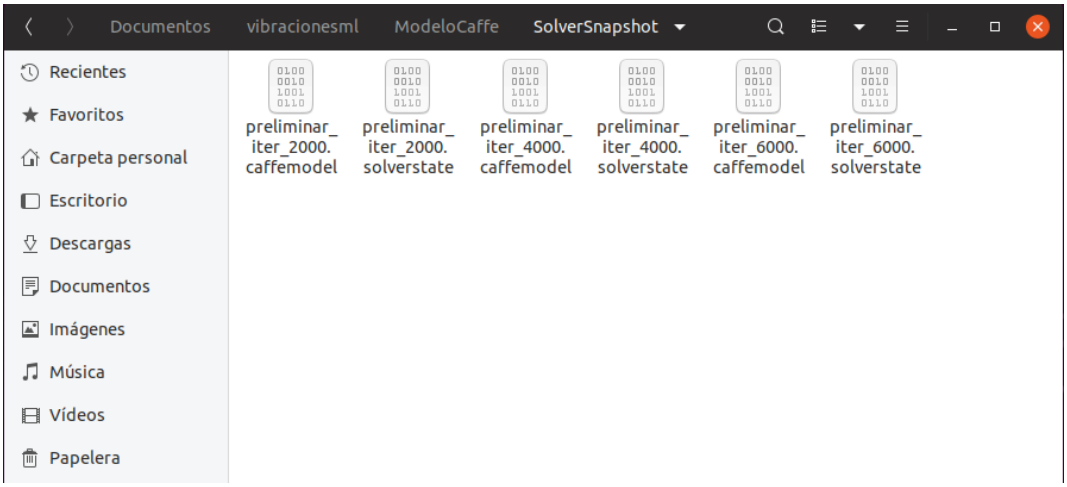

Fig. 3.31 Archivos obtenidos después del entrenamiento

Fuente: Elaborado por el autor

En el proyecto se crea una carpeta la cual guardará los entrenamientos, en este caso para conseguir la mejor precisión se realizó tres entrenamientos. Estos archivos se crean después de realizado el entrenamiento y son de tipo caffemodel y solverstate en los cuales se guardan las características de cada clase a analizarse para este caso particular se guardan las características tanto del funcionamiento normal como del funcionamiento con presencia de fallos. El archivo que contiene las características se usará al cargar la red para no tener que realizar el entrenamiento en cada ejecución.

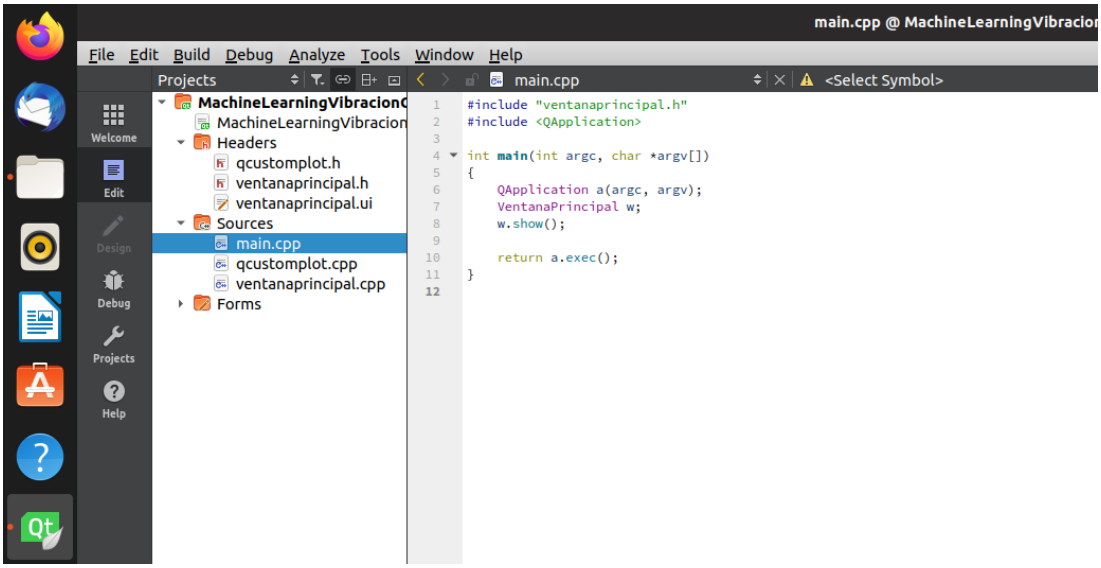

Fig. 3.32 Ejecución del proyecto desarrollada en Qt creator

Fuente: Elaborado por el autor

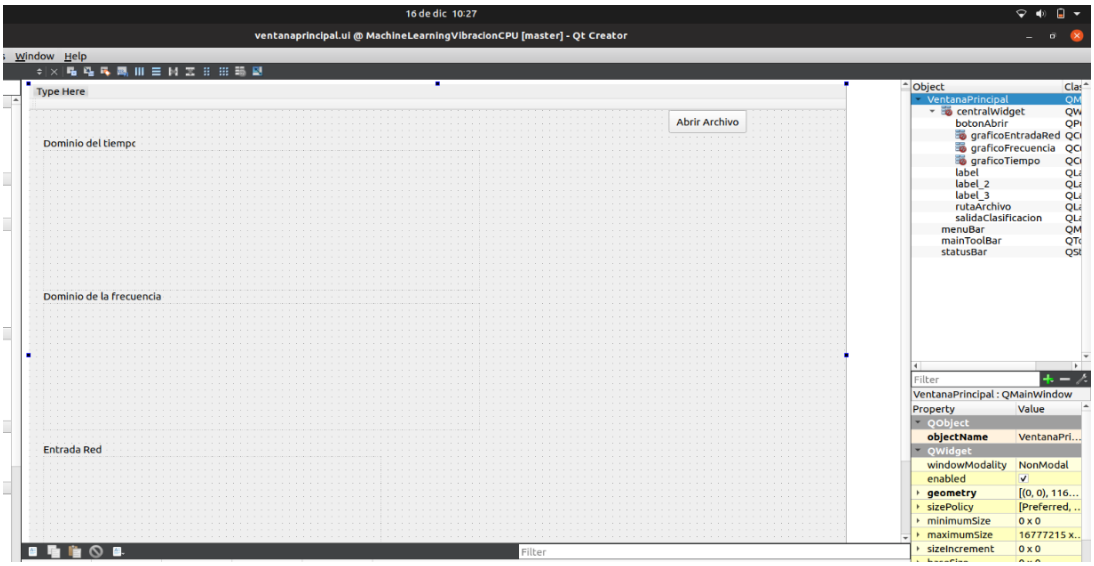

Fig. 3.33 Interfaz a ser visualizada por el usuario

Fuente: Elaborado por el autor

```
#ifndef VENTANAPRINCIPAL_H
#define VENTANAPRINCIPAL_H

#include <QMainWindow>
#include <QFile>
#include <QFileDialog> //Para crear los dialogos de obtener datos
#include <QDir> //Obtiene las direcciones absolutas de la carpeta principal
#include <QFile> //Libreria para obtener archivos de datos
#include <qcustomplot.h>
#include <eigen3/unsupported/Eigen/FFT> //Transformada rápida de Fourier
#define CPU_ONLY 1
#include <caffe/caffe.hpp> //Incluye Caffe, libreria de Deep Learning
#include <caffe/layers/memory_data_layer.hpp>
#include <string>
```

