Detección de fallas en motores trifásicos de inducción utilizando análisis de componentes independientes (ICA)

Lic. Juan Enrique García Bracamonte

Tesis sometida como requisito parcial
para obtener el grado de

Maestro en Ciencias con Especialidad
en Electrónica

Supervisada por

Dr. Juan Manuel Ramírez Cortés

Instituto Nacional de Astrofísica, Óptica y Electrónica

Santa María de Tonantzintla, Puebla, México
Agosto de 2018

Detección de fallas en motores trifásicos de inducción utilizando análisis de componentes independientes (ICA)

Tesis de Maestría

Por:

Juan Enrique García Bracamonte

Asesor:

Dr. Juan Manuel Ramírez Cortés

Instituto Nacional de Astrofísica, Óptica y Electrónica

Coordinación de Electrónica

Tonantzintla, Puebla. Agosto 2018

Agradecimientos

A mi padre y madre por el apoyo que me han brindado durante toda mi formación académica. Ellos son una razón importante de lo lejos que he llegado.

A mi futura esposa por siempre estar presente en mi vida a pesar de la distancia.

Al Dr. Juan Manuel Ramírez, mi asesor académico, por el tiempo dedicado a correcciones para que este proyecto fuese posible.

Al Dr. José Rangel, a la Dra. Pilar Gómez y al Dr. Jorge Carballido por ser mis sinodales y corregir mi trabajo.

A mis amigos Joan, Warnes, Mike, Sergio, Juan, Lalo entre muchos otros que hicieron amena mi estancia en Puebla.

Al Instituto Nacional de Astrofísica, Óptica y Electrónica INAOE por la oportunidad ofrecida y a CONACyT por la beca otorgada.

Resumen

Los motores trifásicos de inducción de tipo “Jaula de ardilla” son por mucho la maquinaria rotatoria más utilizada en la industria. La detección de fallas a tempranas etapas es de gran importancia debido a los problemas tanto técnicos como económicos que se pueden ocasionar, siendo el problema más común en motores de este tipo la ruptura de una o varias barras del rotor. Esta tesis presenta una técnica de detección de fallas utilizando la señal de corriente de una de las fases del motor. Se utiliza la técnica de separación de señales de “análisis de componentes independientes” (ICA) en el espectro de frecuencia de la señal de corriente y de su autocorrelación para extraer características de la falla. Una notable diferencia en la desviación estándar sobre una región de interés puede ser distinguida entre características extraídas de una señal de corriente de un motor sano y de uno con daño en una barra del rotor. Posteriormente, se hace uso de una red neuronal para la clasificación final de la condición del motor. Con las metodologías propuestas se obtienen porcentajes de clasificación distintos dependiendo de las condiciones a trabajar. Clasificando barra rota o sano con umbrales como clasificador y con señales de corriente con 50 % de carga del motor se obtienen exactitudes del 91 % y con datos provenientes de señales de 75 % de carga obtenemos hasta un 95 % de exactitud en promedio. Para estos dos casos anteriores utilizando redes neuronales se obtienen exactitudes del 97 % y 99 % respectivamente. Clasificando entre barra rota, media barra rota y sano, la exactitud alcanzada es de 85 % en el caso de señales de 75 % de carga y 80 % en señales de 50 % de carga en promedio.

Abstract

Squirrel-cage induction motors are among most used rotary machinery in many industrial fields. Fault detection in early stages is in high relevance due to technical and economic issues, and broken bars are among the most common faults in induction motors. This paper presents an approach to carry out detection of this failure using as input the current signal measured from one of the three motors phases. Independent Component Analysis (ICA) is used over the Fourier-domain spectral signals obtained from the input and its autocorrelation function. A notable difference on the standard deviation over a region of interest in one output can be distinguished in the current signals obtained from damaged and healthy motors. Further, a neural network is used to classify the motor's condition. The proposed methodologies obtained different accuracy depending on the considerations. Classifying between broken bar and healthy motor using threshold values on current signals from 50 % load motor the accuracy is around 91 % and working with 75 % load signals the accuracy is around 95 % on average. Classifying the same two motor conditions using neural network the performance of the classification is 97 % and 99 %, respectively. Classifying between broken bar, half broken bar and healthy motor, the accuracy obtained is 85 % working with 75 % load signals and 80 % working with 50 % signals.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Problemática y trabajos relacionados

Los motores de inducción trifásicos son un pilar importante en el mundo de la industria, pero como cualquier otro mecanismo rotatorio electromecánico son susceptibles a muchos tipos de fallas en distintas áreas de aplicación. En las últimas décadas, el diagnóstico y monitoreo de fallas en maquinarias ha sido un tópico de investigación desafiante en la comunidad científica. Específicamente, mucho esfuerzo ha sido dedicado al diagnóstico de fallas en motores de inducción debido a las posibles consecuencias tanto económicas como técnicas que pueden presentarse si los problemas no son atendidos a tiempo. Los motores de inducción del tipo "Jaula de ardilla" son por mucho la maquinaria rotaria más utilizada en la industria, representando aproximadamente el 85 % del consumo de energía en plantas industriales. Un motor de inducción convierte eficientemente energía eléctrica en mecánica haciendo girar un rotor por medio de inducción de un campo magnético rotatorio. A pesar de que este tipo de motores no tiene componentes que hagan fricción ocasionando desgaste en sus elementos, existe una gran variedad de tipos de fallas que se pueden presentar en estos motores, tales como desbalances mecánicos, fallas en rodamiento, asimetría de los bobinados del estator o rotor, falla de sobrecarga, etc., pero a pesar de esto el problema más común es detectar una o varias barras rotas en la jaula del rotor, particularmente en sistemas de trabajo pesado. Un motor con una barra de rotor rota puede aparentar trabajar correctamente, sin embargo, esta falla puede ocasionar mayor consumo de potencia, calentamiento excesivo, vibraciones indeseadas, incluso causar rupturas en las barras sanas del rotor, entre otros problemas. Distintas metodologías para detección de fallas han sido publicadas en las décadas pasadas. La

mayoría de las técnicas de detección utilizan para el análisis señales de corriente [1] [2], señales de vibración [3] [4] [5], o una combinación de ambas [6]. Muchas de estas técnicas propuestas utilizan la transformada rápida de Fourier para analizar el espectro de frecuencia en búsqueda de características espectrales de las fallas. Es común ver el uso de redes neuronales para la clasificación del estado de un motor con distintos enfoques, [7] propone el uso de redes neuronales para detección de fallas utilizando señales de vibraciones en el dominio del tiempo. En [8], redes neuronales son empleadas para detectar fallas de excentricidad en motores de inducción. En trabajos previos está demostrado que el espectro en frecuencia de un motor dañado exhibe componentes indeseadas cercanas a la frecuencia fundamental provenientes de la falla [9] [10] [11]. Sin embargo, las técnicas de separación para la extracción de estas componentes indeseadas son cruciales como primer paso para sistemas de análisis de detección de fallas. Esta tesis presenta un enfoque de análisis de firma de corriente del motor (MCSA por sus siglas en inglés *Motor Current Signature Analysis*) para detección de fallas en motores de inducción de tipo "jaula de ardilla", utilizando el algoritmo ICA en el dominio de la frecuencia de la señal de corriente del estator y de su autocorrelación para separar las características espectrales de de la falla. ICA tiene un sin fin de posibles aplicaciones como por ejemplo procesado de imágenes, sistemas de reconocimiento de voz, procesamiento de señales médicas, de telecomunicaciones, etc. En el diagnóstico de fallas ICA ha sido una técnica relevante con distintos enfoques. En [12], se presenta un análisis de defectos de rodamientos basado en vibraciones. En su enfoque, ICA es usado para la separación de señales multicanal que contienen información de la falla de rodamiento, generando una firma de la falla que se observa en una de las componentes independientes. [13] [14] presentan una interesante metodología con MCSA para la detección en línea de fallas de motores de inducción. En esta metodología ICA es utilizado en el dominio de la frecuencia a las señales espectrales obtenidas de la corriente del estator. Estas señales llamadas FFT-ICA contienen importante información para realizar una clasificación satisfactoria. [15] propone un enfoque de diagnóstico de fallas de motores de inducción combinando el uso de máquinas de soporte vectorial (SVM) e ICA, basándose en señales de corriente de estator y vibraciones del motor. En su enfoque, ICA es usado para extracción de características y reducción de información de las características originales. La aportación principal de este trabajo de tesis es detectar barra rota en un motor de inducción utilizando solamente la corriente de una de las fases del motor. Para lograr esto se utiliza ICA

como técnica de separación de las características de la falla, utilizando como entradas al ICA el espectro de frecuencia de la señal de corriente y el espectro de su autocorrelación. Una notable diferencia de la desviación estándar en una región de interés se presenta en las características extraídas de una señal de corriente de un motor sano y un motor con daño. Esto nos permite utilizar dicho parámetro para la clasificación del estado del motor. Adicionalmente para una mejor clasificación la entropía y la curtosis de la misma región son calculadas para alimentar una red neuronal. Utilizando el método propuesto y experimentando con variantes se logra una exactitud de entre el 95 % y 99 %, siendo la variante más efectiva, la que utiliza redes neuronales como clasificador.

1.2. Organización de la tesis

Esta tesis está organizada en 5 capítulos. El primer capítulo es una introducción a la problemática de la detección de fallas en el sector industrial debido a la alta demanda en el uso de motores de inducción trifásicos en este sector. También se hace mención a trabajos relacionados donde se desarrollan diferentes métodos de detección de fallas monitoreando señales de corriente, vibraciones o ambas. El capítulo 2 contiene las bases teóricas necesarias para entender el problema, así como las técnicas de procesamiento de señales y herramientas matemáticas requeridas para su solución. El tercer capítulo aborda la metodología aplicada para la clasificación del estado de un motor utilizando distintos métodos para la detección de fallas basados en características contenidas en la señal de corriente. El capítulo 4 expone resultados y tablas obtenidas con las metodologías propuestas, también se toca la explicación del conjunto de datos utilizado para reportar un promedio en los resultados, así como la simulación del algoritmo en el software Vivado HLS. Por último, el capítulo 5 presenta algunas conclusiones a las que se llegó, trabajos a futuro para seguir desarrollando este proyecto y artículos publicados derivados del mismo.

Capítulo 2
Marco Teórico

2.1. Motor de inducción de tipo "Jaula de ardilla"

Los motores de inducción son una alternativa rentable debido a que son los más baratos de construir, los de menor mantenimiento y los más confiables para aplicaciones industriales comparados con otras alternativas de motores. Existen dos tipos de motores de inducción, también llamados "asíncronos trifásicos", dependiendo del tipo de rotor que utilizan. Un tipo es el de rotor bobinado (motor de anillos rozantes) y el otro tipo es de rotor en corto circuito (motor de jaula de ardilla), siendo este último el más comúnmente utilizado en la industria.

2.1.1. Partes y funcionamiento interno

Constan de dos partes esenciales, el estator y el rotor (Fig.2.1). El estator es la parte inmóvil y el rotor es la parte rotatoria del motor. El nombre de este tipo de motores viene de la forma tan semejante del rotor a una jaula de ardilla. El estator contiene una serie de devanados de tal manera que al alimentar los devanados con corriente alterna trifásica genera un campo magnético interno rotatorio. Esta variación de campo magnético induce una corriente en las barras del rotor la cual a su vez genera un campo magnético que al interactuar con el campo giratorio del estator genera una fuerza electromotriz que hace girar al rotor. La velocidad de giro del rotor es ligeramente menor a la velocidad de rotación del campo magnético del estator. Esto tiene una explicación muy sencilla, si se iguala la velocidad el rotor a la del campo giratorio , el rotor experimenta un campo magnético constante, lo cual significa que no hay corriente inducida en las barras del rotor por lo que no hay fuerza electromotriz que lo haga girar, esto ocasiona pérdida de velocidad del rotor,

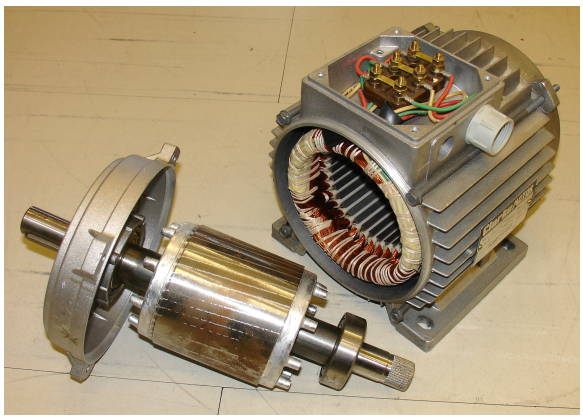


Figura 2.1: Partes esenciales de un motor de jaula de ardilla. Rotor (abajo a la izquierda) y estator (arriba a la derecha.)

el cual vuelve a experimentar campo magnético variable, se induce corriente, y vuelve a existir una fuerza electromotriz que hace girar más rápido al rotor, haciendo esto un ciclo interminable mientras los devanados del estator estén alimentados.

