Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

LAS REDES NEURONALES APLICADAS A LA DETECCIÓN DE PATRONES MUSICALES DE LA GUITARRA Y SU TRADUCCIÓN A LA NOTACIÓN MUSICAL DE TABLATURA

Marvin Adolfo Hernández Alonzo
Asesorado por el Ing. Luis Fernando Quiñonez López

Guatemala, noviembre de 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

LAS REDES NEURONALES APLICADAS A LA DETECCIÓN DE PATRONES MUSICALES DE LA GUITARRA Y SU TRADUCCIÓN A LA NOTACIÓN MUSICAL DE TABLATURA

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

MARVIN ADOLFO HERNÁNDEZ ALONZO
ASESORADO POR EL ING. LUIS FERNANDO QUIÑONEZ LÓPEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, NOVIEMBRE DE 2014

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Inga. Elvia Miriam Ruballos Samayoa
VOCAL IV	Br. Narda Lucía Pacay Barrientos
VOCAL V	Br. Walter Rafael Véliz Muñoz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. César Augusto Fernández Fernández
EXAMINADOR	Ing. José Ricardo Morales Prado
EXAMINADOR	Ing. Roberto Estuardo Ruíz Cruz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

LAS REDES NEURONALES APLICADAS A LA DETECCIÓN DE PATRONES MUSICALES DE LA GUITARRA Y SU TRADUCCIÓN A NOTACIÓN MUSICAL DE TABLATURA

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha11 de septiembre del 2013.

Marvin Adolfo Hernández Alonzo

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala,11 de Septiembre de 2013

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **MARVIN ADOLFO HERNÁNDEZ ALONZO** carné **2004-12371,** titulado: **"LAS REDES NEURONALES APLICADAS A LA DETECCIÓN DE PATRONES MUSICALES DE LA GUITARRA Y SU TRADUCCIÓN A NOTACIÓN MUSICAL DE TABLATURA",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"LAS REDES NEURONALES APLICADAS A LA DETECCIÓN DE PATRONES MUSICALES DE LA GUITARRA Y SU TRADUCCIÓN A LA NOTACIÓN MUSICAL DE TABLATURA"****, realizado por el estudiante MARVIN ADOLFO HERNÁNDEZ ALONZO, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 29 de octubre 2014

Guatemala, Agosto del 2013

Ingeniero
Carlos Azurdia
Tutor de trabajos de graduación

Respetable Ingeniero Azurdia:

Por este medio el informo como asesor del trabajo de graduación del estudiante universitario de la carrera de Ingeniería en Ciencias y Sistemas, MARVIN ADOLFO HERNANDEZ ALONZO, carne 200412371, que he revisado el trabajo de graduación titulado: "LAS REDES NEURONALES APLICADAS A LA DETECCION DE PATRONES MUSICALES DE LA GUITARRA Y SU TRADUCCION A NOTACIÓN MUSICAL DE TABLATURA", y a mi criterio el mismo esta completo, actualizado y cumple con los objetivos propuestos para su desarrollo según el protocolo.

Agradeciendo su atención a la presente,

Atentamente,

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Luis Fernando Quiñónez López
Catedrático
Asesor de trabajo de graduación
Colegiado: 7514

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.D.622-2014

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **LAS REDES NEURONALES APLICADAS A LA DETECCIÓN DE PATRONES MUSICALES DE LA GUITARRA Y SU TRADUCCIÓN A LA NOTACIÓN MUSICAL DE TABLATURA,** presentado por el estudiante universitario: **Marvin Adolfo Hernández Alonzo,** después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, se autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Alfredo Enrique Beber Aceituno
Decano en Funciones

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, noviembre de 2014

/cc

ACTO QUE DEDICO A:

Mis padres	Por el apoyo incondicional, amor y confianza brindada que he recibido en todo momento de mi vida.
Mis hermanos	Por el cariño y apoyo que me han dado en cada momento.
Mis amigos	Por los buenos momentos compartidos y las experiencias adquiridas.
Mis compañeros	Por su solidaridad a lo largo de toda la carrera.

AGRADECIMIENTOS A:

Mis padres	Dominga Alonzo y Maximiliano Hernández, por su apoyo incondicional a lo largo de mi vida y mis estudios.
Mis hermanos	Cristian y Yojana Hernández, por ser una importante influencia en mi carrera, entre otras cosas.
Mis amigos	Diego Vazquez, Ruben Guevara, Andie Flores y Alfredo Verganza, por su compañía y apoyo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
@	Arroba
IA	Inteligencia artificial
$\sum$	Sumatoria

GLOSARIO

Ingeniería Conjunto de conocimientos y técnicas científicas las cuales son aplicadas a la creación.

Tablatura Es una forma de escritura musical especial para ciertos instrumentos, y difiere de la notación musical corriente presentando únicamente posiciones y colocaciones en el instrumento para interpretar la pieza musical.

Neurona Son un tipo de células del sistema nervioso cuya principal características es la excitabilidad eléctrica de su membrana plasmática.

Red En ciencias de la computación, es un sistema entre computadoras o elementos computacionales que permite la transmisión de datos entre computadoras o elementos computacionales.

Metodología Conjunto de procedimientos que determinan una investigación o desarrollo de tipo científico o marcan el rumbo de una exposición doctrinal.

RESUMEN

La inteligencia artificial es un campo de informática que trata de la inteligencia de las computadoras y máquinas; este campo se está expandiendo cada vez más en todas las ciencias y en la vida cotidiana.

Las redes neuronales son un paradigma de aprendizaje y procesamiento de información; estas redes de neuronas artificiales tienen múltiples usos. Los principales usos de las redes neuronales son la detección de patrones, y se pueden aplicar a la detección de imágenes, rostros, inteligencia de negocios y otros.

Los patrones musicales son un campo poco explotado debido a la complejidad y precisión que este exige. Este estudio y desarrollo tiene como meta crear una aplicación basada en redes neuronales que detecte los patrones musicales de la guitarra y lo escriba en notación musical de tablatura.