Fig. 3.34 Librerías utilizadas para la creación de la ventana principal

Fuente: Elaborado por el autor

```
namespace Ui {
class VentanaPrincipal;
}

class VentanaPrincipal : public QMainWindow
{
    Q_OBJECT

public:
    explicit VentanaPrincipal(QWidget *parent = 0);
    ~VentanaPrincipal();
    QVector<QCPGraphData> valoresGrafico, datosEntradaRed, valoresGraficoFrecuencia;
    QCPGraphData valorGraficounico,valorGraficoFrecuenciaunico, datoEntradaRed;
    std::vector<std::string> nombres;

private slots:
    void on_botonAbrir_clicked();

private:
    Ui::VentanaPrincipal *ui;
    std::shared_ptr<caffe::Net<float> > red_;
    int numerodatosentrada = 512;

    std::ifstream nombresdesdearchivo;
    std::string linea;
    std::vector<std::pair<float, std::string> > paresclasificacion;
};

#endif // VENTANAPRINCIPAL_H
```

Fig. 3.35 Código declarador de variables

Fuente: Elaborado por el autor

Los métodos para controlar el funcionamiento de la red, así como el necesario para graficar las muestras están desarrollados en una clase aparte y son solo referenciados, en esta parte del código se encuentra la declaración de todas las variables que se utilizarán y del evento del único botón que se utilizará.

```cpp
#include "ventanaprincipal.h"
#include "ui_ventanaprincipal.h"

VentanaPrincipal::VentanaPrincipal(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::VentanaPrincipal)
{
    ui->setupUi(this);
#ifdef CPU_ONLY
    caffe::Caffe::set_mode(caffe::Caffe::CPU);
#else
    caffe::Caffe::set_mode(caffe::Caffe::GPU);
#endif
    red_.reset(new caffe::Net<float>("../ModeloCaffe/arquitecturaredMCSA.prototxt", caffe::TEST));
    red_->CopyTrainedLayersFrom("../ModeloCaffe/redMCSAmodelo.caffemodel");
    numerodatosentrada = red_->input_blobs()[0]->shape(3);
    nombresdesdearchivo.open("../ModeloCaffe/nombres.txt");

    while (std::getline(nombresdesdearchivo, linea)){
        nombres.push_back(std::string(linea));
        qDebug() << QString::fromStdString(linea);
    }
    ui->graficoTiempo->addGraph();
    ui->graficoTiempo->xAxis->setLabel("Tiempo");
    ui->graficoTiempo->yAxis->setLabel("Valor");
    ui->graficoEntradaRed->addGraph();
    ui->graficoEntradaRed->yAxis->setRange(-1,1);
    ui->graficoFrecuencia->addGraph();
    ui->graficoFrecuencia->xAxis->setLabel("Frecuencia");
    ui->graficoFrecuencia->yAxis->setLabel("Módulo");
}

VentanaPrincipal::~VentanaPrincipal()
{
    delete ui;
}
```

Fig. 3.36 Código para cargar información de la red

Fuente: Elaborado por el autor

En esta clase se encuentran los métodos utilizados en todo el proyecto, entre ellos está el calcular los valores para obtener la gráfica de la muestra en base al tiempo, así como la carga de las características de la red y de los archivos necesarios para el correcto funcionamiento de la misma. En esta parte del código es cargado a la red el archivo contenedor de la arquitectura de la red. Posteriormente es cargado el archivo "redMCSAmodelo.caffemodel" el cual es un archivo que se obtiene después del entrenamiento, el cual contendrá los pesos de cada neurona. Finalmente se carga el archivo con los nombres de los fallos el cual se utilizará para las salidas.

```
void VentanaPrincipal::on_botonAbrir_clicked()
{
    QString ruta =  QFileDialog::getOpenFileName(this, "Abrir el archivo que contiene los datos de vibración", QDir::homePath(), "Archivo CSV (*.csv)");
    ui->rutaArchivo->setText(ruta);

    QFile archivocsv(ruta);
    ui->graficoTiempo->setLocale(QLocale(QLocale::Spanish, QLocale::Ecuador));
    std::vector<float> vectortiempo;
    std::vector<std::complex<float> > vectorfrecuencia;
    float frecuencia = 0;
    double valormenory = 1000000, valormayory = 0;
    valoresGrafico.clear();
    valoresGraficoFrecuencia.clear();
    datosEntradaRed.clear();
    if (archivocsv.open(QIODevice::ReadOnly))
    {
        QTextStream in(&archivocsv);
        while (!in.atEnd())
        {
            QString line = in.readLine();
            valorGraficounico.value = line.split(',').first().toDouble();
            QDateTime tiempo = QDateTime::fromString(line.split(',').back(),Qt::TextDate);
            valorGraficounico.key = vectortiempo.size(); //tiempo.toMSecsSinceEpoch();
            if(valorGraficounico.value < valormenory) valormenory = valorGraficounico.value;
            if(valorGraficounico.value > valormayory) valormayory = valorGraficounico.value;
            vectortiempo.push_back(valorGraficounico.value);      implicit conversion loses floating-point precision: 'double' to 'std::vector<float,
            valoresGrafico.append(valorGraficounico);
        }
        archivocsv.close();
    }
```

Fig. 3.37 Código del evento botonAbrir (leer archivo)