2.1.2. Algunas fallas posibles

A pesar de que la falla que más comúnmente se presenta es tener una o varias barras rotas hay una variedad de tipos de fallas posibles. Por mencionar algunas, las fallas se pueden clasificar en dos grupos importantes, fallas mecánicas y fallas eléctricas. Dentro de las fallas mecánicas se puede presentar desbalance mecánico que produce vibraciones indeseadas ocasionando problemas en elementos rotatorios, en engranes dentados acoplados al motor, en el soporte, entre otros daños. Otra falla mecánica se puede presentar en problemas en el rodamiento, las causas de este problema son variadas, dígase mala calibración, falta de la adecuada lubricación o sellado deficiente. La excentricidad también clasifica en las fallas de tipo mecánicas, ésta se da cuando el eje de rotación no está alineado con el eje del rotor. Entre las fallas eléctricas posibles se encuentran la asimetría tanto en bobinados del estator como en el rotor, esta asimetría se debe principalmente a defectos de fabricación y produce pequeños campos magnéticos que se oponen al campo del estator. Otra falla de tipo eléctrica es tener bobinados sueltos en el estator, esta falla es muy destructiva ya que afecta el aislamiento de los conductores ocasionando cortos circuito en los bobinados, o fallas en el estator.

2.2. Análisis de firma de corriente de motor (MCSA)

El análisis de un motor puede realizarse utilizando diversos tipos de señales, una de las más comúnmente utilizadas son señales de corriente. La confiabilidad de MCSA (por sus siglas en inglés *Motor Current signature analysis*) ha sido un tópico muy importante de investigación en los últimos años. Una barra rota genera componentes indeseadas en el espectro de frecuencia dados por:

$$f_b = (1 \pm 2ks) f_s \qquad k = 1, 2, 3... \tag{2.2.1}$$

Donde f_b corresponde a las componentes indeseadas, s es el deslizamiento por unidad y f_s la frecuencia fundamental de operación. La detección de fallas mediante MCSA y análisis de Fourier se basa en identificar estas componentes parásitas, las cuales en ocasiones se encuentran muy cerca de la frecuencia fundamental de operación. Además de esto, su amplitud es muy pequeña en comparación con la componente de alimentación del motor. Como ya se ha demostrado en otros trabajos, la magnitud y posición de las componentes parásitas depende de la gravedad de la falla [11].

2.3. Análisis de componentes independientes

ICA (por sus siglas en inglés *Independent Component Analysis*) [16] [17] es el método más comúnmente utilizado en separación ciega de señales, cuyo objetivo es básicamente separar señales fuente (componentes independientes) a partir de señales observadas sin tener previo conocimiento de los pesos utilizados en la mezcla. Un enfoque clásico para entender el problema que enfrenta ICA es el esquema denominado "Fiesta de cóctel" (*cocktail party*) [18].

2.3.1. Esquema "Fiesta de cóctel"

Imagine que hay dos personas hablando simultáneamente en una misma habitación en la cual hay dos micrófonos en distintas locaciones. Cada micrófono provee una grabación, las cuales son señales en el tiempo. Cada una de esas grabaciones, que denotaremos como $x_1(t)$ y $x_2(t)$, representa la suma ponderada de los discursos de las

personas. Definiendo $s_1(t)$ y $s_2(t)$ como la voz de cada persona, es decir las fuentes de las señales, podemos expresar lo anterior como una ecuación lineal:

$$x_1(t) = a_{11}s_1 + a_{12}s_2 \tag{2.3.1}$$

$$x_2(t) = a_{21}s_1 + a_{22}s_2 \tag{2.3.2}$$

Si conociéramos los valores de los pesos a_{ij}, podemos estimar las señales fuente con métodos clásicos de resolución de ecuaciones lineales. Sin embargo, al no conocerlos el problema se complica. El problema real es separar las señales s_1 y s_2 sin previo conocimiento de las constantes de peso a_{ij}.

2.3.2. Modelo Matemático de ICA

ICA es un método computacional que tiene como objetivo separar señales fuentes a partir de señales observadas que son consideradas como la combinación lineal de las señales a separar, esta separación se lleva a cabo sin tener previo conocimiento de la mezcla. Utilizando como analogía el esquema "Fiesta de cóctel" podemos encontrar un modelo matemático de ICA. Primeramente definimos al vector $x(t) = (x_1(t), x_2(t), ..., x_n(t))$ como n combinaciones lineales observadas y al vector $s(t) = (s_1(t), s_2(t), ..., s_n(t))$ como las señales fuente. Extendemos las ecuaciones (2.3.1) y (2.3.2) para n señales:

$$x_k = a_{k1}s_1 + a_{k2}s_2 + ... + a_{kn}s_n \tag{2.3.3}$$

para $k = 1, 2, ..., n$.

Obviando la dependencia del tiempo en el modelo de ICA, asumimos que tanto las señales mezcladas x_k como los componentes independientes s_j son variables aleatorias. Sin pérdida de generalidad asumimos media cero en todas las señales. Si esto no fuera cierto, las señales observadas siempre pueden ser centradas extrayendo la media. Utilizando la notación vectorial-matricial y denotando A a la matriz con los elementos a_{ij}, reescribimos (2.3.3) como:

$$x = As \tag{2.3.4}$$

(2.3.4) es conocido como el modelo ICA. La matriz A es considerada desconocida y los componentes independientes s_i no pueden ser medidos directamente. Todo lo

que observamos es el vector x, y partiendo de eso ICA debe estimar A y s. Esto debe realizarse como sea posible bajo ciertas consideraciones generales. El punto de partida de ICA es asumir independencia en los componentes y una distribución no Gaussiana. En este modelo básico son consideradas desconocidas las distribuciones de las señales fuente, debido a que si se conocieran el problema se simplifica. Asumimos también por simplicidad que la matriz A es una matriz cuadrada, aunque esta consideración puede ser flexible en ocasiones. Después de estimar la matriz A, podemos calcular su inversa W y obtener así las componentes independientes s_i simplemente como:

$$s = Wx \tag{2.3.5}$$

En algunas aplicaciones sería más realista aplicar ruido a las señales medidas, pero por simplicidad en el modelo se omite cualquier ruido en la señal, debido a que el modelo sin ruido es suficientemente difícil por sí mismo y para muchas aplicaciones parece ser suficiente.

2.3.3. Consideraciones y fundamentos para la estimación de ICA

El punto de partida de ICA es asumir independencia en las componentes a estimar, así como asumir que la distribución de las señales no es Gaussiana. Para realizar la estimación de los componentes independientes ICA debe tomar un parámetro de identificación que le ayude a determinar si la señal estimada corresponde a una señal fuente o no. Respecto a este parámetro de identificación, ICA tiene varios enfoques que utilizan distintos parámetros, pero todos con un mismo propósito, encontrar la señal que menos se asemeje a una distribución Gaussiana. Esto es debido a que el teorema de límite central nos dice que la suma de dos o más señales siempre tiene una distribución más Gaussiana que las señales por sí solas [19]. Si ambas señales fuente tienen distribución Gaussiana no es posible la estimación de la matriz de mezcla debido a falta de información. Sin embargo, si solo una de las señales es Gaussiana la estimación de ICA es posible [20]. En los distintos enfoques de ICA se utilizan distintos parámetros para medir la Gaussianidad de una señal, el más común es la curtosis pero también se utiliza la negentropía, entre otros. En la sección 2.3.5 se habla un poco más a fondo del enfoque utilizado en esta tesis.

2.3.4. Blanqueo de señales

El blanqueo de los datos ("*Whitening*") tiene como fin tener datos lo menos correlacionados posibles para evitar redundancia. La matriz de covarianza de los datos blanqueados corresponde a la matriz identidad **I**, esto quiere decir que las señales son no correlacionadas y cada una tiene varianza de 1 [21]. El blanqueo es un preprocesamiento necesario en muchos algoritmos, los pasos para el blanqueo de datos son los siguientes:

- Centrar los datos, es decir, extraer la media.
  $X_c = X - E[X]$
- Calcular la matriz de covarianza de los datos centrados $Cov(X_c)$, así como sus valores propios D y vectores propios V.
- Blanquear los datos
  $Z = D^{-1/2} V X_c$

El blanqueo de las señales de entrada a ICA es muy importante ya que algunas de las consideraciones hechas toma en cuenta que las entradas sean datos blanqueados.

2.3.5. ICA por minimización y maximización de la curtosis

Una de las soluciones más utilizadas para la evaluación de Gaussianidad en el problema de ICA, es usar el cuarto momento centrado o curtosis de la señal [16], definida para una variable aleatoria v de media cero como:

$$kurt(v) = E\left\{v^4\right\} - 3(E\left\{v^2\right\})^2 \tag{2.3.6}$$

La curtosis de una variable aleatoria con distribución Gaussiana es cero; para densidades inclinadas a valores cercanas a cero, positiva, y para densidades más planas, negativa. Para minimizar o maximizar $kurt(w^T x)$, un algoritmo neuronal basado en gradiente descendente y/o ascendente puede ser usado. Entonces w es interpretado como el vector de peso de una neurona con entrada el vector x. La función objetivo

puede ser simplificada debido a que las entradas fueron blanqueadas, se mantiene que:

$$kurt(w^T x) = E\left\{(w^T x)^4\right\} - 3[E\left\{(w^T x)^2\right\}]^2 = E\left\{(w^T x)^4\right\} - 3||w||^4 \qquad (2.3.7)$$

La consideración de que $|w| = 1$ debe tomarse en cuenta. Tomando como base (2.3.7) se puede llegar a un algoritmo de rápida convergencia, el cual es mostrado a continuación.

2.3.6. Algoritmo de punto fijo FastICA

Estimar una componente. Para esto se asume que tenemos una señal **x** blanqueada, el cual es el caso de separación ciega de señales que obtiene combinaciones lineales de señales fuente. Utilizando la derivación de la subsección anterior, se obtiene un algoritmo iterativo de punto fijo de ICA, el cual contiene los siguientes pasos:

1. Tomar un vector inicial aleatorio unitario w_0. Inicializar $k = 1$.

2. Estimar un nuevo vector $w_k = E\left\{x(w_{k-1}^T x)^3\right\} - 3w_{k-1}$

3. Normalizar w_k.

4. Checar condición de convergencia. Si $|w_k^T w_{k-1}|$ no está lo suficientemente cerca de 1, hacemos $k = k + 1$ y volvemos a paso 2. De lo contrario, mandar a la salida el vector w_k.

El vector final w_k obtenido por el algoritmo equivale a una columna de la matriz de mezcla B. Para el caso de separación ciega de señales w_k significa un vector para separar una de las señales fuentes no Gaussianas. Es decir, $w_k^T x(t), t = 1, 2, ...$ equivale a una señal fuente.

Estimar múltiples componentes. Para estimar n componentes, solamente efectuamos el algoritmo n veces. Para asegurar la convergencia en un vector w_k distinto en cada ocasión, solo es necesario agregar una simple proyección de ortogonalización dentro del ciclo del algoritmo. Recordando que las columnas de la matriz de mezcla B son ortonormales debido al blanqueado de las señales, podemos estimar las componentes una por una proyectando la solución actual w_k a las columnas de la matriz de mezcla B previamente encontrada. La operación de proyección se agrega al comienzo del paso 3:

3. $w_k = w_k - BB^T w_k$. Normalizar w_k.