OBJETIVOS

General

Realizar una aplicación para la detección de patrones musicales producidos por los primeros cuatro trastes de una guitarra, que serán escritos en notación musical de tablatura.

Específicos

1. Hacer una investigación acerca de redes neuronales completa que sirva de base para el desarrollo de la aplicación.

2. Utilizar el lenguaje de programación orientado a objetos java.

3. Utilizar las redes neuronales y lógica borrosa para la detección de las nota musicales.

4. Desarrollar una aplicación cercana al tiempo real.

INTRODUCCIÓN

En los últimos tiempos la inteligencia artificial ha tomado importancia resolviendo problemas de forma óptima, problemas de alta complejidad que con otros métodos la forma de resolverlos tomaría grandes recursos computacionales. Actualmente las redes neuronales son utilizadas para resolver problemas en donde la lógica clásica no puede ser utilizada debido a que las entradas de los problemas contienen una gran cantidad de datos con ruido.

Las notas musicales están compuestas por tres propiedades básicas, la frecuencia y tono de la onda de sonido; estos están acompañados por la forma de la onda de sonido con la que se puede diferenciar una nota musical de otra.

La propuesta es una aplicación que detecte los patrones de una onda de sonido de una nota perfecta y esta se compare con la entrada del micrófono y pueda detectar dicho patrón con un nivel de confianza aceptable, para luego colocarse de forma escrita como expresión musical.

1. REDES NEURONALES

Una red neuronal es la composición de funciones no lineales de dos o más neuronas (Dreyfus, 2010).

Es un sistema de neuronas interconectado en forma de red, relacionados de tal manera que interactúan y colaboran para obtener un estímulo de salida adecuado.

Una red neuronal artificial es el análogo de un sistema nervioso de los animales en donde un conjunto de neuronas interactúan para producir resultados positivos al individuo.

El nivel de aprendizaje y procesamiento de una red neuronal está íntimamente ligado a la complejidad de las neuronas que lo componen y también está ligado al número de neuronas que interactúan para obtener el estímulo de salida. Debido a esto las redes neuronales heredan las ventajas y desventajas de las neuronas artificiales como por ejemplo el rango de error de una neurona.

Las redes neuronales artificiales pueden llegar a componerse desde un par de neuronas hasta miles, tal vez millones de neuronas funcionando en un solo sistema para emular o estudiar la forma de razonamiento de un ser vivo.

La tolerancia a fallos es una de las ventajas de una red neuronal artificial, debido a que una neurona puede desempeñar varias funciones y que cada neurona funciona prácticamente independiente de las demás si se extrae una o

varias neuronas de una red neuronal, dicha red neuronal debería de continuar funcionando incluso otras neuronas podrían realizar el trabajo de las neuronas extraídas de la red..

Otra ventaja de las redes neuronales es la alta adaptabilidad de una red neuronal, debido a que cada neurona puede “aprender”, una red neuronal está en toda la capacidad de aprender situaciones con un grado de complejidad considerable dependiendo del número de neuronas que se tienen en la red.

Actualmente las redes neuronales se empiezan a utilizar en muchos campos debido a su flexibilidad; por ejemplo se utilizan en inteligencia de negocios para analizar tendencias y patrones; se puede encontrar su uso en muchos sistemas expertos destinados para biología, botánica y economía; se han hecho muchos estudios de la naturaleza del razonamiento humano utilizando redes neuronales de miles de neuronas. Últimamente se han estado utilizando en redes sociales para analizar el comportamiento de los usuarios, patrones y así realizar búsquedas, publicidad y sugerencias más acertadas y sensibles de contexto.

1.1. Neurona artificial

Una neurona es una función no lineal, parametrizada y bien delimitada (Dreyfus, 2010).

“Es una unidad de cálculo que intenta modelar el comportamiento de una neurona natural, similares a las que constituyen el cerebro humano” (Rojas, 1996).

Una neurona tiene un conjunto de variables de entradas, esta también posee una o más variables de salida.

Por conveniencia muchos autores representan una neurona mediante un grafo dirigido. Esta representación fue inspirada en biología desde los primeros estudios formales de la neurona en de 1943 de McCulloch y Minsky en 1969.

La función que contiene la neurona debe ser ajustada a la tarea que realizará la neurona. Regularmente se utilizan dos tipos de parametrización para una neurona, que a continuación se describen:

Los parámetros son asignados a las entradas de la neurona, la salida de la neurona es una combinación no lineal de entradas cada entrada tiene un peso. Este peso está inspirado en los pesos de las sinapsis de una red neuronal biológica.

Las neuronas artificiales son útiles en áreas en donde la respuesta del sistema no necesita ser exacta, debido a la naturaleza de la lógica que se utiliza en su implementación.

1.1.1. Neurona biológica

La palabra neurona proviene del griego *veüpov*, que significa cuerda o nervio; son células que componen la gran mayoría del sistema nervioso, estas células reciben un estímulo y este lo conducen hacia otras células en forma de potencial de acción Las células biológicas tienen un cuerpo celular llamado soma, y un conjunto de prolongaciones que transmiten los impulsos llamados dendritas; así también tiene una prolongación llamada axón que es la que conduce los impulsos desde el soma hacia otras células.

Las neuronas biológicas pueden comunicarse de manera precisa y rápida a través de impulsos nerviosos, estos impulsos nerviosos se propagan a velocidades cercanas a 3 ms y con un voltaje aproximado de entre 65mV y 75mV.

Los estilos son una herramienta importante que se utilizó para facilitar el uso de la plantilla. A continuación se presenta una figura para poder habilitar los estilos.

1.1.2. Notación matemática

La siguiente notación es una de las más utilizadas.

$$v = w_0 + \sum_{n-i}^{i=1} w_i x_i$$

En donde:

v Es la suma de las entradas, es la salida de la neurona.

w_o Es una constante, comúnmente llamada “bias”.

w_i Conjunto de pesos de cada entrada de la neurona.

x_i Conjunto de entradas de la neurona.