Fuente: Elaborado por el autor

En esta parte del código se encuentra el funcionamiento del botón el cual realiza varias acciones, al momento de su activación se lee un archivo en formato csv el cual contendrá los datos de la onda que se va a analizar, posteriormente se preprocesa y procesa con el propósito de que el archivo se quede con 512 puntos con los cuales trabajara la red.

```
float tiempo = (valoresGrafico.last().key - valoresGrafico.first().key)/1000;                                    implicit conversion loses floating-
frecuencia = 1 /(tiempo/valoresGrafico.size()) ;
Eigen::FFT<float> fft;
fft.fwd(vectorfrecuencia, vectortiempo);

double valormenorx = 1000000, valormayorx = 0;
for(size_t i = 0; i < vectorfrecuencia.size(); i++){
    valorGraficoFrecuenciaunico.key = i*frecuencia/vectorfrecuencia.size();                                      implicit conversion increases floating-
    valorGraficoFrecuenciaunico.value = 20*log(std::abs(vectorfrecuencia.at(i))/vectorfrecuencia.size()); //Change to power spectrum values    implicit conversi
    if(valorGraficoFrecuenciaunico.value < valormenorx) valormenorx = valorGraficoFrecuenciaunico.value;
    if(valorGraficoFrecuenciaunico.value > valormayorx) valormayorx = valorGraficoFrecuenciaunico.value;
    valoresGraficoFrecuencia.append(valorGraficoFrecuenciaunico);
}


ui->graficoTiempo->graph()->data()->set(valoresGrafico);
ui->graficoTiempo->xAxis->setRange(valoresGrafico.first().key, valoresGrafico.last().key);
ui->graficoTiempo->yAxis->setRange(valormenory - abs(valormenory - valormayory)/50.0, valormayory + abs(valormenory - valormayory)/10.0);
ui->graficoTiempo->replot();

// set axes ranges, so we see all data:

ui->graficoFrecuencia->graph()->data()->set(valoresGraficoFrecuencia);
ui->graficoFrecuencia->xAxis->setRange(valoresGraficoFrecuencia.first().key - abs(valoresGraficoFrecuencia.first().key - valoresGraficoFrecuencia.last().key)/10.0
ui->graficoFrecuencia->yAxis->setRange(valormenorx - abs(valormenorx - valormayorx)/50.0, valormayorx + abs(valormenorx - valormayorx)/10.0);
ui->graficoFrecuencia->replot();
```

Fig. 3.38 Código del evento botonAbrir (aplicación FFT y gráfica)

Fuente: Elaborado por el autor

Después de procesar la muestra y obtener los 512 puntos se realiza el FFT el cual es posible utilizar de manera automática debido al uso de la librería Eigen, lo obtenido es normalizado y escalado para poder ser graficado, de esta manera se gráfica la onda en el dominio del tiempo y el dominio de la frecuencia.

```
//AÑADOR ENTRADA DE RED
bool calcular = true;
if(vectorfrecuencia.size() > numerodatosentrada){                    comparison of integers of different signs: 'std::vec
    qDebug() << "Muy grande" << endl;
    //QMessageBox::StandardButton aceptar = QMessageBox::Information(this, "Información","Vector muy grande que al de entrada de red, reduciendo vector
}
else if(vectorfrecuencia.size() < numerodatosentrada){               comparison of integers of different signs: 'std::vec
    //QMessageBox::StandardButton aceptar = QMessageBox::Information(this, "Información","Vector muy pequeño, no se puede calcular, ingrese otro archivo
    calcular = false;
}

if(calcular){
    red_->ClearParamDiffs();
    caffe::Blob<float> *blob_entrada = red_->input_blobs()[0];
    float* vectorEntradaRed = blob_entrada->mutable_cpu_data();

    float maximo = -10000000000000;
    float minimo = 0;
    for(size_t j = 0; j < valoresGraficoFrecuencia.size(); j++){          comparison of integers of different
        if(maximo < valoresGraficoFrecuencia.at(j).value) maximo = valoresGraficoFrecuencia.at(j).value;    implicit conversion increases floating-poi
        if(minimo > valoresGraficoFrecuencia.at(j).value) minimo = valoresGraficoFrecuencia.at(j).value;    implicit conversion increases floating-poi
    }

    float limiteinferior = -1, limitesuperior = 1;;
    for(size_t j = 0; j < valoresGraficoFrecuencia.size(); j++){          comparison of integers of different
        datoEntradaRed.value = limiteinferior + ((valoresGraficoFrecuencia.at(j).value - minimo)*(limitesuperior - limiteinferior))/(maximo - minimo);
        datoEntradaRed.key = j;
        datosEntradaRed.push_back(datoEntradaRed);
        vectorEntradaRed[j] = datoEntradaRed.value;                                                    implicit conversion lose
    }
    ui->graficoEntradaRed->xAxis->setRange(0,numerodatosentrada);
    ui->graficoEntradaRed->graph()->data()->set(datosEntradaRed);
    ui->graficoEntradaRed->replot();
```

Fig. 3.39 Código del evento botonAbrir (selección y gráfica de la entrada)

Fuente: Elaborado por el autor

En esta parte del método se define las entradas a la red y a su vez se crea un vector para guardar los elementos que serán analizados posteriormente, estos elementos también son graficados en pantalla para su visualización.

```
red_->Forward();
boost::shared_ptr<caffe::Blob<float> > capa_salida = red_->blob_by_name("prob");
const float* begin = capa_salida->cpu_data();
const float* end = begin + capa_salida->channels();

std::vector<float> resultados(begin, end);
```

Fig. 4.40 Código del evento botonAbrir (ejecución de la red)

Fuente: Elaborado por el autor

El comando necesario para la ejecución de la red es Forward. Este comando es utilizado para inicializar el análisis de la muestra. Para obtener los datos de la capa de salida se requiere de un vector él cual contendrá las probabilidades de que sea alguna de las clases analizadas, estos valores serán tomados posteriormente para determinar a qué clase pertenece y de esta manera reflejar predicción.

```
    std::vector<float> resultados(begin, end);


    //Código red
    paresclasificacion.clear();
    for(size_t i = 0; i < resultados.size(); i++){
        paresclasificacion.push_back(std::make_pair(resultados.at(i),nombres.at(i)));
    }
    qDebug() << "Salida: Probabilidad: " << paresclasificacion.at(0).first << " Clase: " << QString::fromStdString(paresclasificacion.at(0).second);
    qDebug() << "Salida: Probabilidad: " << paresclasificacion.at(1).first << " Clase: " << QString::fromStdString(paresclasificacion.at(1).second);
    qDebug() << "Salida: Probabilidad: " << paresclasificacion.at(2).first << " Clase: " << QString::fromStdString(paresclasificacion.at(2).second);
    std::sort(paresclasificacion.begin(),paresclasificacion.end(), [](const std::pair<float, std::string> a, const std::pair<float, std::string> b) {
        return a.first > b.first;
    } );
    ui->salidaClasificacion->setText(QString("Es ") + QString::fromStdString(paresclasificacion.at(0).second) + " con un " + QString::number(paresclasificacion.at(0).first*100)
    qDebug() << "Salida: Probabilidad: " << paresclasificacion.at(0).first << " Clase: " << QString::fromStdString(paresclasificacion.at(0).second);
}
```

Fig. 3.41 Código del evento botonAbrir (Muestra del resultado obtenido)

Fuente: Elaborado por el autor

Finalmente, para culminar el proceso los valores obtenidos de las probabilidades seran almacenados en un vector. Posteriormente se itera el vector para emparejar los resultados con el nombre de la clase a la que pertenece, tanto las probabilidades de cada clase como la selección de la mayor son mostradas por consola, así como por la interfaz.