2.4. Transformada discreta de Fourier

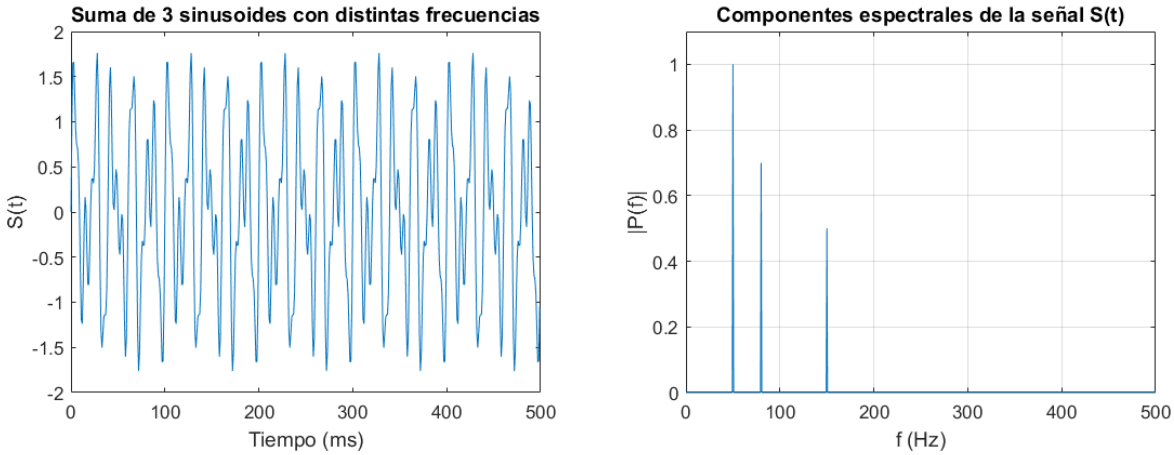
La DFT (por sus siglas en inglés *Discrete Fourier Transform*) es un tipo de transformada discreta utilizada en el análisis de Fourier que transforma una señal en otra, obteniendo una representación en el dominio de la frecuencia, siendo la señal original una señal en el dominio del tiempo (Fig. 2.2). La DFT requiere que la señal de entrada sea una secuencia discreta y de duración finita, por esta última razón la DFT es ampliamente utilizada en diversas áreas de la electrónica debido a que la electrónica nos obliga a trabajar con un número finito de datos discretos que además tienen una precisión finita. La entrada de la DFT es una secuencia finita de números reales o complejos, usualmente muestreadas a partir de una función continua, de modo que es ideal para procesar información almacenada en soportes digitales. En particular, la DFT se utiliza comúnmente en procesado digital de señales y otros campos relacionados dedicados a analizar las frecuencias que contiene una señal muestreada. Un factor muy importante para este tipo de aplicaciones es que la DFT puede ser calculada de forma eficiente en la práctica utilizando el algoritmo de la transformada rápida de Fourier o FFT (por sus siglas en inglés *Fast Fourier Transform*). El cálculo de FFT toma aproximadamente $N * log_2(N)$ operaciones, mientras que DFT toma aproximadamente N^2 operaciones.

La secuencia de N números complejos $x_0, ..., x_{(N-1)}$ se transforma en la secuencia $X_0, ..., X_{(N-1)}$ mediante la DFT utilizando la fórmula:

$$X_k = \sum_{n=0}^{N-1} x_n \mathbf{e}^{\frac{-2\pi}{N} kn} \qquad k = 0, ..., N-1 \tag{2.4.1}$$

Donde **i** es la unidad imaginaria. La DFT es reversible, siendo capaz de transformarse en cualquiera de los dominios al otro. La transformada inversa discreta de Fourier (IDFT) viene dada por:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \mathbf{e}^{\frac{2\pi}{N} kn} \qquad n = 0, ..., N-1 \tag{2.4.2}$$



(a) (b)

Figura 2.2: a) Señal en el dominio del tiempo $S(t)$ correspondiente a la suma de tres sinusoides de distintas frecuencias, b) Componentes espectrales de $S(t)$, correspondientes a las frecuencias presentes en la señal $S(t)$.

2.5. Red Neuronal Artificial (RNA)

El cerebro humano tiene la habilidad de pensar, recordar, aprender y resolver problemas, esta habilidad ha sido una inspiración para muchas investigaciones para desarrollar modelos artificiales cuyas bases de aprendizaje sean muy similares a las de una neurona biológica. Dichas investigaciones dieron como resultado RNAs, que son simplemente modelos basados en el proceso biológico de aprendizaje del cerebro humano [22]. Una RNA consiste en un cierto número de neuronas artificiales conectadas entre sí llamadas nodos, este conjunto de conexiones agrupadas por capas forman una red. Una RNA típica tiene una capa de entrada, una capa oculta y una capa de salida (Fig. 2.3), este esquema típico es llamado RNA de dos capas, la capa de entrada no es contada como capa debido a que no se efectúan cálculos en ella. El número de nodos de la capa de entrada y de la capa de salida son determinados por la naturaleza del problema a resolver. En todas las capas de la red, exceptuando la capa de entrada, cada neurona efectúa la siguiente operación:

$$sum = \sum X_i W_i - \theta_k \qquad i = 1, 2, ..., N; k = 1, 2, ..., L \tag{2.5.1}$$

Donde N es el número de entradas, L el número de neuronas en la capa oculta, X_i representa cada entrada a la neurona, W_i su correspondiente peso y θ_k es valor

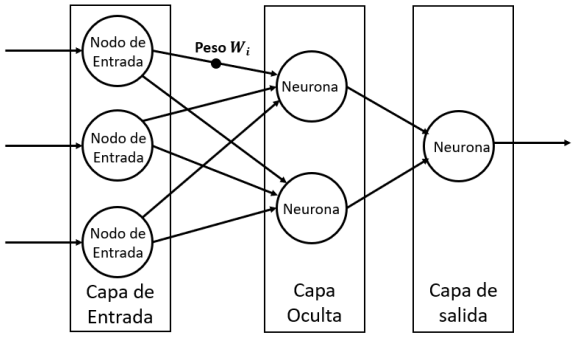


Figura 2.3: Un esquema típico de una red neuronal artificial de dos capas.

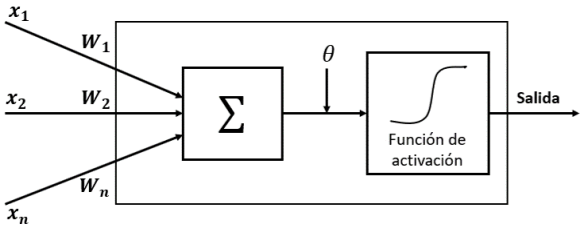


Figura 2.4: Funcionamiento de una neurona artificial en una capa de la red.

adicional presente en cada neurona conocido como sesgo. Una función de activación no lineal transforma el resultado de la ecuación (2.5.1) en el valor de salida de la neurona. La sigmoide es la función de activación más comúnmente utilizada en RNAs, la cual restringe la salida de las neuronas entre 1 y 0. También es común utilizar como función de activación la función tangente hiperbólica o la sigmoide simétrica. Una vista esquemática del funcionamiento típico de una neurona explicado anteriormente es mostrado en la Fig. 2.4.

2.6. Correlación cruzada y autocorrelación

La correlación en procesado de señales nos permite determinar como cambia una señal en el tiempo, si existe cierto parecido en la forma de onda $x(n)$ si consideramos diferentes intervalos temporales. La correlación cruzada es utilizada para obtener información del parecido entre formas de onda diferentes y la autocorrelación, cuando se quiere información del parecido dentro de una misma señal $x(n)$ [23]. La correlación es un proceso parecido a la convolución con la diferencia de que ninguna señal es

invertida. Se define correlación cruzada entre dos señales $x(n)$ e $y(n)$ como:

$$r_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)y(n-k) \qquad k = 0, \pm 1, \pm 2, ... \tag{2.6.1}$$

Se define autocorrelación de una señal $x(n)$ como:

$$r_{xx}(k) = \sum_{n=-\infty}^{\infty} x(n)x(n-k) = r_{xx}(-k) \qquad k = 0, \pm 1, \pm 2, ... \tag{2.6.2}$$

La energía de la señal corresponde a $r_{xx}(0)$.

Capítulo 3
Metodología

3.1. Arreglo experimental del banco de señales

Las señales de corriente con las que se prueba la metodología propuesta en este trabajo de tesis fueron medidas con un motor de inducción de 1-hp, realizando mediciones con dos condiciones de carga del motor distintas, con el 50 % y con el 75 % de su capacidad máxima. La gravedad de la falla del motor se clasifica con los milímetros de ruptura que tiene la barra, siendo 10mm el máximo o lo que es lo mismo barra rota. El banco de datos consiste en un conjunto de 150 señales de corriente de una fase del motor, correspondientes a tres condiciones de daño del motor, es decir, 50 señales de cada condición de daño. Las tres condiciones son: motor sano (0mm de daño); media barra rota (5mm de daño); barra rota (10mm de daño). La base de datos fue generada a una frecuencia de muestreo de 3.2kHz correspondiente a las características del convertidor AD utilizado. Las señales de corrientes son medidas con un sensor lineal de corriente basado en el efecto Hall (ACS758LCB-050B) y un amplificador instrumental (INA126) para ajustar las mediciones al rango requerido. Después las señales son digitalizadas con un convertidor AD de 12-bits (ADS7841). El arreglo experimental es mostrado en la Fig.3.1.

3.2. Metodología general

El diagrama de bloques de la metodología general propuesta es mostrado en Fig. 3.2. Las señales procesadas a través del algoritmo ICA corresponden a la transformada de Fourier de la señal de corriente de una de las fases del motor y de su autocorrelación. Una ventana normalizada de 1024 puntos es toma-

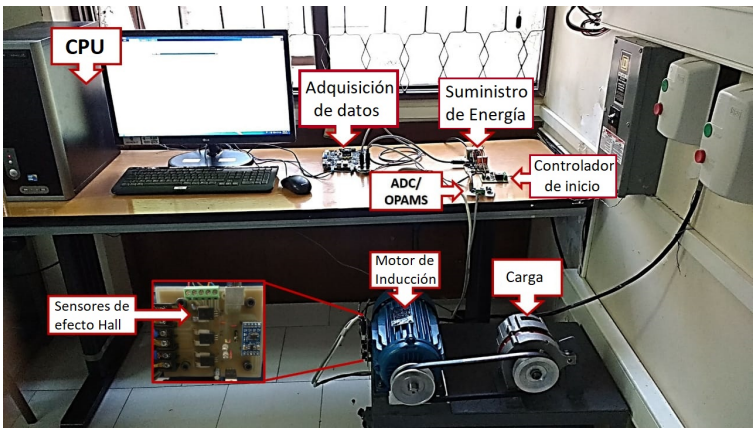


Figura 3.1: Arreglo experimental montado con el fin de obtener mediciones de la señal de corriente de una fase de un motor de inducción en distintas condiciones de carga y daño.

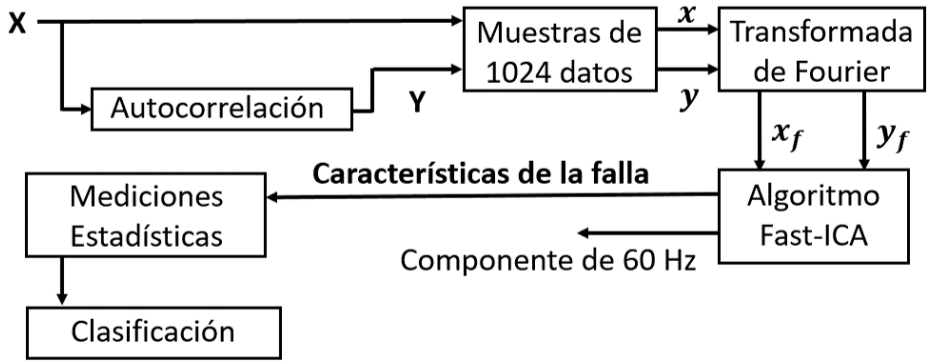


Figura 3.2: Diagrama de bloques de la metodología propuesta, donde X representa la señal de corriente de una fase de un motor de inducción.

da tanto de la señal de corriente como de su autocorrelación para ser procesadas con la FFT obteniendo dos señales en el dominio de la frecuencia que denotaremos como X_f y Y_f, estas señales son entonces utilizadas para la extracción de características espectrales de la falla. El algoritmo ICA provee dos salidas (Fig.3.3), una de las cuales corresponde a la componente de 60 Hz presente en la línea eléctrica y la otra corresponde a las características de la falla extraídas de la señal de corriente, la cual es utilizada para discriminar el estado del motor. La Fig. 3.4 muestra las características extraídas en dos estados del motor distintos: sano y con 10mm de dañado. De esta imagen de ejemplo se puede notar que hay una región de interés donde los componentes espectrales muestran diferencias acordes al nivel de daño del motor. La metodología aprovecha estas diferencias realizando un análisis estadístico en la región de interés con motivo de realizar la clasificación requerida. La desviación estándar de los componentes espectrales situados a la izquierda

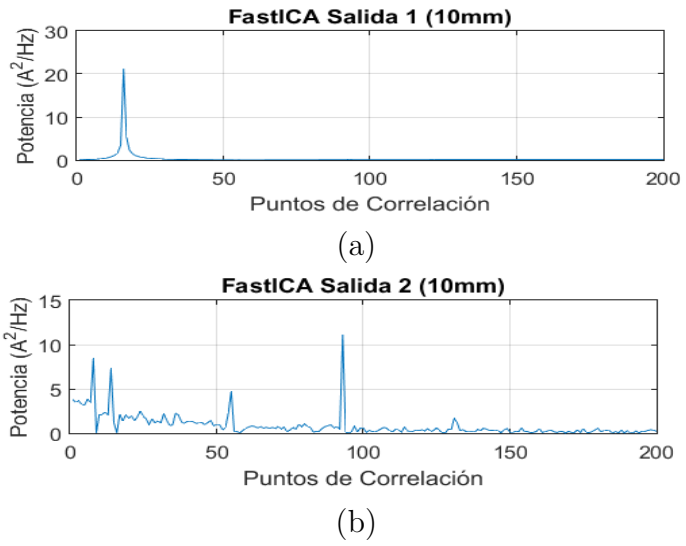


Figura 3.3: Señales de salida del algoritmo Fast-ICA de un motor con 10mm de daño, a) Señal correspondiente a la componente de 60 Hz de la línea eléctrica y b)Características de la falla extraídas.

del pico más grande, el cual no es incluido en el análisis, es usada para discriminar el estado del motor. La región de interés entonces son los primeros 90 puntos de los componentes espectrales obtenidos de la salida de ICA. Ahora bien, diferentes métodos son usados a manera de obtener la mejor clasificación de la manera más sencilla. Llamaremos "experimento" a una prueba con una señal de 1024 puntos, al tomar otro kernel de la señal de corriente tendremos un experimento distinto.