1.1.3. Notación como grafo dirigido

La notación de grado dirigido es una de las más utilizadas en las redes neuronales debido a que es una notación gráfica.

v Es la suma de las entradas, es la salida de la neurona.

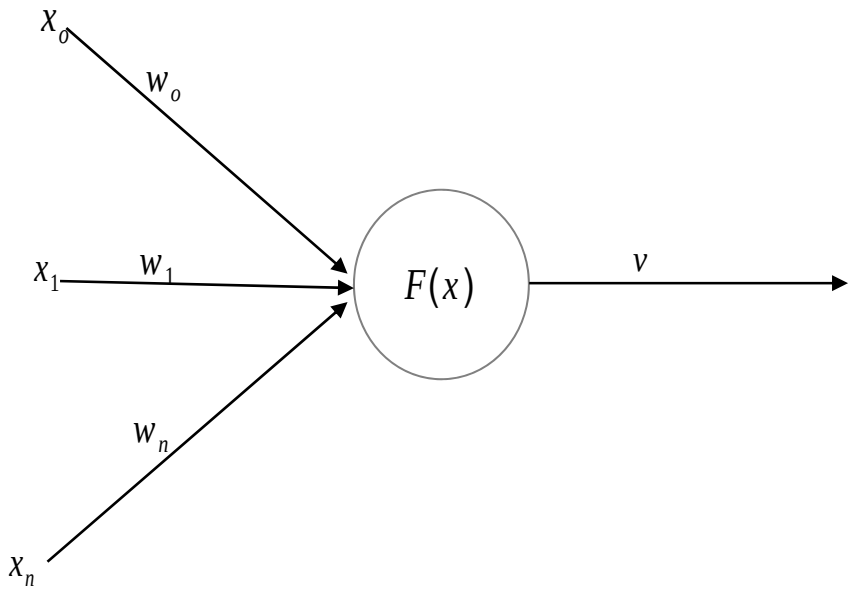
w_o Es una constante, comúnmente llamada “bias”.

w_i Conjunto de pesos de cada entrada de la neurona.

x_i Conjunto de entradas de la neurona.

$F(x)$ Es la función de activación.

Figura 1. **Ingeniería de software – arquitectura adecuada**



Fuente: elaboración propia.

1.1.4. Función de activación

La función que contiene la neurona es llamada función de activación. Esta función de activación se acostumbra a que sea una función con signo; como por ejemplo alguna función trigonométrica, esta función es la analogía del cuerpo de la neurona (soma).

La función de activación debe elegirse tomando en cuenta como mínimo el número de variables de entrada y salida de la neurona, exactitud en las salidas de la neurona y tiempo de respuesta requerido.

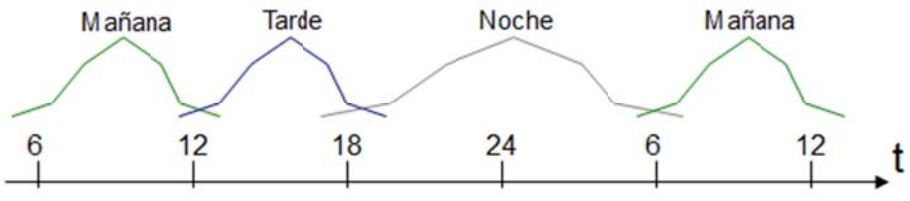
Existen varios tipos de funciones de activación, siendo los más comunes la función de umbral, de segmentos lineales y funciones de activación logísticas.

1.1.4.1. Función de umbral (Heaviside)

Las entradas con sus respectivos pesos tienen un tratamiento de forma lógica, estructurando así una salida lo más lógica posible. Este tipo de función de activación tiene poca operación matemática, en cambio contiene una lógica muy bien definida, regularmente se utilizan funciones estadísticas para determinar la salida de este tipo de función de activación.

En este tipo de función de activación resultan muy útiles las distribuciones estadísticas como la distribución normal, en donde la neurona realiza una prueba de hipótesis con los datos de entrada y la salida de la neurona es el resultado de la prueba de hipótesis.

Figura 2. **Función de umbral**

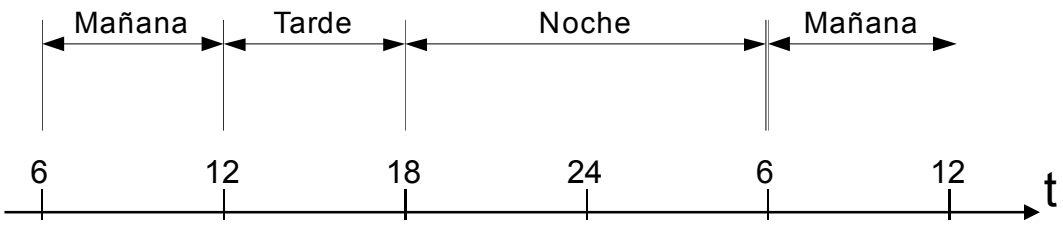


Fuente: elaboración propia.

1.1.4.2. Función de segmentos lineales

En este tipo de función de activación se establece un rango de acción para cada entrada, estableciendo así relaciones lineales entre entradas y salidas. Este tipo de función de activación es una de las más rápidas, debido al poco tratamiento que tienen las entradas en la función, pero debido al funcionamiento difuso de la neurona puede llevar a ciertos resultados no deseados o inapropiados.

Figura 3. **Función de segmentos lineales**



Fuente: elaboración propia.

1.1.5. Función logística

Esta función de activación es la combinación de umbral y de segmentos lineales. Este tipo de función de activación combina una parte de tratamiento lógico de las entradas y una parte de rango de acción para los resultados intermedios. Como consecuencia de la complejidad de este tipo de función de activación es la más lenta y costosa en recursos computacionales.

1.2. Comparación entre red neuronal biológica y red neuronal artificial

Cuando se compara un sistema computacional con el cerebro humano se puede notar que la velocidad de procesamiento de un computador es superior a la de un cerebro humano, ya que el cerebro humano tiene una velocidad para procesar datos de aproximadamente y la velocidad de un computador para procesar datos es de aproximadamente.