3.2.7 Descripción de los entrenamientos de la red con diversas muestras.

Los entrenamientos de la red fueron varios usando diferentes cantidades de muestras, utilizando varias muestras se llegó a determinar el número necesario para que la red llegue a tener una precisión aceptable.

Se realizó varios entrenamientos con diversas cantidades de datos para definir lo necesarios para conseguir un entrenamiento completo. El entrenamiento realizado con alrededor de 2500 muestras y se realizó 800 pruebas con lo cual se determinó que la red obtuvo una precisión del 23% lo cual demostró que es necesaria una muestra más grande. El segundo entrenamiento se lo realizó con 40000 muestras y 8000 pruebas aquí el resultado obtenido fue de un 84% de precisión el cual es un entrenamiento aceptable.

```
I1102 08:26:19.276559  9185 solver.cpp:397]     Test net output #0: accuracy = 0.8399
I1102 08:26:19.276605  9185 solver.cpp:397]     Test net output #1: loss = 0.404513 (* 1 = 0.404513 loss)
I1102 08:26:19.276613  9185 solver.cpp:315] Optimization Done.
I1102 08:26:19.276633  9185 caffe.cpp:259] Optimization Done.
```

Fig. 3.42 Resultado del entrenamiento con 40000 muestras y 8000 pruebas

Fuente: Elaborado por el autor

El último entrenamiento realizado fue con el que se consiguió una precisión absoluta la cual es del 97.6% este entrenamiento se realizó con 64000 muestras, esta precisión brinda la seguridad de acertar con el fallo que se analice, en cualquier caso.

```
I1110 13:10:50.968523 26269 solver.cpp:310] Iteration 6000, loss = 0.0028552
I1110 13:10:50.968549 26269 solver.cpp:330] Iteration 6000, Testing net (#0)
I1110 13:11:08.459336 26269 solver.cpp:397]     Test net output #0: accuracy = 1
I1110 13:11:08.459367 26269 solver.cpp:397]     Test net output #1: loss = 0.00473697 (* 1 = 0.00473697 loss)
I1110 13:11:08.459372 26269 solver.cpp:315] Optimization Done.
```

Fig. 3.43 Resultado del entrenamiento con 64000 muestras

Fuente: Elaborado por el autor

Para el entrenamiento de la red es necesario la creación de una carpeta la cual guardará el resultado del entrenamiento realizado, por esta razón el entrenamiento se lo realiza una sola vez al inicio del programa y después solo se precargará el entrenamiento en cada ejecución del programa.

Para graficar la evolución de los datos durante el entrenamiento. Caffe incluye 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 se requiere especificar el parámetro 2. 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.

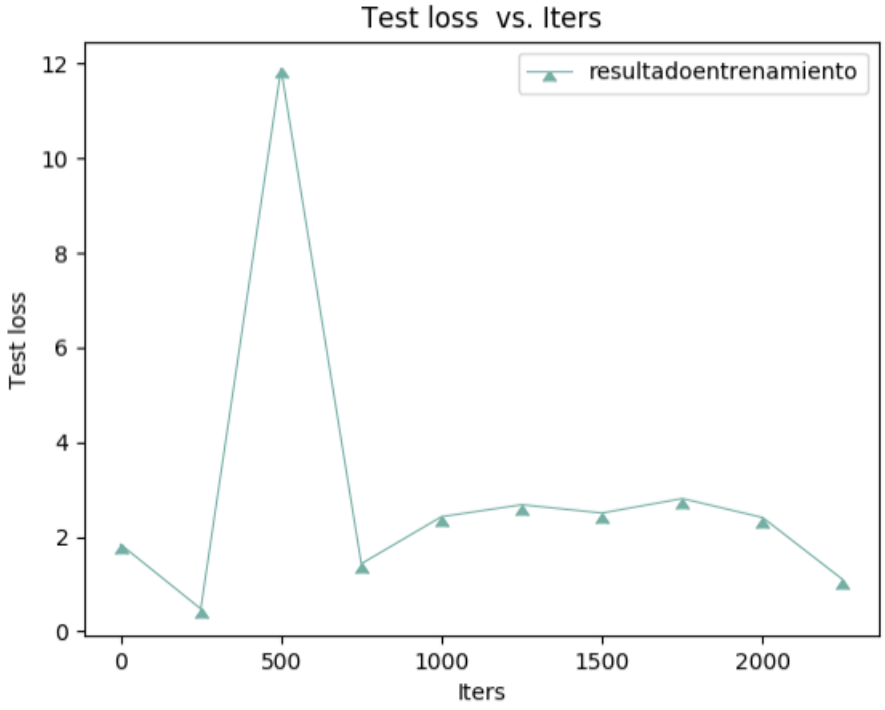

Fig. 3.44 Gráfica de función Loos en base a las iteraciones

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.

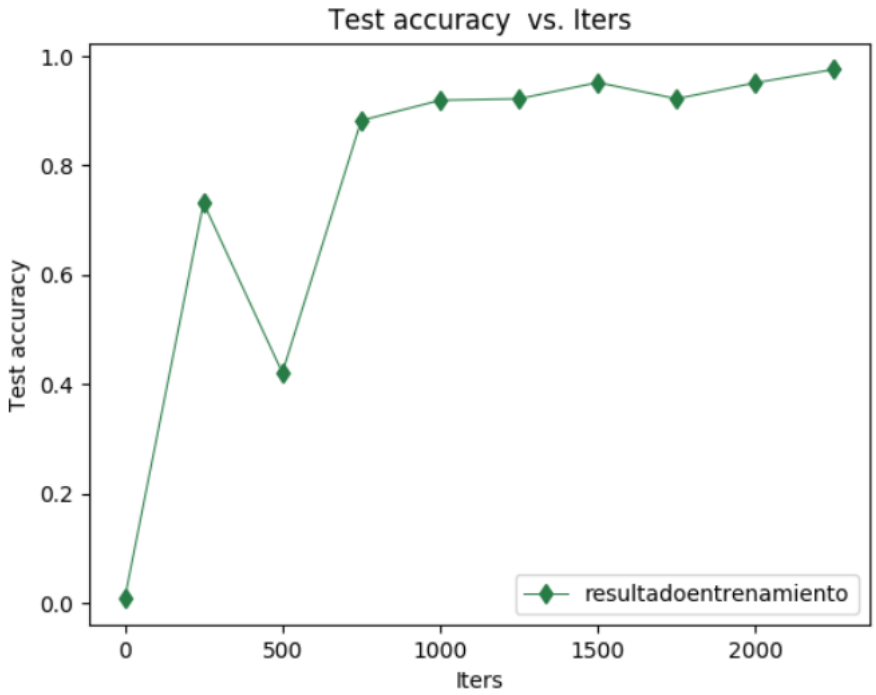

Fig. 3.45 Gráfica de precisión en base a las iteraciones
Fuente: Elaborado por el autor

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 convolucional fue del 97.6%, esto se consiguió con 2250 iteraciones.