3.3. Métodos propuestos para clasificación

3.3.1. Desviación estándar de toda la región de interés

La región analizada fueron los primeros 90 puntos de las características extraídas (Fig.3.6a). El análisis de la desviación estándar de la región de interés muestra que este parámetro estadístico puede ser utilizado para discriminar la condición de un motor. Un valor límite que llamaremos umbral es tomado para la clasificación, el cual es obtenido con un promedio de 500 experimentos sanos y 500 dañados de la desviación estándar de la región de interés. Un ejemplo de los valores de desviación de la región de interés puede verse en la Fig.3.5.

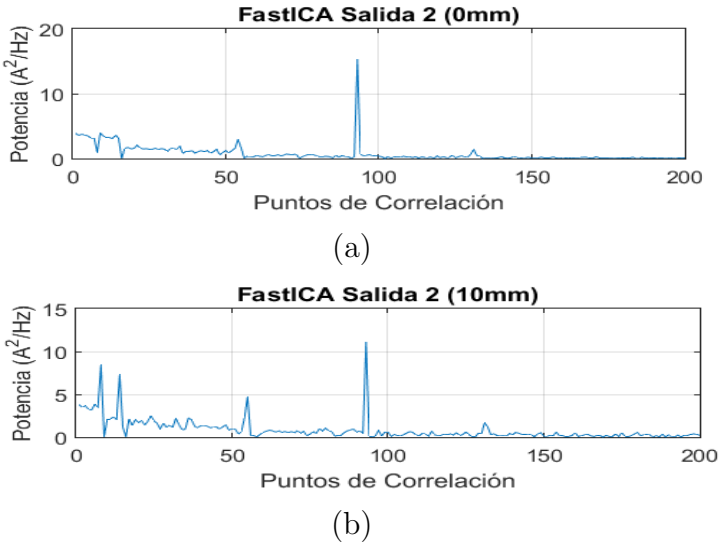


Figura 3.4: a) Características extraídas de una señal de corriente de un motor con 0mm de daño, es decir, motor sano, b) Características extraídas de una señal de corriente de un motor con 10mm de daño o bien, barra rota.

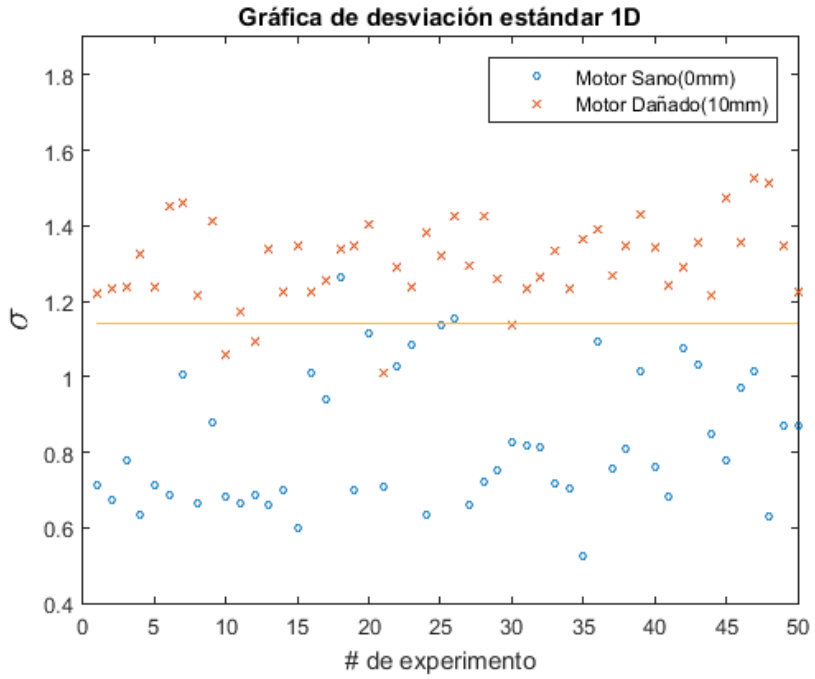
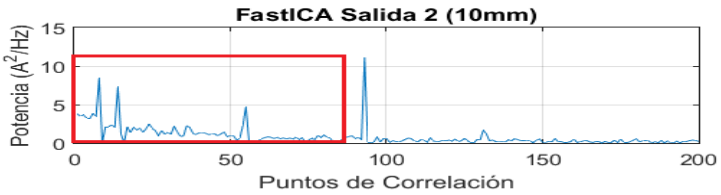
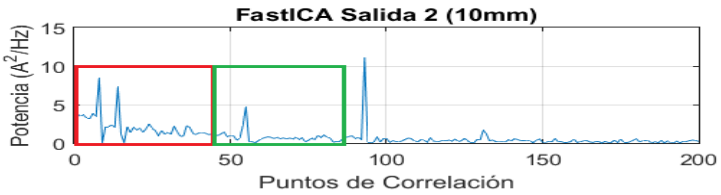


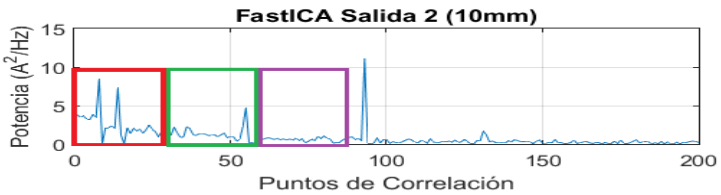
Figura 3.5: Desviación de la región de interés completa de 100 experimentos (50 sanos y 50 dañados). La línea continua corresponde al umbral de clasificación.



(a)



(b)



(c)

Figura 3.6: Región de interés dividida en partes de la misma longitud. a) Una sola partición de 90 puntos, b) dos particiones de 45 puntos y c) tres particiones de 30 puntos.

3.3.2. Vector de desviaciones estándar: dos dimensiones

Si la región de interés es dividida en partes iguales y se calcula la desviación de cada partición como resultado se tiene un vector de desviaciones que en teoría ayuda a tener una mejor clasificación. El proceso de dividir la región de interés viene después de aplicar la extracción de características con ICA, entonces los pasos anteriores a este proceso permanecen exactamente igual en cada metodología explicada. La región de interés es cortada en 2 ó 3 partes dependiendo del caso (Fig. 3.6), en este caso es dividida en dos partes iguales (Fig. 3.6b). La desviación de cada partición es calculada y guardada en un vector que denotaremos como $\overrightarrow{V_{2std}}$, esto resulta que en lugar de tener un valor tengamos dos valores de desviación correspondientes a un punto en un plano. De manera similar al umbral, para la clasificación se utiliza una línea trazada a partir de una ecuación obtenida con el criterio de distancia Euclidiana de la muestra con respecto a la media (ecuación (3.3.1)).

$$|\overrightarrow{V_{2std}} - \overrightarrow{\mu_{sano}}| < |\overrightarrow{V_{2std}} - \overrightarrow{\mu_{10mm}}| \tag{3.3.1}$$

Donde $\overrightarrow{V_{2std}}$ representa el vector de desviaciones de la muestra actual, $\overrightarrow{\mu_{sano}}$ el vector promedio de 500 vectores de desviación sanos y $\overrightarrow{\mu_{10mm}}$ el vector promedio de 500 vectores con 10mm de daño en la barra. Si se satisface la desigualdad de la ecuación (3.3.1), la muestra actual es clasificada como sana debido a que la distancia euclidiana es menor hacia la media de las muestras sanas, de lo contrario la muestra es clasificada como dañada. Una gráfica de dispersión de los valores de desviación estándar de las particiones es mostrado en la Fig.3.7.

3.3.3. Vector de desviaciones estándar: tres dimensiones

Para este caso la región de interés es dividida en 3 partes de la misma longitud (Fig. 3.6c) y la desviación estándar de cada parte es calculada dando como resultado un vector de tres dimensiones, denotemos este vector como $\overrightarrow{V_{3std}}$. Ahora bien, cada vector $\overrightarrow{V_{3std}}$ representa un punto en el espacio y para realizar la clasificación del estado del motor se utiliza un plano obtenido con el criterio de distancia euclidiana (ecuación (3.3.1)) de la misma manera que en dos dimensiones sólo que esta ocasión existe una componente extra. La gráfica de dispersión correspondiente a la región de interés dividida en tres particiones se muestra en la Fig.3.8.

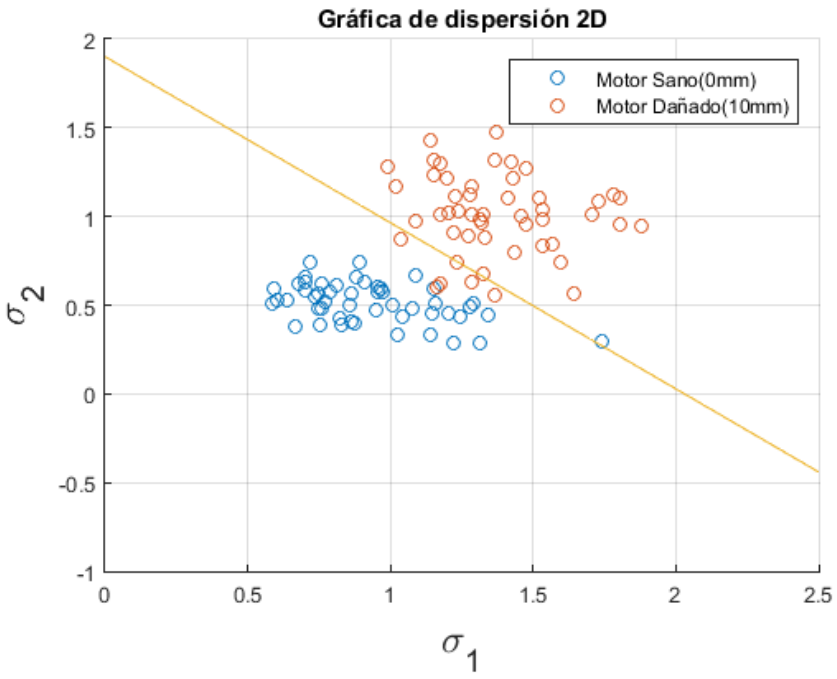


Figura 3.7: Gráfica de dispersión de 100 vectores V_{2std} (50 sanos y 50 dañados). La línea continua corresponde al umbral de clasificación.

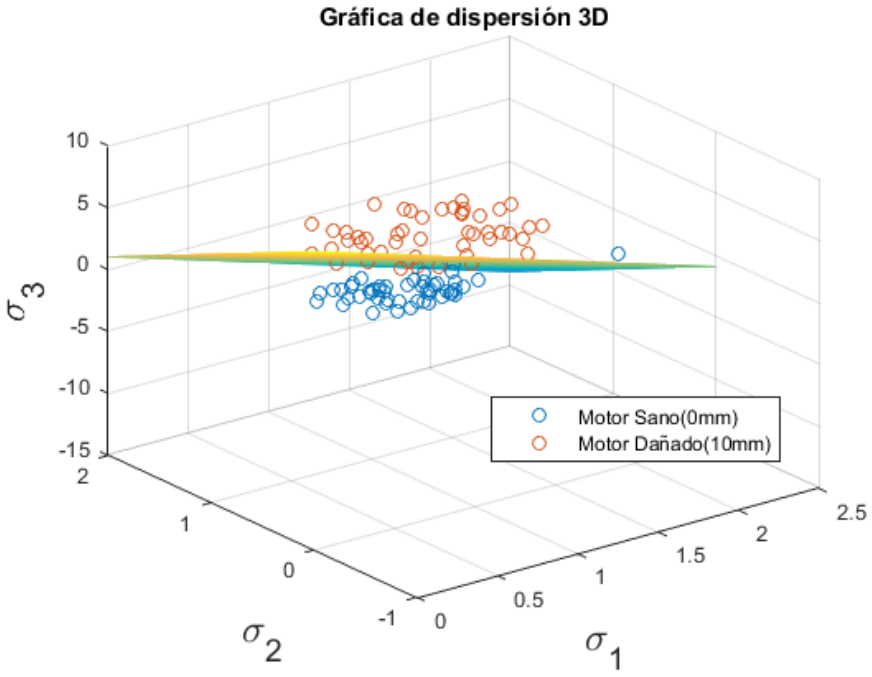


Figura 3.8: Gráfica de dispersión de 100 vectores V_{3std} (50 sanos y 50 dañados). El plano corresponde al umbral de clasificación.