Comparando el número de procesadores el cerebro humano tiene una ventaja muy clara con unos procesadores trabajando en paralelo, mientras que un computador de los más potentes tiene unos 250 procesadores trabajando; esto quiere decir que hasta el computador más potente tiene apenas el porcentaje de los procesadores de un cerebro humano.

Todo esto sin siquiera considerar el número de conexiones que tiene cada unidad de procesamiento del cerebro, la alta tolerancia a fallos contra la prácticamente inexistente tolerancia a fallos de un sistema computacional a nivel de datos (Raquel Flórez López, 2008).

1.3. Arquitectura de una red neuronal

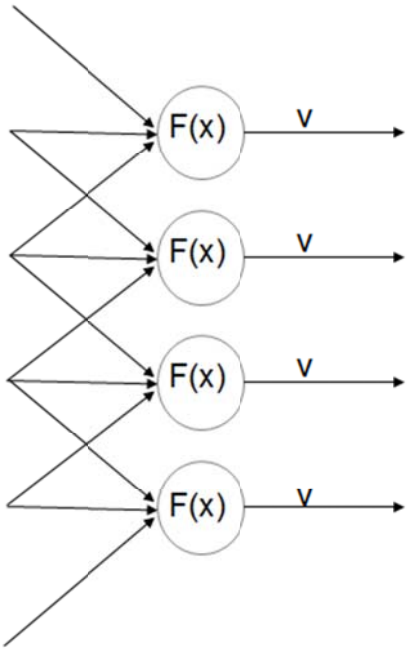
Cuando se refiere a la arquitectura de una red neuronal se refiere a la organización, disposición y estructura de las neuronas artificiales en una red neuronal; esta organización, disposición y estructura tiene un impacto directo sobre el rendimiento y procesado de los datos procesados por la neurona. Debe tenerse en cuenta también la precisión que requieren las salidas de la red neuronal, ya que la velocidad de comunicación entre neuronas, número de conexiones de las neuronas y retroalimentación de cada neurona tiene un impacto de rendimiento directo sobre la red neuronal completa.

1.3.1. *FeedForward* de capa simple

En este tipo de red neuronal las neuronas son organizadas de tal forma que solo se tiene una capa de neuronas sin ningún tipo de ciclo. Este tipo de red neuronal tiende a ser rápida debido a que todas las neuronas están organizadas en una sola capa, por lo que el tiempo de respuesta de la red neuronal esta dado por la neurona con el menor tiempo de respuesta.

Por otro lado se tiene que también es una arquitectura que no tiene retroalimentación, por lo que su capacidad de aprender es limitada y muchas veces requiere de un aprendizaje monitorizado para lograr el resultado esperado.

Figura 4. **Arquitectura de red neuronal *feedforward* de capa simple**



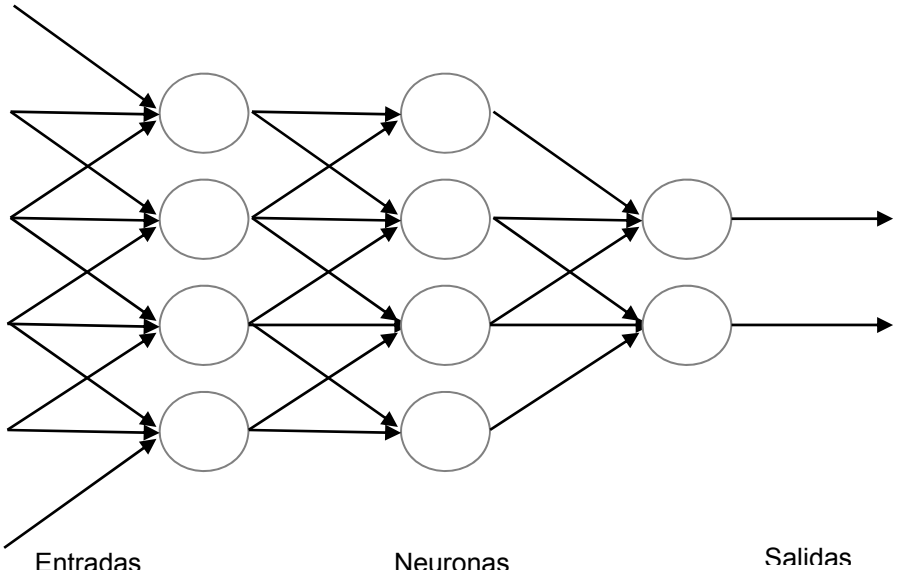
Fuente: elaboración propia.

1.3.2. *FeedForward* multicapa

Este tipo de arquitectura de redes neuronales tiene una o más capas intermedias entre la entrada y la salida de la red neuronal, estas capas suman capacidad de procesamiento a la red neuronal lo que hace que la red neuronal tenga una capacidad de resolver problemas más complejos.

La velocidad de este tipo de red neuronal está dado por la neurona más lenta de cada capa; esto quiere decir que la velocidad de la red neuronal tiene por lo menos dos variables el número de capas de la red neuronal, y mientras más capas tenga la red neuronal más lenta será la velocidad de respuesta de dicha red neuronal, pero mayor será la capacidad de aprendizaje y resolución de problemas de dicha red neuronal.

Figura 5. **Arquitectura de red neuronal *feedforward* multicapa**



Fuente: elaboración propia.

1.3.3. Redes recurrentes

Esta arquitectura de red neuronal tiene por lo menos un ciclo que sirve para retroalimentación. La capacidad de procesamiento de estas redes es depende del número de capas, número de neuronas, cantidad de entradas y salidas, y, número de conexiones de retroalimentación, por lo tanto esta es el tipo de arquitectura de redes neuronales más robusta y compleja.