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

La evaluación de la red se la realizó con diversas muestras para comprobar su correcto funcionamiento, las muestras fueron clasificadas con respecto a sus clases para realizar una comprobación ordenada.

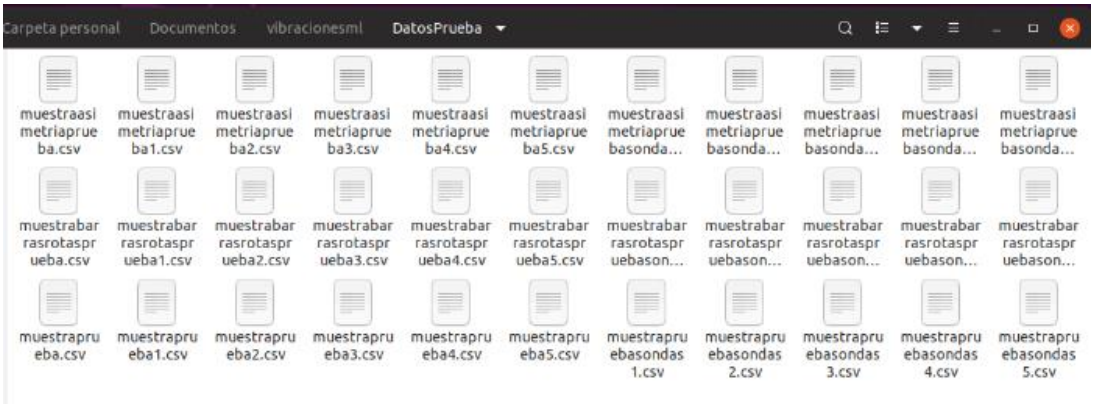

Fig. 3.46 Diferentes muestras obtenidas para pruebas de la red

Fuente: Elaborado por el autor

Se obtuvieron en total 33 muestras para la comprobación, todas ellas en el formato requerido para poder ser analizadas.

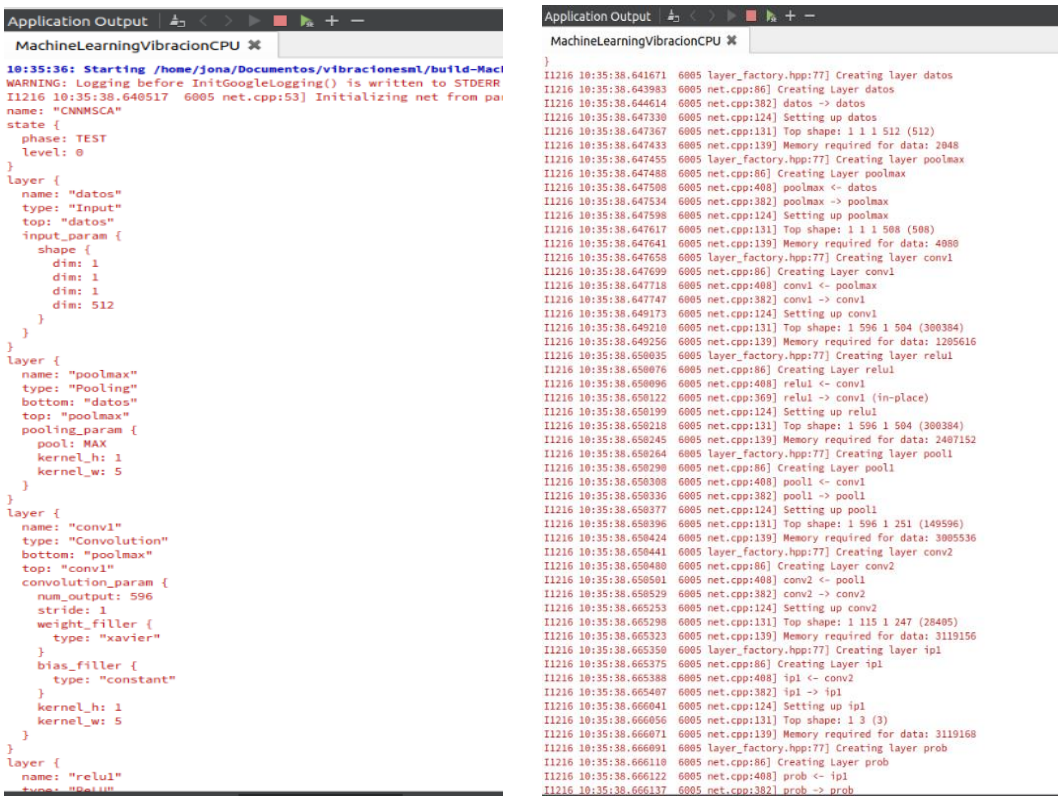

Fig. 3.47 Ejecución del proyecto en consola

Fuente: Elaborado por el autor

Al momento de ejecutar el proyecto en consola se puede visualisar como se realiza la carga de la arquitectura y como se va realizando la configuración de la misma y la carga del entrenamiento realizado anteriormente.