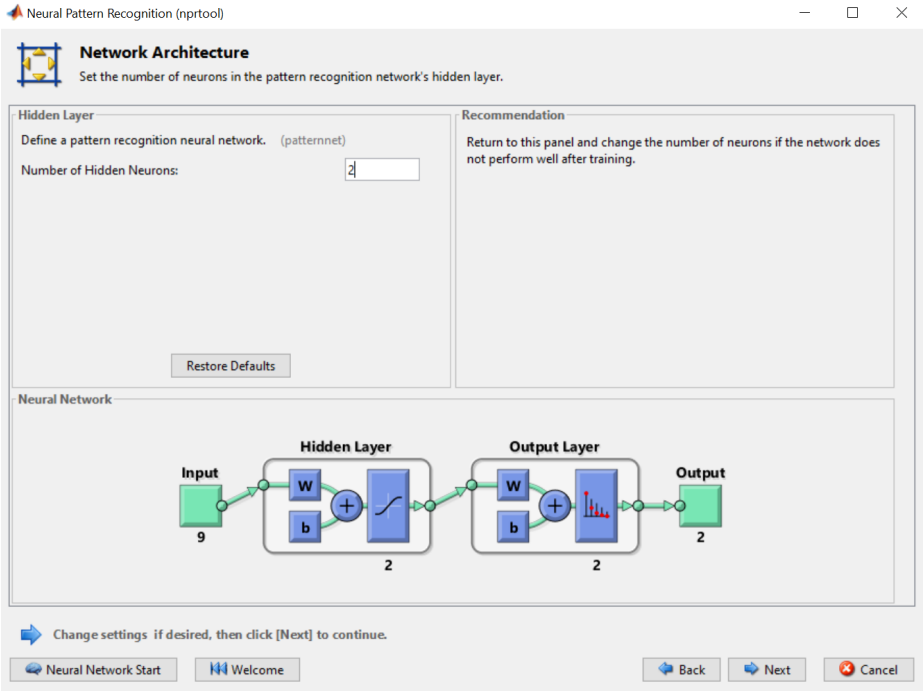


Figura 3.9: Captura de pantalla de la interfaz de la herramienta de redes neuronales de MATLAB

3.3.4. Red Neuronal en vectores de tres dimensiones

Una red neuronal artificial es usada en esta ocasión como clasificador. MATLAB provee una herramienta de reconocimiento de patrones neuronales, a la cual se accede con el comando "nprtool" por sus siglas en inglés *Neural Pattern Recognition Tool*, esta herramienta permite al usuario utilizar una red neuronal con una capa oculta, permitiéndole así mismo modificar el número de neuronas de dicha capa (Fig.3.9). MATLAB utiliza cierto porcentaje de los vectores de entrada para entrenar, validar y probar la red. Los porcentajes asignados para cada tarea pueden ser modificados, pero en este caso se dejan los valores predeterminados por la herramienta. El 70 % de los vectores de entrada son para entrenar, 15 % para validar y el 15 % restante es para probar la red. El número seleccionado de neuronas en la capa oculta es dos, no es necesario utilizar un número mayor de neuronas debido a que los resultados son prácticamente idénticos y al realizar alguna implementación posterior representa menor consumo de recursos. Las funciones de activación que utiliza la herramienta son sigmoide simétrica. La interfaz entrega como resultados una matriz de confusión (Fig.3.10) exponiendo los casos clasificados correctamente y en los que no.

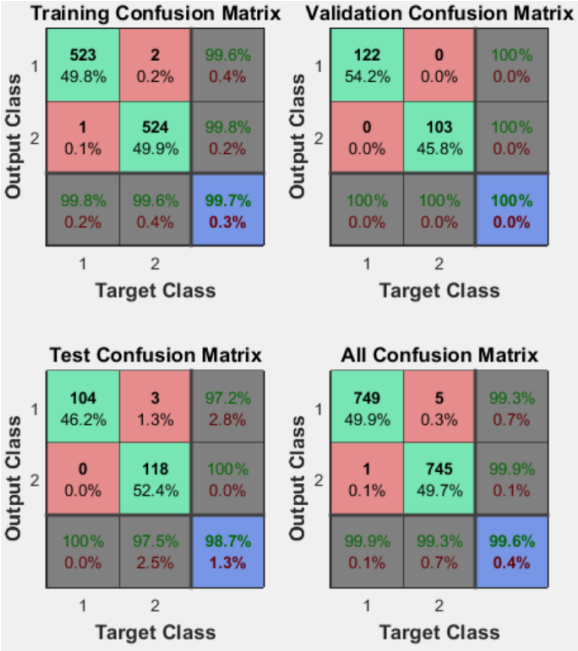


Figura 3.10: Captura de pantalla la matriz de confusión que otorga como resultado la herramienta *nprtool*.

3.3.5. Red Neuronal en vectores de 9 dimensiones

Las particiones de la región de interés siguen siendo sólo tres, para obtener las 9 componentes del vector se calcula también la curtosis y la entropía de cada partición. Definimos la entropía informática o entropía de Shannon como una medida de qué tanta información contiene una variable. Por ejemplo, si consideramos dos imágenes del mismo tamaño, una de ellas de un solo color y la otra de un paisaje, obviamente la imagen del paisaje tiene mayor información que la otra, en otras palabras su entropía es mayor. La entropía se calcula con la ecuación (3.3.2). Volviendo al tema, cada partición aporta 3 componentes al vector $\overrightarrow{V_{9dim}}$, siendo el orden seleccionado: desviación estándar, curtosis y entropía, al tener tres particiones que aportan estas medidas estadísticas, como resultado tenemos un vector de 9 dimensiones el cual será utilizado como entrada para la red neuronal. Los parámetros de la herramienta de MATLAB permanecen tal como fueron explicados previamente.

$$E_{shannon} = -\sum_{i} s_i^2 log(s_i^2) \tag{3.3.2}$$

Capítulo 4

Tablas y Resultados

4.1. Desempeño de las distintas metodologías propuestas

Las metodologías propuestas para la clasificación son aplicadas a una colección de señales de corriente obtenidas de experimentos utilizando motores de inducción con 50 % y 75 % de su capacidad de carga máxima, dichas señales son tomadas de motores con daño en una barra y sanos. La manera de medir el desempeño de cada metodología de clasificación propuesta es calcular su exactitud. Los puntos sanos correctamente clasificados son etiquetados como verdaderos positivos (VP), mientras que los puntos dañados correctamente clasificados son etiquetados como verdaderos negativos (VN). La exactitud del sistema es medida con la ecuación (4.1.1), donde N_s y N_d son el total de experimentos sanos y dañados, respectivamente.

$$Exactitud = \frac{VP + VN}{N_s + N_d} \tag{4.1.1}$$

Para probar el desempeño las metodologías que utilizan un umbral, línea o plano, se calculó la exactitud de un grupo de prueba con 100 experimentos (50 sanos y 50 dañados). Esto fue repetido 10 veces a manera de reportar un promedio, entonces el número total de experimentos es 1000. Cada experimento es llevado a cabo con un kernel distinto tomado de las señales de corriente de la fase del motor. Resultados para señales con 50 % y 75 % de carga fueron obtenidos. En el caso de las redes neuronales, se utilizó un número distinto de experimentos para evaluar el desempeño, pero utilizando el mismo principio, calcular la exactitud. Así mismo se agregó una condición extra de daño del motor para probar la eficiencia de la red en distintas

condiciones. Las condiciones con las que se trabaja son 0mm(Sano), 5mm y 10mm de daño. Las redes neuronales son utilizadas en tres casos de clasificación, para cada uno de ellos la red fue entrenada 10 veces, tomando en cada ocasión un distinto 70 % de los datos de entrada para entrenar. La red neuronal con el mejor desempeño en las 10 veces que se entrenó es seleccionada para realizar una clasificación con cuatro grupos más de vectores y con el grupo que fue entrenada, en total 5 grupos de vectores son utilizados para reportar un promedio. Los casos de clasificación donde se utilizan redes neuronales son explicados a continuación. El primer caso consiste en clasificar vectores provenientes de señales de 0mm y de 10mm de daño, para este caso 750 vectores de cada condición son creados para utilizarlos como entradas. El segundo caso tiene como vectores de entrada vectores de 0mm, 5mm y 10mm de daño, donde el clasificador no discrimina muestras de 5mm de daño de las muestras de 10mm, esto quiere decir que la clasificación se lleva a cabo determinando si el vector de entrada corresponde a un vector sano o uno dañado independientemente de si es daño de 5mm o de 10mm. El banco de muestras utilizado para entrenar y probar la red neuronal del segundo caso es más grande, se crean 750 vectores de cada condición, teniendo así un total de 2250 vectores que serán utilizados para entrenar y probar la red neuronal. Por último, el tercer caso de igual manera tiene como entradas vectores de 0mm, 5mm y 10mm, pero este tiene la diferencia de que discrimina el daño de 5mm del de 10mm. El set de vectores de entrada son 750 vectores de cada condición, dando un total de 2250 vectores. Los resultados para las metodologías que utilizan umbral, línea o plano para la clasificación se muestran en la tabla 4.1, considerando señales de 50 % de carga y 75 %. De la misma forma los resultados de evaluar el desempeño de las redes neuronales en los tres casos explicados son expuestos en la tabla 4.2, con señales de 50 % y 75 % de carga.

Observando las tablas de resultados se puede concluir que utilizando la red neuronal para clasificar entre sano y barra rota los resultados son superiores que a un simple umbral. Sin embargo, el desempeño de la red neuronal en los demás casos comienza a ser deficiente. Centrándonos primeramente en las señales de 75 % de carga, en el caso 3 tenemos ese descenso en la exactitud, esto radica en que las muestras dañadas, tanto de 5mm como de 10mm, comienzan a ser semejantes por lo que el clasificador tiene problemas al diferenciar de qué estado de daño proviene el vector de muestra, porque claramente en el caso dos sin discriminar la gravedad de la falla los resultados son muy satisfactorios. Ahora bien, en el caso de las señales de 50 % de carga, las

Tabla 4.1: Exactitud de las metodologías que clasifican mediante umbrales, con vectores de una, dos y tres dimensiones. La exactitud es mostrada para señales de 50 % de carga y 75 %.

	50 % de carga			75 % de carga		
Grupo de Señales	1D	2D	3D	1D	2D	3D
1	0.91	0.91	0.92	0.94	0.99	0.98
2	0.95	0.95	0.88	0.94	0.96	0.96
3	0.93	0.88	0.91	0.97	0.96	0.96
4	0.95	0.91	0.92	0.95	0.97	0.99
5	0.91	0.94	0.93	0.95	0.97	0.99
6	0.93	0.90	0.87	0.93	0.93	1
7	0.91	0.94	0.97	0.93	0.96	0.95
8	0.93	0.91	0.86	0.96	0.95	0.93
9	0.92	0.93	0.89	0.92	0.97	0.98
10	0.95	0.96	0.92	0.96	0.97	0.98
Promedio	0.929	0.923	0.907	0.945	0.963	0.972

Tabla 4.2: Desempeño de las redes neuronales para vectores de 3 dimensiones, clasificando dos y tres condiciones del motor. La tabla contiene la exactitud para señales de 50 % de carga y 75 %.

	50 % de carga			75 % de carga		
Grupo de Vectores	Caso 1	Caso 2	Caso 3	Caso 1	Caso 2	Caso 3
1	0.975	0.78	0.777	0.993	0.985	0.865
2	0.969	0.762	0.744	0.994	0.987	0.872
3	0.971	0.761	0.771	0.992	0.986	0.860
4	0.970	0.763	0.769	0.992	0.985	0.868
5	0.983	0.761	0.771	0.994	0.987	0.877
Promedio	0.974	0.766	0.766	0.993	0.986	0.869

Tabla 4.3: Desempeño de redes neuronales para vectores de 9 dimensiones. La tabla contiene la exactitud para señales de 50 % de carga y 75 %.

	50 % de carga			75 % de carga		
Grupo de Vectores	Caso 1	Caso 2	Caso 3	Caso 1	Caso 2	Caso 3
1	0.995	0.844	0.845	1	1	0.921
2	0.998	0.819	0.82	1	0.999	0.935
3	0.993	0.83	0.835	0.999	0.999	0.926
4	0.996	0.827	0.818	0.999	1	0.931
5	0.999	0.832	0.824	1	1	0.919
Promedio	0.996	0.83	0.829	0.999	0.999	0.927

muestras sanas y las dañadas son muy cercanas por lo que la clasificación comienza a complicarse, esta conclusión de que el problema radica en la confusión de 0mm y 5mm es deducida de los valores de exactitud que se obtienen en el caso dos, ya que el caso anterior donde separaba barra rota de motor sano la clasificación es muy buena. Observando la deficiencia de la red utilizando vectores de dimensión 3 se opta por realizar estos 3 casos de clasificación con vectores de 9 dimensiones, explicados en el apartado de metodología. Los resultados para vectores de 9 dimensiones se exponen en la tabla 4.3.