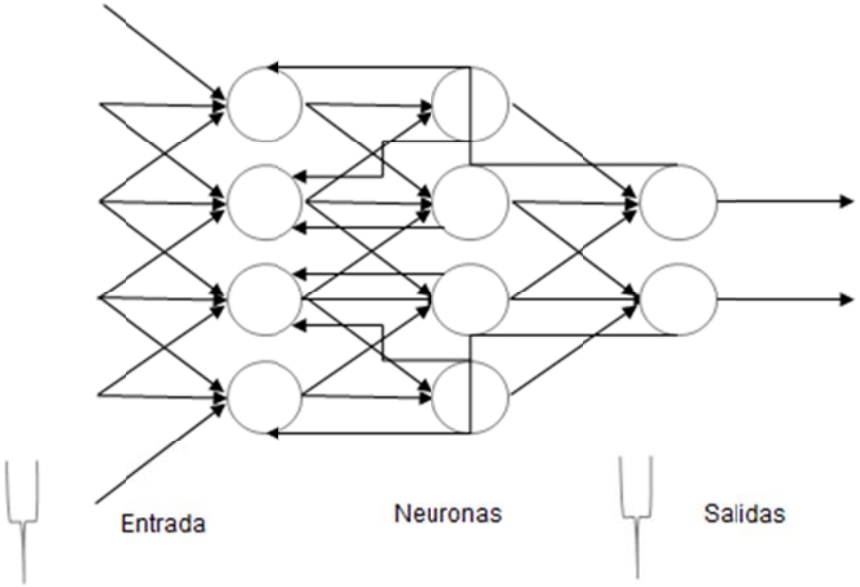
Debido a la complejidad de este tipo de arquitectura de red neuronal el tiempo de respuesta es complejo de calcular, siendo la opción más viable el cálculo estadístico de tiempos medios con sus respectivas desviaciones.

Este tipo de redes neuronales suelen dar respuestas que no son inmediatas, regularmente dan una solución a un problema que van mejorando a medida que la retroalimentación fluye hacia la red neuronal, es decir la respuesta va mejorando con el tiempo.

Este tipo de redes neuronales es una mejor aproximacion a la forma en la que trabajan los cerebros de los animales y por esta razon es que este tipo de redes neuronales se han utilizado para imitar la forma en la que el cerebro de los animales funciona a nivel de procesamiento y almacenamiento de datos.

Debido a que la complejidad de representacion matematica de este tipo de redes neuronales es muy complicada regularmente se opta por una notacion de grafo dirigido; en donde cualquier ciclo en el grafo representa una retroalimentacion a la red; sin embargo, aun con la notacion de grafo dirigido resulta imposible expresar algunas situaciones por ejemplo: cuantas veces se puede retroalimentar una neurona por el mismo impulso.

Figura 6. **Arquitectura de red recurrente**



Fuente: elaboración propia.

1.4. Lógica difusa

"La lógica borrosa es una rama de la inteligencia artificial gracias a la cual los ordenadores pueden diluir el blanco y el negro de lo lógica ordinaria en los grises con el que el sentido común percibe el mundo incierto" (Bosko, 1993).

La lógica difusa, en muchas veces llamada lógica heurística es un tipo de lógica que intenta aproximarse a el razonamiento humano regular; este tipo de lógica fue propuesta por LotfiZadeh en 1965.

La lógica tradicional que tiene un universo de dos elementos conformados por falso regularmente representado también por cero (0) y verdadero

representado también por uno (1); de esta manera se representan todas las posibles situaciones y aseveraciones de forma formal en la lógica tradicional.

En comparación con la lógica difusa, esta puede adoptar cualquier valor entre cero (0) y uno (1) incluyendo al cero y uno; los infinitos valores entre cero y uno son todos los valores que puede adoptar una variable de lógica difusa. Este universo de posibles valores hace posible extender el concepto de verdadero o falso y soporta conceptos como verdad parcial.

La lógica formal tradicional aplica pocas veces para el mundo real, reduciendo su uso a comprobaciones matemáticas, problemas de computo o problemas de decisión, búsqueda u optimización; quedando fuera de este tipo de lógica todas las situaciones o problemas con cierto grado de incertidumbre, para estas situaciones o problemas la lógica difusa funciona bien dando resultados en donde la lógica convencional no tiene sentido, permitiendo aproximar valores, aceptando situaciones y datos ambiguos.

Por lo general se utiliza una función estadística con algún grado de aceptación para dar alguna solución a un problema de esta índole.

En su libro fundamento de inteligencia artificial Luis Alvarezez Mnarriz define un esquema lingüístico rellenado por términos con significado; por medio de ella se representan los elementos de un universo de discurso, cuyos valore son palabras o enunciados del lenguaje natural o artificial, y se puede simbolizar a través de la quíntupla:

{ X, T(X),U,G,M }En donde:

X: es el nombre de la variable que refiere los objetos, cosas o propiedades: edad, altura, temperatura y demás.

T(X): hace referencia al conjunto de términos asociados o valores lingüísticos de la variable, y que son expresados en el lenguaje natural con diversos grados. Por ejemplo:

Términos primarios: joven-viejo
Modificadores: no muy joven
Combinaciones con conectores lógicos: muy joven y bastante pesado.

G: gramática o conjunto de reglas sintácticas que permite generar términos en T(X).

M: regla semántica por medio de la cual se asigna a cada término de la variable un significado según la función de pertenecía que se ha expuesto anteriormente par los subconjuntos difusos.

Ahora pues con la definición de lógica difusa bien establecida se puede dar unos ejemplos para mejorar la comprensión del lector:

Es una persona muy alta.

En este ejemplo se tiene que el objeto de variable que es la “persona” y un conjunto de términos modificadores a dicha variable “muy alta”; esto hace de dicha oración una afirmación difusa, debido a que no se establece ninguna métrica de que rangos de estatura tiene una persona alta, dejando así el contexto y a la interpretación del lector

Si vienen temprano tal vez vamos a la playa.

Este ejemplo es un poco más complejo, debido a que se tiene dos variables ambiguas; el primero es el término primario “temprano” y el segundo que es el conector lógico “tal vez”, debido al nivel de complejidad de esta oración es imposible representarla con lógica tradicional; en cambio puede tener una serie de representaciones en lógica difusa.