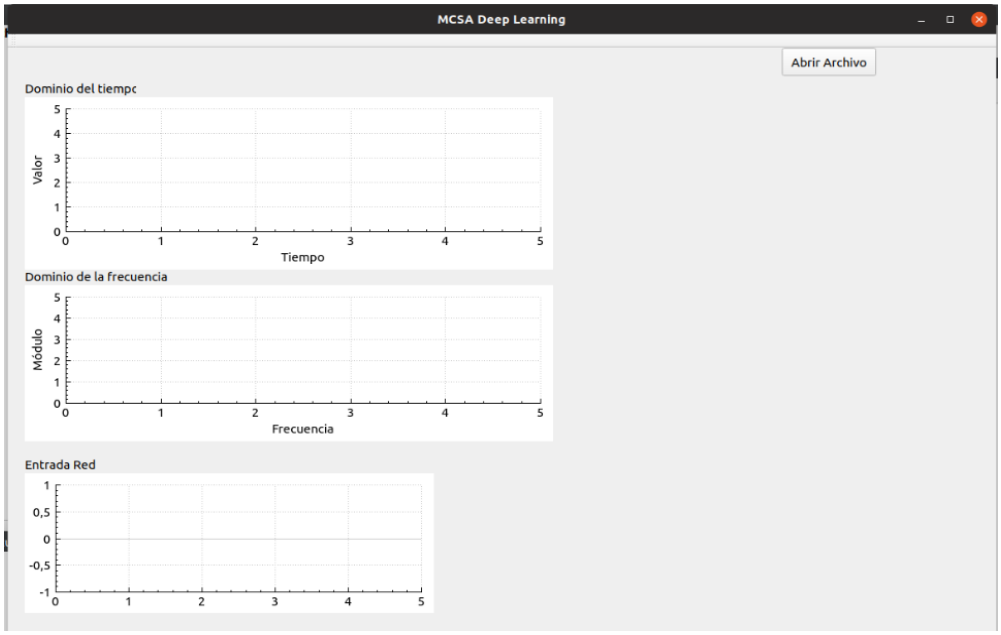

Fig. 3.48 Ejecución de la interfaz del proyecto

Fuente: Elaborado por el autor

El proyecto no posee una gran interfaz debido a que no es necesario, la interfaz posee las 3 graficas que serán cargadas al momento de cargar la muestra, el botón abrir archivo es el que inicia todo el proyecto.

```
I1216 10:35:38.821230  6005 net.cpp:746] Ignoring source layer loss
"normal"
"BarrasRotas"
"AsimetriaEje"
Salida: Probabilidad:  0.999999  Clase:  "normal"
Salida: Probabilidad:  8.56474e-42  Clase:  "BarrasRotas"
Salida: Probabilidad:  9.68769e-07  Clase:  "AsimetriaEje"
Salida: Probabilidad:  0.999999  Clase:  "normal"
```

Fig. 3.49 Salida del proyecto después de una ejecución

Fuente: Elaborado por el autor

La salida del proyecto se ve reflejada tanto por consola como de manera visual, la salida por consola nos permite visualizar todas las salidas obtenidas por la red y la que es elegida la probabilidad más alta.

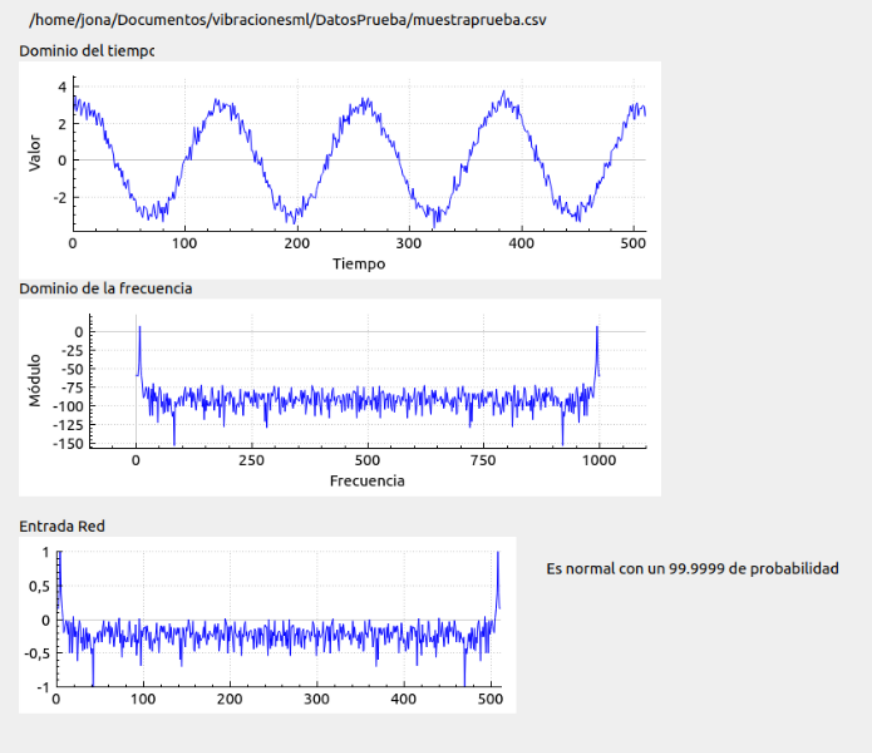

Fig. 3.50 Salida del proyecto en interfaz

Fuente: Elaborado por el autor

La interfaz en cambio al momento de elegir la muestra nos gráfica la onda en dominio del tiempo de la frecuencia y la onda que va a entrar en la red y nos mostrara como un mensaje la probabilidad más alta a la que se asemeje la muestra, el proyecto puede analizar un sinfín de muestras en una sola ejecución.