Los resultados obtenidos con la clasificación de vectores de 9 dimensiones son claramente mejores. Sin embargo, se sigue teniendo un déficit en la clasificación del caso 3, donde las muestras provenientes de 50 % de carga de 5mm de daño y de motor sano siguen siendo confundidas por la red. A pequeña conclusión diría que trabajando con señales provenientes de motores con 75 % de carga la clasificación es satisfactoria exceptuando el caso 3. Sin embargo, las señales de 50 % de carga solo podríamos decir con exactitud si la barra está completamente rota o es motor sano, con riesgo de tener alguna barra en proceso de ruptura. No es posible trabajar con señales de 50 % de carga con una buena exactitud.
Con el objetivo de visualizar de una manera más fácil los resultados de las metodologías expuestos en tablas, en las figuras Fig.4.1, Fig.4.2 y Fig. podemos observar la exactitud graficada contra el número de grupos de muestra reportados en cada una de las metodologías de clasificación.

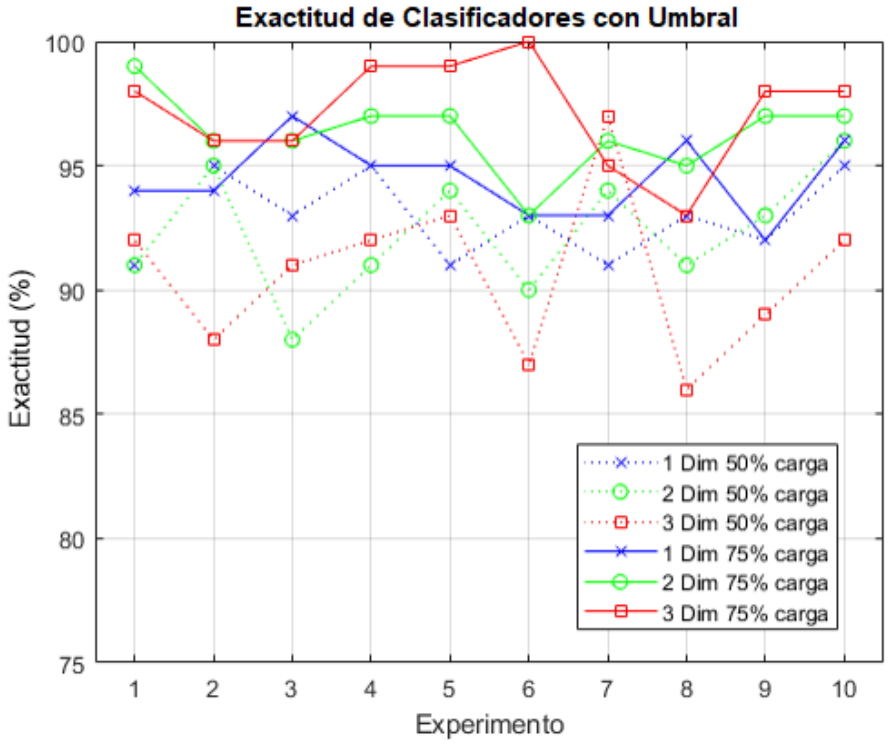


Figura 4.1: Clasificación con Umbral. Resultados de la tabla 4.1 expuestos de forma gráfica.

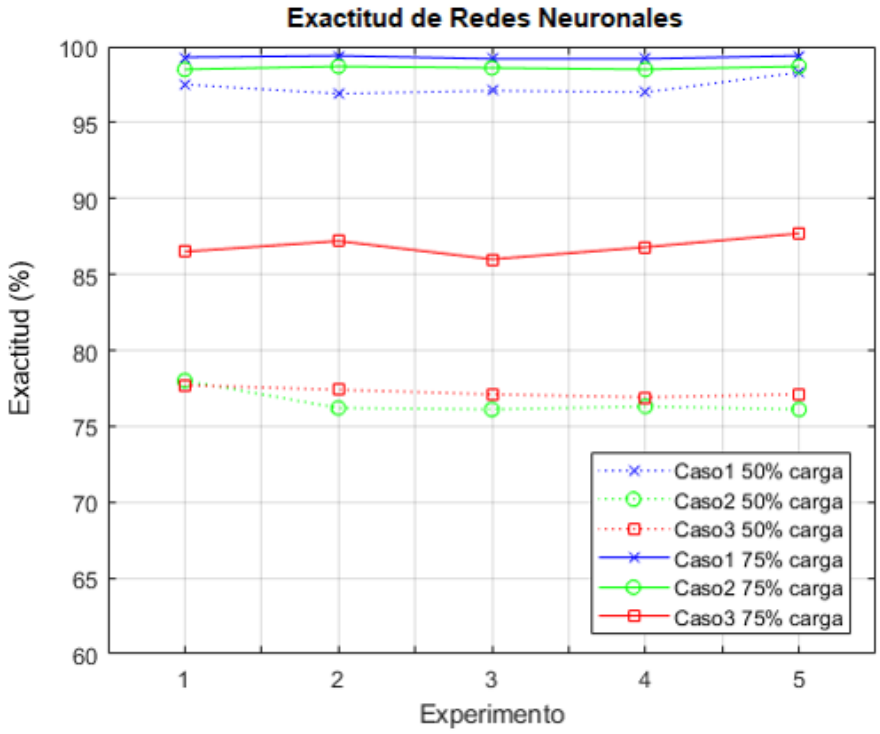


Figura 4.2: Clasificación de vectores de 3 dimensiones con redes neuronales. Resultados de la tabla 4.2 expuestos de forma gráfica.

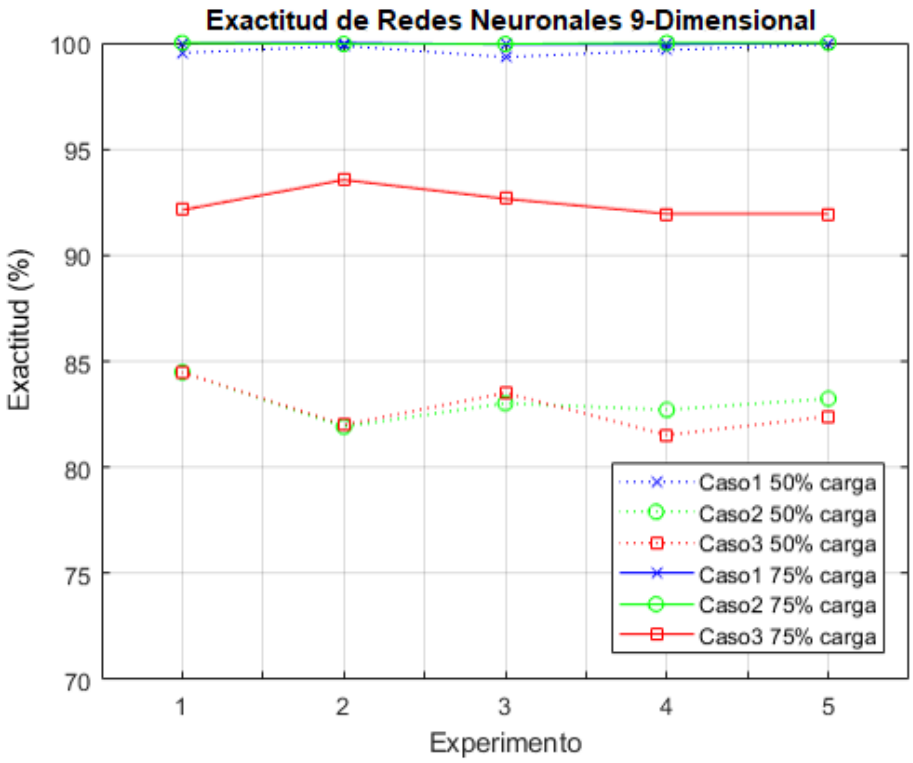


Figura 4.3: Clasificación de vectores de 9 dimensiones con redes neuronales. Resultados de la tabla 4.3 expuestos de forma gráfica.

4.2. Vivado Software

Vivado Design suite es un sotfware desarrollado con Xilinx para sintetizar y analizar diseños HDL. Vivado también incluye Vivado HLS (por sus siglas en inglés *High Level Syntesis*) que es una herramienta para sintetizar y convertir código escrito en el lenguaje C/C++ en código lógico programable, dígase VHDL o Verilog [24].
Una de las mayores ventajas que ofrece este software es la facilidad para el programador de implementar código de C/C++ en FPGA's (*Field Programmable Gate Array*) sin la complicación de manejar bit por bit como se hace trabajando directamente con VHDL o Verilog. Vivado HLS tiene soporte para muchas tarjetas en las que incluye Virtex 7, Kintex 7 y Spartan 6.

4.3. Algoritmo y Simulación

El algoritmo que se programó para la simulación es mostrado en la Fig.4.4, haciendo énfasis en el algoritmo Fast-ICA en la Fig.4.5. El lenguaje de codificación utilizado es C/C++. Para la simulación se selecciona un FPGA de la familia kintex7, el seleccionado fue la tarjeta "xc7k325tffg900-2".

Los recursos utilizados al momento de realizar la síntesis del código se muestran en la Fig. 4.6, donde especifica el número de componentes utilizados, así como los disponibles en la tarjeta.
Para los valores de los pesos W_i y los sesgos θ_k de la red neuronal, se utilizaron valores exportados por MATLAB, es decir, la red fue entrenada de manera ajena a la implementación. Las funciones de activación de la red simulada son sigmoides, tanto en la capa oculta como en la salida.

Para la simulación, Vivado HLS lee dos señales guardados en archivos con extensión ".dat" correspondientes a espectros de frecuencia de la señal de corriente y de su autocorrelación. Como resultado de la simulación tenemos la clasificación de la red (Fig. 4.7).

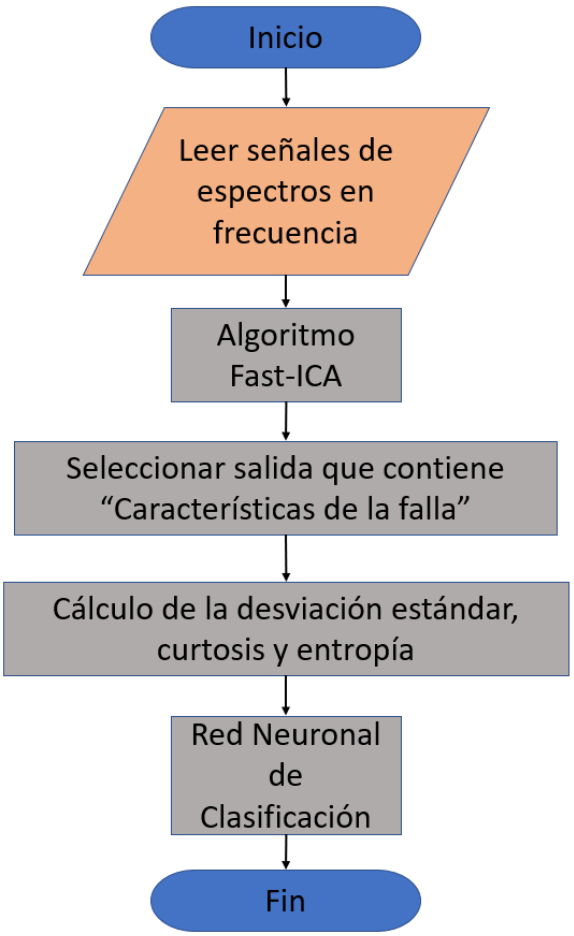


Figura 4.4: Diagrama de flujo del algoritmo programado en Vivado HLS.

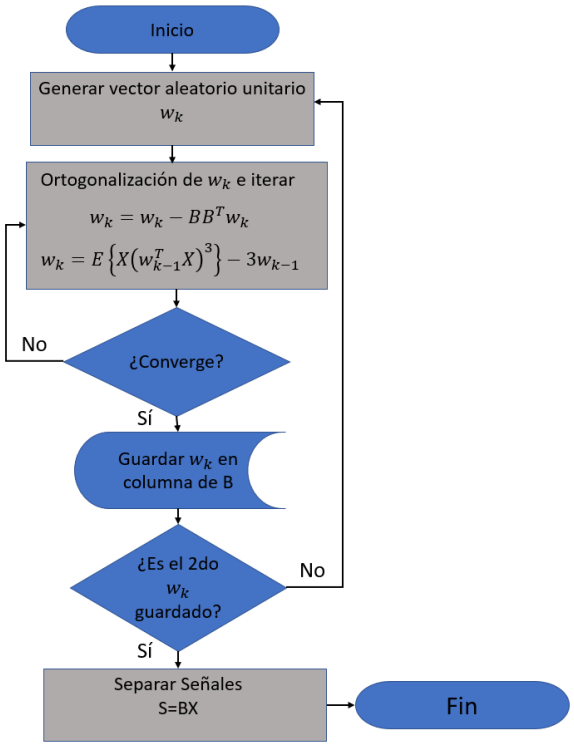


Figura 4.5: Diagrama de flujo del algoritmo Fast-ICA programado en Vivado HLS. Donde X es la matriz que contiene las señales mezcladas; S es la matriz con las señales separadas; B es la matriz de separación.