1.5. Aprendizaje de una red neuronal

El aprendizaje de una red neuronal es común que se defina dependiendo del contexto y objetivos de la red neuronal.

1.5.1. Aprendizaje supervisado

Esta técnica de aprendizaje de la red neuronal consiste en estimular la red neuronal con patrones y salidas ya conocidas, para cada salida de la red neuronal que se asemeje más a la salida correcta se da un estímulo positivo a la red neuronal.

Este proceso de aprendizaje se realiza de forma iterativa; es decir al terminar de ajustar el último peso de todas las neuronas se debe iniciar de nuevo con la estimulación de los patrones, esto debido a que los pesos de las neuronas han cambiado y necesitan ajustarse. Cuando los valores de las entradas coinciden con el grado de precisión deseada de las salidas se para el ciclo de aprendizaje y se dice que el algoritmo de aprendizaje ha convergido.

Se puede ejemplificar este método de aprendizaje como cuando una persona da un estímulo a un perro, por ejemplo lanzar una varita y cuando el

perro lleva la varita a su dueño este estimula al perro de manera positiva, dándole una galletita. En este ejemplo se pueden notar todas las partes del aprendizaje supervisado: un estímulo de entrada (el lanzamiento de la varita), un proceso del ente inteligente (el perro va por la varita), una salida (el perro entrega la varita a la persona), un estímulo positivo cuando se obtiene el resultado esperado (premiar al perro con la galletita).

Este tipo de aprendizaje de la red neuronal permite ajustar los pesos de cada entrada a la neurona, esto hace posible llegar a funciones de activación mucho más refinadas y ajustadas, que pueden dar como resultado redes neuronales mucho más exactas y rápidas.

Como contra-parte se tiene que para redes con un número alto de neuronas se vuelve casi imposible ajustar manualmente el peso de cada entrada de cada neuronal, convirtiéndose en un trabajo extremadamente tedioso y susceptible de errores.

1.5.2. Aprendizaje no supervisado

En este tipo de técnica de aprendizaje la red neuronal no es estimulada cuando se produce una respuesta; dicho de otra forma no se le indica a la red neuronal cual debe de ser la respuesta correcta.

Debido a que la red neuronal no tiene ningún indicio de cual es la respuesta correcta la red neuronal debe asociar o aprender por si misma; esto exige una cantidad de datos de entrada bastante alta, por lo menos lo suficiente para que la red neuronal pueda realizar las asociaciones necesarias para converger.

2. INTRODUCCIÓN A NOTACIÓN MUSICAL

Notación musical es el nombre designado para cualquier sistema de escritura que represente una composición o pieza musical. Esta notación permite al lector de la notación interpretar la composición o pieza musical como el compositor la ha diseñado.

Existen múltiples sistemas de notación musical que van desde la antigua Mesopotamia hasta algunos diseñados en el último siglo utilizados en música popular; cada uno de ellos con sus propias características que permiten la interpretación de la composición o pieza musical sea más exacta en su interpretación o menos exacta en su interpretación

Actualmente existen notación musical que puede indicar al intérprete las alturas y duraciones de cada nota, proveyendo una exactitud bastante alta a la hora de interpretar una composición o pieza musical; debido a esto existen múltiples herramientas de software que permiten que un computador pueda ejecutar una pieza musical. A la herramienta de software se le provee de una entrada que sería la notación musical y el programa tiene una salida que serian los sonidos producidos al interpretar la pieza musical de forma automatizada.

Las interpretaciones producidas por una herramienta de software tienen una exactitud de milésimas de segundo y pueden interpretar las más complejas melodías, sin embargo estas interpretaciones son mecánicas y a menudo se escuchan “robotizadas” por la interpretación tan estricta realizada por una computadora.

Por otro lado, actualmente no existe ninguna herramienta popular que tenga la capacidad de traducir un sonido a su notación musical debido a la complejidad de este problema. Por un lado se tiene la peculiar forma de interpretación de cada ser humano, y por otro lado se tiene las mínimas, pero susceptibles variaciones de sonido al utilizar instrumentos de la misma índole.

Incluso utilizando el mismo instrumento pueden existir variaciones en los sonidos debido a cambios de componentes del instrumento musical, por ejemplo al cambiar una cuerda de una guitarra.

Sumado a todo esto se tiene que la entrada de los sonidos para ser procesada esta en un ambiente en donde pueden haber ruidos y situaciones que la herramienta no tiene la capacidad de diferenciar, por lo que realizar una aplicación de software de este tipo es una tarea compleja y no puede ser realizada utilizando soluciones tradicionales debido a la poca exactitud que puede llegar a tenerse con la entrada a la aplicación de software.

2.1. El Pentagrama

También conocida como notación patrón. Esta escrita sobre cinco líneas y cuatro espacios a las que se les llama pentagrama. Sobre las líneas y espacios se anotan las notas musicales, acordes, tiempos, duraciones, expresión son solo algunos elementos que se representan en una partitura sobre el pentagrama.

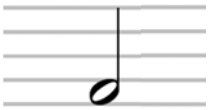
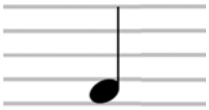
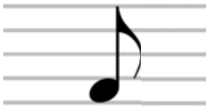
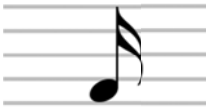
2.1.1. Tiempo y compas

Indican las unidades de tiempo que debe hacer en cada compas; estos determinan la estructura rítmica de la pieza musical.

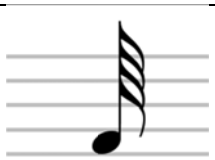
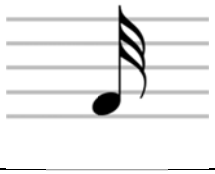
2.1.2. Figuras musicales o rítmicas

Son símbolos que representan el tiempo de duración de las notas musicales. Cada símbolo representa una duración de tiempo para ser ejecutado. La siguiente tabla contiene los símbolos y sus duraciones.