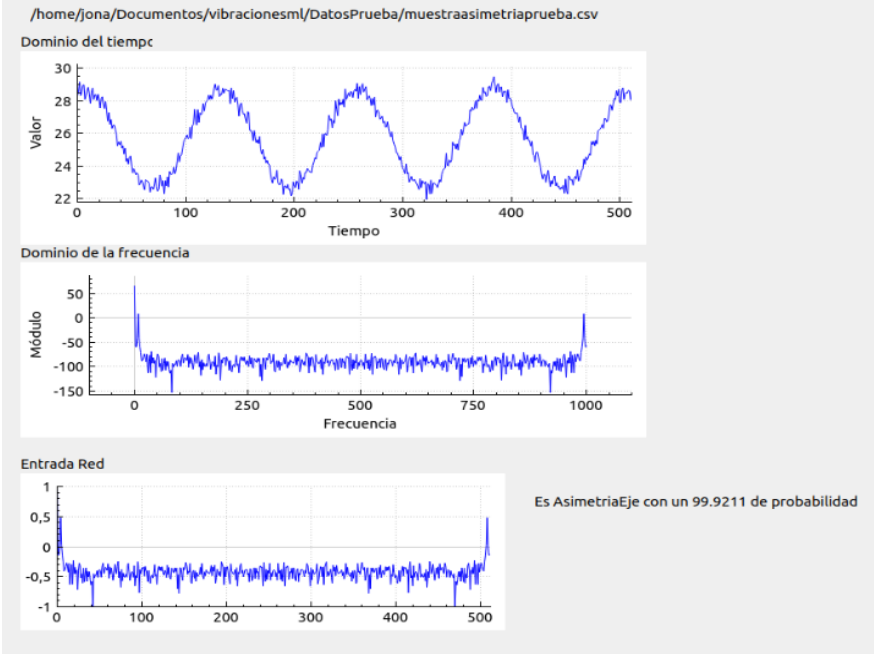

Fig. 3.51 Salida con fallo de asimetría de eje

Fuente: Elaborado por el autor

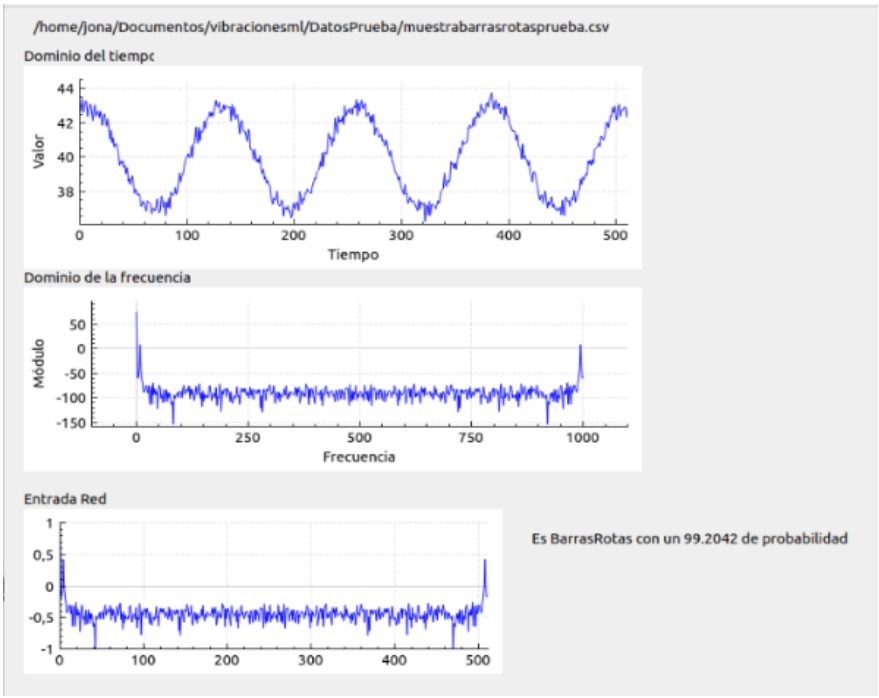

Fig. 3.52 Salida con fallo de barras rotas

Fuente: Elaborado por el autor

En perspectiva las gráficas son iguales en cada muestra, esto es debido a que por el tamaño de la gráfica no se puede percibir las diferencias que tienen entre sí.

Pruebas

Pruebas de aceptación

Prueba de aceptación	
Numero: 1	**Historia de usuario:** Dataset para entrenamiento
Nombre: Diseño de Dataset para entrenamiento	
Descripción: La red neuronal convolucional requiere de gran cantidad de muestras para su entrenamiento razón por la cual se desea generar un dataset con muestras de funcionamientos de un motor en normalidad, así como bajo la influencia de fallos para alcanzar un nivel alto de acierto en la predicción.	
Condiciones de ejecución: Ser el investigador	
Entrada: -	
Resultado esperado: Dado que se requiere una estructura especifica para el manejo de las muestras, es necesario la investigación de una librería que permita el manejo de datos similar al de un blob de Caffe. Para manejar las muestras que serán simuladas, dentro de la red y poder clasificarlas.	
Evaluación de la prueba: Prueba Satisfactoria	

Tabla 3.31 Prueba de aceptación 1 – Dataset para entrenamiento

Fuente: Elaborado por el autor

Prueba de aceptación	
Numero: 2	**Historia de usuario:** Generador de muestras
Nombre: Desarrollo de generador de muestras	
Descripción: Teniendo en cuenta la necesidad de una gran cantidad de muestras y la dificultad para conseguirlas, se requiere un pequeño sistema capaz de generar muestras tanto de funcionamiento normal como con presencia de fallos, esto con el uso de algoritmos, de esta manera se logrará conseguir muestras similares a las reales.	
Condiciones de ejecución: Desarrollar el aplicativo para generar muestras	
Entrada: El usuario debe acceder a la ubicación del generador, para ingresar el comando de ejecución.	

Resultado esperado: Se inicia la generación y se va mostrando en pantalla el proceso que realiza hasta culminar con el mensaje de muestras generadas.
Evaluación de la prueba: Prueba satisfactoria

Tabla 3.32 Prueba de aceptación 2 – Generador de muestras

Fuente: Elaborado por el autor

Prueba de aceptación	
Numero: 3	**Historia de usuario:** Selección de la arquitectura
Nombre: Investigación	
Descripción: Se requiere una arquitectura con las características para agilitar la clasificación de fallos, debido a que por medio una buena elección el clasificador será optimo, para la selección se requiere de una arquitectura probada.	
Condiciones de ejecución: Ser el investigador	
Entrada: -	
Resultado esperado: Dado que se requiere una arquitectura que permita optimizar el funcionamiento de la red, cuando se realice la clasificación de una muestra. La arquitectura tiene que ser validada por medio de un estudio en artículos científicos.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 3.33 Prueba de aceptación 3 – Selección de la arquitectura

Fuente: Elaborado por el autor

Prueba de aceptación	
Numero: 4	**Historia de usuario:** Desarrollo de la red neuronal convolucional
Nombre: Desarrollo de la red neuronal convolucional	
Descripción: El desarrollo de la red debe seguir la arquitectura elegida, debido a que de esta manera se conseguirá reducir el procesamiento en la clasificación y brindará velocidad al análisis, para lograr un funcionamiento rápido y ligero del programa.	
Condiciones de ejecución: Desarrollar la red neuronal convolucional	
Entrada: El usuario debe ingresar una muestra	
Resultado esperado: Con la aplicación iniciada se realiza la carga de una muestra, cuando es ingresada la muestra empieza a ser analizada por la red neuronal,	

entonces una vez finalizada la clasificación se presentan los resultados a modo de probabilidades.
Evaluación de la prueba: Prueba satisfactoria.

Tabla 3.34 Prueba de aceptación 4 – Desarrollo de la red neuronal convolucional

Fuente: Elaborado por el autor

Prueba de aceptación	
Numero: 5	**Historia de usuario:** Desarrollo de interfaz
Nombre: Desarrollo de interfaz	
Descripción: El proyecto requiere de un interfaz simple la cual permitirá elegir una muestra para analizarse y a su vez permitirá visualizar la muestra graficada y la sección que se tomara para el análisis, debido a que lo importante del proyecto es el análisis de los datos para la obtención de una pronta predicción.	
Condiciones de ejecución: Haber diseñado la interfaz de usuario	
Entrada: El usuario abre la aplicación y selecciona una muestra	
Resultado esperado: Se inicia la aplicación, después de seleccionada la muestra se gráfica la onda para su visualización y una vez terminado el proceso de clasificación se visualiza la predicción.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 3.35 Prueba de aceptación 5 – Desarrollo de interfaz