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	3423
FIFO	-	-	-	-
Instance	-	58	6620	11538
Memory	6	-	32	9
Multiplexer	-	-	-	6270
Register	-	-	5879	-
Total	6	58	12531	21240
Available	890	840	407600	203800
Utilization (%)	~0	6	3	10

Figura 4.6: Tabla con los recursos requeridos para la implementación del código en Vivado HLS.

```
1   Compiling ../../../../testBench.cpp in debug mode
2   Compiling ../../../../Funciones.cpp in debug mode
3   Generating csim.exe
4 5.89066e-005,0.999967
5 Motor Sano
6 INFO: [SIM 1] CSim done with 0 errors.
7
```

Figura 4.7: Línea de comandos de Vivado HLS donde muestra los resultados de la simulación. Donde la salida es [0,1] en caso de ser sano y [1,0] en caso de ser dañado.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

Una metodología de detección de fallas en motores de inducción del tipo jaula de ardilla fue descrita. Con la ayuda de la técnica ICA se efectúa la separación de las características espectrales de la falla para posteriormente realizar la debida clasificación del estado del motor. Para la detección de una barra rota, el mejor desempeño de la metodología fue visto con señales provenientes de motores con 75 % de carga, sin embargo, la clasificación también es posible con señales provenientes de motores con 50 % de carga. La clasificación con el uso de redes neuronales es claramente superior al uso de umbrales promedio, es por eso que se tomó como metodología definitiva el uso de redes neuronales. Al detectar la ruptura completa de la barra la metodología no tiene problema alguno ya que tanto en señales de 50 % de carga como de 75 % la exactitud promedio obtenida en las pruebas es del 99 %. Sin embargo, las redes neuronales también son capaces de discriminar cuando la barra está a la mitad de su ruptura, siendo estos casos los reportados con menor exactitud. Por ese motivo se llega a la conclusión de utilizar el método a manera de detectar la ruptura completa de una barra, es decir, el caso 1, y asegurar la correcta clasificación en las dos condiciones de carga trabajadas.

5.2. Trabajo futuro

Ya se desarrolló el código de la metodología en el lenguaje de programación C, para ser implementado con el software de Xilinx, Vivado HLS. Sin embargo, el proyecto no llegó a la implementación directa. Se tiene como meta a futuro implementar la

metodología a tiempo real para la detección de fallas en un motor que esté instalado y desarrollando su función.
A manera de propuesta se puede trabajar con más variedad de cargas del motor y tratar de identificar otro tipo de parámetros que nos ayuden a la clasificación para mejorar la exactitud en detectar media barra rota.

5.3. Artículos publicados

Una artículo derivado de este trabajo de tesis fue publicado en la conferencia *International Instrumentation & Measurement Technology conference* en el presente año (I2MTC 2018), el cual expone los primeros pasos de esta metodología exponiendo una correcta detección de fallas del 95 % [25]. El artículo publicado se titula "*Induction motors fault detection using independent component analysis on phase current signals*", el cual dio paso a exponer un póster en la conferencia antes mencionada en Mayo del 2018. Un artículo extendido ya fue enviado a los revisores para una posible publicación, siendo ésta la segunda publicación derivada de este trabajo.

Apéndices

Apéndice

Código C++ de la implementación

TestBench

```
#include <iostream>
#include <math.h>
#include <C:\Users\JohnyTesla\Dropbox\TesisPCAICA\DevC\Libraries
\EigenLib/Eigen/Eigenvalues>
#include "Funciones.h"
#include <stdlib.h>
#include<ctime>
#include<cstdlib>
using namespace Eigen;
using namespace std;

int main() {
float SalidaNN[2]={0};
float S1[tamano]={
#include "Signal_1.dat"
};
float S2[tamano]={
#include "Signal_2.dat"
};
Clasificador(S1,S2,SalidaNN);
cout<<SalidaNN[0]<<","<<SalidaNN[1]<<endl;
if(SalidaNN[0]<SalidaNN[1]){
cout<<"Motor Sano"<<endl;
```

```
}else{
cout<<"Motor Dañado"<<endl;
}
return 0;
}
```

Funciones

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "Funciones.h"
#include <C:\Users\JohnyTesla\Dropbox\TesisPCAICA\DevC\Libraries\
EigenLib/Eigen/Eigenvalues>
#include <stdlib.h> /* abs */
#include <ctime>
#include <cstdlib>
using namespace Eigen;
using namespace std;

void Clasificador(float vA[tamano],float vB[tamano],float out2[dimen]){
int i,k,t,j,ics,flagSalida;
float R[dimen],IC[dimen][tamano],vw[dimen],vaux[dimen],wold[dimen]={0},NNin[9]={0};
float Xm=0,Ym=0,acc=0,normaW=0,acumulado=0,acumulado1=0,acumulado2=0,acumulado3=0;
float stdTic1=0,stdTic2=0,std1=0,std2=0,std3=0,Xm1=0,Ym1=0,Xm2=0,Ym2=0;
float entro1=0,entro2=0,entro3=0;
float DV[dimen][dimen],Z[dimen][tamano],Xc[dimen][tamano];
float mB[dimen][dimen]={0,0},Taux[dimen][dimen],Wm[dimen][dimen]={0,0};
float MLL[2][2],QQ[2][2],RR[2][2],QR[2][2],ec1[2],ttt[2][2],aa2[2],uu1[2],uu2[2];
float WhiteM[2][2],LL[2],cc1=0,e1[2][2];
// Calcular media de las señales leidas
for (j=0;j<tamano;j++){
  Xm=Xm+vA[j];
  Ym=Ym+vB[j];
}
```

```
Xm=Xm/tamano; Ym=Ym/tamano;
//Centrar dataset
for (j=0;j<tamano;j++){
vA[j]=vA[j]-Xm;
vB[j]=vB[j]-Ym;
}
//calcular matriz de covarianza
for(int t=0;t<tamano;t++){
acumulado =acumulado +vA[t]*vA[t];
acumulado1=acumulado1+vA[t]*vB[t];
acumulado2=acumulado2+vB[t]*vA[t];
acumulado3=acumulado3+vB[t]*vB[t];
}
ttt[0][0]=acumulado/tamano;
ttt[0][1]=acumulado1/tamano;
ttt[1][0]=acumulado2/tamano;
ttt[1][1]=acumulado3/tamano;
//Calcular eigenvalores
aa2[0]=ttt[0][1];aa2[1]=ttt[1][1];
RR[0][0]=0;RR[0][1]=0;RR[1][0]=0;RR[1][1]=0;
QQ[0][0]=0;QQ[0][1]=0;QQ[1][0]=0;QQ[1][1]=0;
uu1[0]=ttt[0][0];uu1[1]=ttt[1][0];//aa1[0];uu1[1]=aa1[1];
acc=(aa2[0]*uu1[0]+aa2[1]*uu1[1])/(uu1[0]*uu1[0]+uu1[1]*uu1[1]);
uu2[0]=aa2[0]-acc*uu1[0]; uu2[1]=aa2[1]-acc*uu1[1];
for(j=0;j<dimen;j++){
normaW=normaW+uu1[j]*uu1[j];
}
normaW=sqrt(normaW);
RR[0][0]=normaW;
normaW=0;
for(j=0;j<dimen;j++){
normaW=normaW+uu2[j]*uu2[j];
}
normaW=sqrt(normaW);
```

```
RR[1][1]=normaW;
QQ[0][0]=uu1[0]/RR[0][0]; QQ[1][0]=uu1[1]/RR[0][0];
QQ[0][1]=uu2[0]/RR[1][1]; QQ[1][1]=uu2[1]/RR[1][1];
RR[0][1]=(QQ[0][0]*aa2[0]+QQ[1][0]*aa2[1]);
for(int i=0;i<dimen;i++){
for(int k=0;k<dimen;k++){
QR[i][k]=RR[i][0]*QQ[0][k]+RR[i][1]*QQ[1][k];
}
}
LL[0]=QR[0][0]; LL[1]=QR[1][1];
// cout<<LL[0]<<","<<LL[1]<<endl;
//Calcular eigenvectores
for (i=0;i<dimen;i++){
aa2[0]=LL[i]-ttt[0][0]; aa2[1]=-ttt[0][1];
cc1=-aa2[1]/aa2[0];
e1[i][0]=cc1;e1[i][1]=1;
normaW=0;
for(j=0;j<dimen;j++){
normaW=normaW+e1[i][j]*e1[i][j];
}
normaW=sqrt(normaW);
for(j=0;j<dimen;j++){
e1[i][j]=-e1[i][j]/normaW;
}
// cout<<LL[i]<<endl;
// cout<<e1[i][0]<<","<<e1[i][1]<<endl;
} //lo guarda transpuesto directamente
// inversa de la matriz de eigenvalores
QQ[0][0]=LL[0];QQ[0][1]=0;QQ[1][0]=0;QQ[1][1]=LL[1];
// cout<<QQ[0][0]<<"  "<<QQ[0][1]<<endl;
// cout<<QQ[1][0]<<"  "<<QQ[1][1]<<endl;
normaW=QQ[0][0]*QQ[1][1]-QQ[1][0]*QQ[0][1];
MLL[0][0]=sqrt(QQ[1][1]/normaW);
MLL[1][1]=sqrt(QQ[0][0]/normaW);
```

```
MLL[1][0]=0;MLL[0][1]=0;
// cout<<MLL[0][0]<<"  "<<MLL[0][1]<<endl;
// cout<<MLL[1][0]<<"  "<<MLL[1][1]<<endl;
for(int i=0;i<dimen;i++){
for(int k=0;k<dimen;k++){
QR[i][k]=MLL[i][0]*e1[0][k]+MLL[i][1]*e1[1][k];
}
}
// cout<<WhiteM[0][0]<<"  "<<WhiteM[0][1]<<endl;
// cout<<WhiteM[1][0]<<"  "<<WhiteM[1][1]<<endl;
//Calcular datos blanqueados
for(int i=0;i<dimen;i++){
for(int k=0;k<tamano;k++){
Z[i][k]=QR[i][0]*vA[k]+QR[i][1]*vB[k];
}
}
for(int i=0;i<tamano;i++){
vA[i]=Z[0][i];
vB[i]=Z[1][i];
}
/* Algoritmo FastICA  */
for(ics=0;ics<dimen;ics++){
/*srand( unsigned(time(NULL)));
for(i=0;i<dimen;i++){
vw[i]=(float)rand()/(RAND_MAX + 1)+(rand()%1);
}*/
vw[0]=2;vw[1]=2.64;
normal(vw);
// cout<<"vector random"<<vw[0]<<",,,"<<vw[1]<<endl;
for (int k=0;k<10;k++){
//ortogonaluizacion y normalizar
for (i=0;i<dimen;i++){
for (j=0;j<dimen;j++){
Taux[i][j]=mB[j][0]*mB[i][0]+mB[j][1]*mB[i][1];
```

```
}
}
for (j=0;j<dimen;j++){
   vaux[j]=vw[j]-Taux[j][0]*vw[0]-Taux[j][1]*vw[1];
}
normal(vaux);
for(j=0;j<dimen;j++){
vw[j]=vaux[j];
}
            //producto punto y condicion de convergencia
acc=0;
for (int i=0;i<dimen;i++){
acc=acc+vw[i]*wold[i];
}
if (acc<0){
acc=-acc;
}
if (acc==0){
acc=acc+1.5;
}
for (int j=0;j<dimen;j++){
wold[j]=vw[j];
}
acc=acc-1;
if (acc<0){
acc=-acc;
}
if ((acc)<0.001){
mB[0][ics]=vw[0]; mB[1][ics]=vw[1];
Wm[ics][0]=vw[0]; Wm[ics][1]=vw[1];
wold[0]=0; wold[1]=0;
if (ics==1){
flagSalida=1;
}
```