Tabla I. **Figuras musicales**

Nombre	Figura musical	Duración
Redonda		1
Blanca		½
Negra		¼
Corchea		1/8
Semicorchea		1/16

Continuación de la tabla I.

Fusa		1/32
Semifusa		1/64
Figura negra con puntillo		3/8

Fuente: elaboración propia

2.1.3. Silencios o Pausas

Es el tiempo en el que la voz o instrumento no produce sonido. Las pausas tienen duraciones muy similares a las de las figuras musicales. Debido al alcance propuesto no se estudiara con detalle estos símbolos musicales, esto se debe a que una partitura si expresa los tiempos de una pieza música, sin embargo una tablatura.

2.1.4. Puntillo

Se agrega a una nota musical o silencio del lado derecho y este añade la mitad del la duración de la nota que la precede.

2.1.5. Doble puntillo

Son dos puntillos que se agregan a una nota musical y agregan duración a la nota o silencio que le precede; el primer puntillo agrega la mitad de la duración de la nota mientras que el segundo agrega un cuarto de la duración de la nota que le precede.

2.1.6. Claves

Estas regularmente son llamadas claves y son el símbolo con el que se inicia la partitura, estas indican la afinación del instrumento en el que se ejecutara la pieza musical.

2.1.7. Alturas

Las alturas de las notas se representan por la posición en la pauta en la que referencia a la nota definida por la clave utilizada, esta se refiere a la frecuencia de la nota musical (medida en hertzios) y no al volumen en el que se ejecuta la nota musical.

2.1.8. Alteraciones

Son alteraciones a la nota que será modificada respecto a tono. Entre las alteraciones de tono se mencionan:

- El sostenido: indica que la nota se ejecuta con medio tono arriba
- El bemol: indica que la nota se ejecuta con medio tono abajo
- El doble sostenido: indica que la nota se ejecuta un tono arriba
- El doble bemol: indica que la nota se ejecuta un tono abajo

2.1.9. Expresión

Existen símbolos que indican al interprete la forma de ejecutar la partitura, incluyendo las variaciones de volumen y tiempo; así como la manera correcta de articular las notas y separar en frases (Wikipedia, 2013).

2.2. Tablatura

Cuando se habla de la guitarra, tablatura es una notación que representa la colocación de los dedos en los trastes de la guitarra en vez de la notas; esto simplifica la interpretación de la pieza musical pero carece ritmo, tiempo, dinámica y cinética en su escritura.

Una tablaturaestá compuesta por el número de cuerdas que tiene el instrumento, en el caso de la guitarra está compuesta por 6 cuerdas.

Cada línea de la tablatura puede indicar la afinación del instrumento, en donde se utiliza nomenclatura americana; aunque regularmente puede omitirse.

```
e||---------0---------|-----------0----|
B||-----1-------1---1-|----1---1-----3-|
G||--0----------------|-0--------------|
D||-------------------|----------------|
A||-------------------|----------------|
E||-------------------|----------------|
 .....Arroz con le-che   me quiero casar
```

Fuente: http://acordes.lacuerda.net/canciones/arroz_con_leche.shtml. Consulta: 01 de septiembre de 2013.

3. ANÁLISIS Y DISEÑO DE UNA HERRAMIENTA QUE UTILICE REDES NEURONALES PARA DETECTAR PATRONES MUSICALES Y LOS TRADUZCA A NOTACIÓN DE TABLATURA UTILIZANDO LA GUITARRA COMO INSTRUMENTO DE TRADUCCIÓN

En este apartado se analizaran algunas de las metodologías para poder elegir una que se adecue al desarrollo de la herramienta para detectar patrones musicales, así también se realizara el diseño de dicha herramienta utilizando la metodología escogida.

3.1. Introducción

La construcción de una herramienta que utilice redes neuronales para la detección de patrones musicales y los traduzca a notación de tablatura utilizando el marco teórico desarrollado en los capítulos anteriores como objetivo primario.

En este capítulo se lleva a cabo la especificación de requerimientos de dicho software, utilizando el estándar IEEE 830 para la especificación de requerimientos.

3.2. Propósito

Desarrollar una aplicación que utilice redes neuronales para la detección de patrones musicales y los traduzca a notación de tablatura.

Esta aplicación puede ser de interés para audiencia académica interesada en el estudio de redes neuronales así como su aplicación en la vida cotidiana y otras ciencias y disciplinas.

También puede llegar a ser de interés a los entusiastas de la música, comprobando así como el avance de la tecnología empieza a abarcar casi todos los aspectos de la ciencia y arte.

3.3. Alcance

El nombre del producto de software es Neuro-Traductor, este es un producto de software que recibe una entrada de audio de la computadora y realiza una detección de sus patrones musicales, los cuales coloca en forma de tablatura.

Entre las limitaciones se tiene que el audio debe ser proveído por una guitarra para su correcta traducción. La aplicación será capaz de detectar los patrones musicales en los primeros tres trastes de la guitarra, esto debido a la complejidad de la detección de los patrones musicales en toda la guitarra.

Respecto a la entrada de audio, proveído por una guitarra; es estrictamente necesario que está este afinada con una calidad aceptable para la correcta detección de patrones musicales.

3.4. Definiciones, acrónimos y abreviaturas

- Nota Musical: el elemento más básico y primordial de la música.

- Tablatura: forma de escritura especial para ciertos instrumentos que presentan únicamente posiciones y colocaciones.

- Pentagrama: conjunto de cinco líneas rectas paralelas equidistantes, sobre el que se escriben las notas y signos musicales.

- Frecuencia: número de oscilaciones, vibraciones u ondas de tiempo en cualquier fenómeno periódico.

- Tono: propiedad de los sonidos que permite ordenarlos de graves a agudos según su frecuencia.

3.5. Perspectiva del producto

Neuro-Traductor es un producto de software autónomo, es decir que no forma parte de ningún otro producto de software. Neuro-Traductor es un producto portable, inter-operable y depende únicamente de java 1.6 o superior para su ejecución.