Fuente: Elaborado por el autor

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Existe una amplia recopilación de información con respecto a las redes neuronales convolucionales. El uso de este tipo de redes neuronales se encuentra en crecimiento, aunque a nivel nacional no existe gran utilización de estas.
- Las redes neuronales convolucionales son utilizadas para la clasificación de imágenes, razón por la cual son utilizadas para reconocimientos de grandes volúmenes de imágenes con el propósito de identificar pequeñas características comunes, razón por la cual son ideales para la clasificación de ondas de corriente.
- A nivel industrial maquinarias rotativas son utilizadas para la producción, razón por la cual, en el momento que sufren fallos graves que ocasionan su detención las industrias detienen su producción lo que ocasiona grandes pérdidas.
- Las maquinarias rotativas se ven expuestas a diversos tipos de fallos debido a una mala instalación o un mal mantenimiento, los fallos corregidos a tiempo pueden evitar que se agraven, así como el paro de las máquinas
- Las redes neuronales convolucionales son grandes identificadores de características, pero para mejorar su precisión es necesaria una gran cantidad de muestras para su entrenamiento, esto garantiza que la red pueda catalogar de manera ideal todas las imágenes a analizar.

4.2 Recomendaciones

- Para el desarrollo de una red neuronal convolucional es recomendable investigar las arquitecturas existentes que han sido probadas con anterioridad para tener la seguridad de que la red cumplirá con su propósito si no se tiene tiempo para realizar pruebas de funcionamiento.
- Se recomienda usar repositorios internacionales en inglés para la búsqueda de mejor información, debido a que en ellos encuentra explicaciones más detallada y de mayor utilidad sobre su uso y funcionamiento.
- Se recomienda el uso de caffe para el desarrollo de redes neuronales convolucionales, en vista que posee librerías que facilitan su desarrollo y su utilización.
- Es recomendable la utilización de este tipo de tecnologías para realizar mantenimientos predictivos de maquinaria rotativa, ya que con su funcionalidad se conseguiría optimizar recursos y reducir los tiempos de parada de la producción.
- Es recomendable utilizar capas de reducción de muestra, para hacer que la clasificación sea más rápida, esto es una característica de las redes neuronales convoluciones, lo que se consigue es ir verificando similitudes por sectores para reducir el procesamiento.
- Es recomendable utilizar una gran muestra de datos para el entrenamiento, la importancia radica en que a mayor entrenamiento se obtendrá una mayor precisión al momento de la clasificación.
- Para el entrenamiento se recomienda datos reales, pero por la dificultad de obtención de grandes muestras se puede utilizar fórmulas que simulan los fallos, una vez simulado es recomendable la adición de ruido blanco a las ondas para asemejar los fallos reales.

Bibliografías

[1] M. E. Arias Infante, «Control de la gesticulación de un robot socialinteractivo con aspecto humanoide», 2018.

[2] T. D. E. Titulación, P. A. La, O. Del, T. D. E. Ingeniero, y E. N. Electrónica, «Reconocimiento de imágenes en frames de video utilizando redes neuronales», 2016.

[3] M. S. Alta Alta, «Desarrollo de un sistema de visión artificial para detectar automóviles estacionados en lugares no permitidos», 2018.

[4] J. A. Redondo Martín, «Google Imágenes y el machine learning que todo lo ve», *IMF Business School*, 2017. [En línea]. Disponible en: https://blogs.imf-formacion.com/blog/tecnologia/google-imagenes-cerebro-electronico-lo-ve-201706/.

[5] I. Diego y C. M. Sc, «Procesamiento de imágenes mediante el uso de un vehículo aéreo no tripulado (uav) para el monitoreo y detección de incendios forestales», 2016.

[6] L. J. Aguiar y H. A. Rodriguez, «Análisis de modos y efectos de falla para mejorar la disponibilidad operacional en la línea de producción de gaseosas No3», p. 300, 2014.

[7] L. J. Aguiar y H. A. Rodriguez, «Análisis de modos y efectos de falla para mejorar la disponibilidad operacional en la línea de producción de gaseosas No3», 2014.

[8] J. Antonino-daviu, M. Rubbiolo, y A. Q. López, «Advanced analysis of motor currents for the diagnosis of the rotor condition in electric motors operating in mining facilities», vol. 9994, n.º c, 2018, doi: 10.1109/TIA.2018.2818671.

[9] W. Oñate, R. Pérez, y G. Caiza, «Diagnóstico de fallas incipientes en motores de inducción mediante MCSA», pp. 1-13.

[10] O. Oswaldo y R. Contero, «Diagnóstico de fallas incipientes en motores de inducción por MCSA utilizando la transformada de Hilber», 2016.

[11] M. P. De Alcázar, «Inteligencia artificial para el bien en el mundo», 2017.

[12] MathWorks, «Deep Learning», *MathWorks*, 2019. [En línea]. Disponible en: https://la.mathworks.com/discovery/deep-learning.html.

[13] S. G. De Deloitte, «Inteligencia de máquina», 2017.

[14] S. Sumathi, L. Kumar Ashok, y P. Surekha, *Computational Intelligence Paradigms for Optimization Problems Using MATLAB/SIMULINK*, 1st ed. 2015.

[15] R. Benítez, G. Escudero, y S. Kannan, *Inteligencia Artificial Avanzada*, 1st ed. 2014.

[16] S. Khan, H. Rahmani, S. Afaq, A. Shah, y M. Bennamoun, *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool Publishers, 2018.

[17] M. Lopez y W. Yu Liu, «Identificación de sistemas no lineales mediante redes neuronales convolucionales», 2017.

[18] G. Nikou, «Predicting Defective Engines using Convolutional Neural Networks on Temporal Vibration Signals», n.º 3, pp. 92-102, 2017.

[19] E. Shelhamer y J. Yangqing, «Blobs, Layers, and Nets: anatomy of a Caffe model», *Caffe*. [En línea]. Disponible en: https://caffe.berkeleyvision.org/tutorial/net_layer_blob.html.

[20] H. Qiu, J. Lee, J. Lin, y G. Yu, «Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics», *J. Sound Vib.*, vol. 289, n.º 4-5, pp. 1066-1090, doi: 10.1016/j.jsv.2005.03.007.

[21] D. E. I. E. N. Sistemas y E. E. Industrial, «Sistema open source con conexión a la plataforma.», 2019.

[22] E. D. Gamboa Teneta, «Prototipo de un chatbot para compras online utilizando bot framework.», 2019.

[23] D. X. Anrrango Benavides, «Sistema web para la gestión de matrículas y calificaciones de la unidad educativa fiscomisional fray bartolomé de las casas salasaca», 2020.

[24] M. Cerna y A. F. Harvey, «The Fundamentals of FFT-Based Signal Analysis and Measurement», n.º July, pp. 1-20.