```
break;
}
iterar(vA,vB,vw);
}
if (flagSalida==1){
break;
}
}
for (i=0;i<dimen;i++){
for(j=0;j<tamano;j++){
IC[i][j]=abs(Wm[i][0]*Z[0][j]+Wm[i][1]*Z[1][j]);
}
}
/* aqui iba la impreision de los ICS*/
/* Calculo de STD de las caracteristicas extraidas */
Xm=0;Ym=0;acumulado=0;acumulado1=0;
for (j=0;j<30;j++){
  Xm=Xm+IC[0][j];
  Ym=Ym+IC[1][j];
  Xm1=Xm1+IC[0][30+j];
  Ym1=Ym1+IC[1][30+j];
  Xm2=Xm2+IC[0][60+j];
  Ym2=Ym2+IC[1][60+j];
}
Xm=Xm/30; Ym=Ym/30;
Xm1=Xm1/30; Ym1=Ym1/30;
Xm2=Xm2/30; Ym2=Ym2/30;
for (j=0;j<30;j++){
acumulado=acumulado+(IC[0][j]-Xm)*(IC[0][j]-Xm);
acumulado1=acumulado1+(IC[1][j]-Ym)*(IC[1][j]-Ym);
}   /* std (1:30) de ambas señales  */
stdTic1=sqrt(acumulado/29);
stdTic2=sqrt(acumulado1/29);
if (stdTic1<stdTic2){  //tomar la señal con menor std
```

```
NNin[0]=stdTic1;
std1=sqrt(acumulado/30);
acumulado=0;acumulado1=0;
for(j=0;j<30;j++){
acumulado=acumulado+(IC[0][30+j]-Xm1)*(IC[0][30+j]-Xm1);
acumulado1=acumulado1+(IC[0][60+j]-Xm2)*(IC[0][60+j]-Xm2);
NNin[6]=NNin[6]-(IC[0][j]*IC[0][j])*log(IC[0][j]*IC[0][j]);
NNin[7]=NNin[7]-(IC[0][30+j]*IC[0][30+j])*log(IC[0][30+j]*IC[0][30+j]);
NNin[8]=NNin[8]-(IC[0][60+j]*IC[0][60+j])*log(IC[0][60+j]*IC[0][60+j]);
}
std2=sqrt(acumulado/30);std3=sqrt(acumulado1/30);
NNin[1]=sqrt(acumulado/29); NNin[2]=sqrt(acumulado1/29);
acumulado=0;acumulado1=0;acumulado2=0;
for(j=0;j<30;j++){
acumulado=acumulado+(IC[0][j]-Xm)*(IC[0][j]-Xm)*(IC[0][j]-Xm)*(IC[0][j]-Xm);
acumulado1=acumulado1+(IC[0][30+j]-Xm1)*(IC[0][30+j]-Xm1)*
(IC[0][30+j]-Xm1)*(IC[0][30+j]-Xm1);
acumulado2=acumulado2+(IC[0][60+j]-Xm2)*(IC[0][60+j]-Xm2)*
(IC[0][60+j]-Xm2)*(IC[0][60+j]-Xm2);
}
NNin[3]=acumulado/(30*std1*std1*std1*std1);
NNin[4]=acumulado1/(30*std2*std2*std2*std2);
NNin[5]=acumulado2/(30*std3*std3*std3*std3);
}else{
NNin[0]=stdTic2;
    std1=sqrt(acumulado1/30);
acumulado=0;acumulado1=0;
for(j=0;j<30;j++){
acumulado=acumulado+(IC[1][30+j]-Ym1)*(IC[1][30+j]-Ym1);
acumulado1=acumulado1+(IC[1][60+j]-Ym2)*(IC[1][60+j]-Ym2);
NNin[6]=NNin[6]-(IC[1][j]*IC[1][j])*log(IC[1][j]*IC[1][j]);
NNin[7]=NNin[7]-(IC[1][30+j]*IC[1][30+j])*log(IC[1][30+j]*IC[1][30+j]);
NNin[8]=NNin[8]-(IC[1][60+j]*IC[1][60+j])*log(IC[1][60+j]*IC[1][60+j]);
}
```

```
std2=sqrt(acumulado/30);std3=sqrt(acumulado1/30);
NNin[1]=sqrt(acumulado/29); NNin[2]=sqrt(acumulado1/29);
acumulado=0;acumulado1=0;acumulado2=0;
for(j=0;j<30;j++){
acumulado=acumulado+(IC[1][j]-Ym)*(IC[1][j]-Ym)*(IC[1][j]-Ym)*(IC[1][j]-Ym);
acumulado1=acumulado1+(IC[1][30+j]-Ym1)*(IC[1][30+j]-Ym1)*
(IC[1][30+j]-Ym1)*(IC[1][30+j]-Ym1);
acumulado2=acumulado2+(IC[1][60+j]-Ym2)*(IC[1][60+j]-Ym2)*
(IC[1][60+j]-Ym2)*(IC[1][60+j]-Ym2);
}
NNin[3]=acumulado/(30*std1*std1*std1*std1);
NNin[4]=acumulado1/(30*std2*std2*std2*std2);
NNin[5]=acumulado2/(30*std3*std3*std3*std3);
}
/* apartado de la Red Neuronal*/
int N=9,L=2,M=2;
float w1[L][N],w2[M][L],b1[L],b2[M];
float net1[L],net2[M],out1[L]; //out2[M]
//Valores de w1 y w2 para red entrenada cpara separar 0mm de 10mm 75% de carga
w1[0][0]=-0.1560;w1[0][1]=-8.6500;w1[0][2]=-11.5058;
w1[0][3]=0.03150;w1[0][4]=0.23700;w1[0][5]=0.234800;
w1[0][6]=0.00890;w1[0][7]=0.03280;w1[0][8]=0.344900;
w1[1][0]=0.3624;w1[1][1]=1.5812;w1[1][2]=3.0664;
w1[1][3]=1.9411;w1[1][4]=2.5282;w1[1][5]=2.6722;
w1[1][6]=-0.0861;w1[1][7]=0.7593;w1[1][8]=3.3837;
w2[0][0]=-23.0043;w2[0][1]=-1.8836;
w2[1][0]=22.8346;w2[1][1]=0.0912;
b1[0]=10.6002;b1[1]=0.6286;
b2[0]=13.2814;b2[1]=-10.7532;
//capa oculta
for(i=0;i<L;i++){
net1[i]=0;
for(j=0;j<N;j++){
net1[i]=net1[i]+w1[i][j]*NNin[j];
```

```
}
net1[i]=net1[i]+b1[i];
out1[i]=1/(1+exp(-net1[i]));
}
//capa salida
for (i=0;i<M;i++){
net2[i]=0;
for(j=0;j<L;j++){
net2[i]=net2[i]+w2[i][j]*out1[j];
}
net2[i]=net2[i]+b2[i];
out2[i]=1/(1+exp(-net2[i]));
}
return;
}
void iterar(float A[tamano], float B[tamano],float w[dimen]){
int j;
float R[dimen],C[tamano];
float normaW=0;
/* realizar operacion ans1= (X' w)^3  */
for (j=0;j<tamano;j++){
C[j]=A[j]*w[0]+B[j]*w[1];
C[j]=C[j]*C[j]*C[j];
}
/* realizar operacion  X ans1 - 3 w*/
R[0]=0,R[1]=0;
for (j=0;j<tamano;j++)
{R[0]=R[0]+A[j]*C[j];
 R[1]=R[1]+B[j]*C[j];
}
R[0]=R[0]/tamano-3*w[0]; R[1]=R[1]/tamano-3*w[1];
normal(R); //normalizar vector R
return;
}
```

```
void normal(float vv[dimen]){
// float koko=0;
int j;
float normaW=0;
for(j=0;j<dimen;j++){
normaW=normaW+vv[j]*vv[j];
}
normaW=sqrt(normaW);
for(j=0;j<dimen;j++){
vv[j]=vv[j]/normaW;
}
}
```

Bibliografía

[1] Ricardo Valles-Novo, Jose de Jesus Rangel-Magdaleno, Juan Manuel Ramirez-Cortes, Hayde Peregrina-Barreto, and Roberto Morales-Caporal. Empirical mode decomposition analysis for broken-bar detection on squirrel cage induction motors. *IEEE Transactions on Instrumentation and Measurement*, 64(5):1118–1128, 2015.

[2] William T Thomson and Mark Fenger. Current signature analysis to detect induction motor faults. *IEEE Industry Applications Magazine*, 7(4):26–34, 2001.

[3] Israel Cruz-Vega, Jose Rangel-Magdaleno, Juan Ramirez-Cortes, and Hayde Peregrina-Barreto. Automatic progressive damage detection of rotor bar in induction motor using vibration analysis and multiple classifiers. *Journal of Mechanical Science and Technology*, 31(6):2651–2662, 2017.

[4] Jafar Zarei and Javad Poshtan. Bearing fault detection using wavelet packet transform of induction motor stator current. *Tribology International*, 40(5):763–769, 2007.

[5] Jose Rangel-Magdaleno, Hayde Peregrina-Barreto, Juan Ramirez-Cortes, Roberto Morales-Caporal, and Israel Cruz-Vega. Vibration analysis of partially damaged rotor bar in induction motor under different load condition using dwt. *Shock and Vibration*, 2016, 2016.

[6] Jose de Jesus Rangel-Magdaleno, Rene de Jesus Romero-Troncoso, Roque Alfredo Osornio-Rios, Eduardo Cabal-Yepez, and Luis Miguel Contreras-Medina. Novel methodology for online half-broken-bar detection on induction motors. *IEEE Transactions on Instrumentation and Measurement*, 58(5):1690–1698, 2009.

[7] B Samanta and KR Al-Balushi. Artificial neural network based fault diagnostics of rolling element bearings using time-domain features. *Mechanical systems and signal processing*, 17(2):317–328, 2003.

[8] Dragan Matic, Filip Kulic, Manuel Pineda-Sanchez, and Joan Pons-Llinares. Artificial neural networks eccentricity fault detection of induction motor. In *Computing in the Global Information Technology (ICCGI), 2010 Fifth International Multi-Conference on*, pages 1–4. IEEE, 2010.

[9] Fernando M Janeiro, João F Martins, V Fernao Pires, Pedro M Ramos, and Armando J Pires. Induction motor broken bars online detection. In *Instrumentation and Measurement Technology Conference Proceedings, 2008. IMTC 2008.*, pages 2137–2140. IEEE, 2008.

[10] Bo Liang, Simon D Iwnicki, and Andrew D Ball. Asymmetrical stator and rotor faulty detection using vibration, phase current and transient speed analysis. *Mechanical systems and signal processing*, 17(4):857–869, 2003.

[11] Jose Rangel-Magdaleno, Hayde Peregrina-Barreto, Juan Ramirez-Cortes, and Israel Cruz-Vega. Hilbert spectrum analysis of induction motors for the detection of incipient broken rotor bars. *Measurement*, 2017.

[12] Fang Duan, Michael Corsar, Linghao Zhou, and David Mba. Using independent component analysis scheme for helicopter main gearbox bearing defect identification. In *Prognostics and Health Management (ICPHM), 2017 IEEE International Conference on*, pages 252–259. IEEE, 2017.

[13] Z Wang and CS Chang. Online fault detection of induction motors using frequency domain independent components analysis. In *Industrial Electronics (ISIE), 2011 IEEE International Symposium on*, pages 2132–2137. IEEE, 2011.

[14] Ting Yang, Haibo Pen, Zhaoxia Wang, and Che Sau Chang. Feature knowledge based fault detection of induction motors through the analysis of stator current data. *IEEE Transactions on Instrumentation and Measurement*, 65(3):549–558, 2016.

[15] Achmad Widodo, Bo-Suk Yang, and Tian Han. Combination of independent

component analysis and support vector machines for intelligent faults diagnosis of induction motors. *Expert systems with applications*, 32(2):299–312, 2007.

[16] Aapo Hyvarinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural computation*, 9(7):1483–1492, 1997.

[17] Aapo Hyvärinen and Erkki Oja. One-unit learning rules for independent component analysis. In *Advances in neural information processing systems*, pages 480–486, 1997.

[18] Simon Haykin and Zhe Chen. The cocktail party problem. *Neural computation*, 17(9):1875–1902, 2005.

[19] Jimena Blaiotta and Pablo Delieutraz. Teorema central del límite. *Universidad de Buenos Aires, Facultad de Ciencias exactas*, 2004.

[20] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

[21] Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, pages 1–6, 2018.

[22] Pedro Larranaga, Inaki Inza, and Abdelmalik Moujahid. Tema 8. redes neuronales. *Redes Neuronales, U. del P. Vasco*, pages 12–17, 1997.

[23] John Odland. Spatial autocorrelation. Technical report, Regional Research Institute, West Virginia University, 1985.

[24] Tom Feist. Vivado design suite. *White Paper*, 5, 2012.

[25] Juan Garcia-Bracamonte, Jose de Jesus Rangel-Magdaleno, Juan Manuel Ramirez-Cortes, Pilar Gomez-Gil, and Hayde Peregrina-Barreto. Induction motors fault detection using independent component analysis on phase current signals. *IEEE International Conference*, 2018.