3.5.1. Funciones del producto

Las funciones principales del Neuro-Traductor se listan a continuación (solo requerimientos funcionales).

- Mostrar la detección de un patrón musical graficado en la interfaz de usuario.

- Mostrar la detección de un patrón musical como tablatura

- Permitir guardar la tablatura generada en la detección

3.6. Características del usuario

Con el objetivo de lograr una detección aceptable de patrones musicales se necesita que los usuarios tengan un conocimiento básico del uso de la computadora en general.

Además se requiere que el usuario tenga un conocimiento básico del uso del instrumento musical, que para este caso es la guitarra.

3.7. Restricciones

El producto de software generado está bajo licencia GLP y puede ser estudiado, copiado y reproducido haciendo mención de la fuente y el autor.

- Supuestos y dependencias

    - La herramienta depende directamente de la máquina virtual de java versión 1.6 o superior. Así como también una entrada de audio, la aplicación por defecto toma la entrada del micrófono frontal como fuente de datos.

    - La herramienta hace uso intensivo de hilos por lo que se recomienda una computadora con un mínimo de dos procesadores para mantener un rendimiento aceptable.

3.8. Requerimientos específico

Interfaces externas.

Para poder guardar una tablatura se requiere un espacio mínimo en alguna unidad de almacenamiento de la computadora.

La interfaz de usuario debe actualizarse con un retardo máximo de medio segundo.

3.8.1. Detectar patrón musical

Actores: Usuario
Tipo: Primario
Descripción: El usuario puede tocar una cuerda de la guitarra, la cual será detectada por el producto de software Neuro-Traductor y se mostrara en la interfaz gráfica.

3.8.2. Escribir patrón musical

Actores: Usuario
Tipo: Primario
Descripción: El usuario puede tocar una cuerda de la guitarra, la cual se traducirá a notación de tablatura y se mostrara en la interfaz gráfica con todos los patrones musicales detectados con anterioridad.

3.8.3. Limpiar tablatura

Actores: Usuario
Tipo: Primario
Descripción: El usuario puede limpiar todos los patrones musicales detectados hasta el momento al escoger esta opción; esto también se ve reflejado en la interfaz gráfica.

3.8.4. Guardar tablatura

Actores: Usuario
Tipo: Primario
Descripción: El usuario puede guardar la tablatura que se muestra en la interfaz gráfica lo que abre un cuadro de diálogo para guardar dicha tablatura en un fichero de texto plano.

3.9. Requerimientos no funcionales

Son características del software que describen cualidades no funcionales del mismo, estas cualidades indican cómo debe comportarse el software, por ejemplo rendimiento, seguridad, interoperabilidad, compatibilidad, usabilidad, concurrencia.

3.9.1. (Características operativas) corrección

El software debe cumplir con los objetivos principales y específicos del proyecto con un precisión del 90 por ciento de cada objetivo.

3.9.2. Fiabilidad

La detección de patrones musicales debe tener una precisión aproximada del 75 por ciento en un ambiente libre de ruidos; por lo que se espera una precisión aproximada del 60 por ciento en un ambiente con ruido como la entrada de micrófono estándar.

3.9.3. Eficiencia

Con el fin de hacer uso correcto del hardware el producto de software se debe ejecutar en un computador con doble núcleo y utilizar un máximo del 70 por ciento del CPU, además debe hacer utilizar un máximo de 60 megas de memoria RAM en su ejecución.

3.9.4. Facilidad de Uso

Para medir la facilidad de uso se realizará una encuesta a 10 usuarios que probaran la aplicación Neuro-Traductor y esta deberá tener un puntaje de por lo menos el 80 por ciento de facilidad de uso.

3.9.5. Capacidad de soportar los cambios. Facilidad de mantenimiento

Se deberá cumplir con un estándar de codificación sugerido para que el código sea entendible fácilmente.

El código fuente debe estar debidamente documentado y comentado para su comprensión de alto nivel.

Por facilidad de depuración y compresión se utilizará un IDE de código abierto para realizar el desarrollo

3.9.6. Flexibilidad

La aplicación generada debe ser capaz de distinguir las diferentes notas de los primeros tres trastes no importando el material de la cuerda de la guitarra.

La aplicación generada debe ser capaz de distinguir las diferentes notas de los primeros tres trastes no importando si la guitarra es acústica o eléctrica, solo necesita como entrada el sonido de sus cuerdas.

3.9.7. Facilidad de prueba

Cualquier individuo que cuenta con una guitarra afinada de forma convencional, un computador con la máquina virtual de java y un micrófono debe estar en la completa capacidad de utilizar el producto de software.

3.9.8. (Adaptabilidad a nuevos entornos) portabilidad

El producto de software Neuro-Traductor deberá poder ejecutarse en entornos GNU Linux, Windows xp y Windows 7 conservando todas sus características.

3.10. Decisiones de diseño

En esta parte se describen diferentes decisiones respecto del diseño de la aplicación de software Neuro-Traductor, estas decisiones esta enfocadas al cumplimiento de requerimientos tanto funcionales como no funcionales.

3.10.1. Decisiones respecto a los requerimientos no funcionales

Para poder garantizar el cumplimiento de los requerimientos no funcionales la toma de decisiones de arquitectura, plataforma, librerías, lenguajes y todo lo referente al desarrollo y producto de software han sido enfocados hacia el cumplimiento de dichos requerimientos no funcionales.

Portabilidad: la aplicación se desarrollara sobre el lenguaje java, para poder migrar de forma trasparente a otros sistemas operativos sin necesidad de cambios en el código.

Usabilidad: que el programa sea intuitivo y altamente gráfico, por lo que se necesite un manual mínimo para el uso del producto de software.

Funcionalidad: el producto de software debe cumplir con la lista de requerimientos planteados y sus métricas establecidas.

Rendimiento: se debe tener como mínimo una latencia de detección de sonidos de por lo menos 3 veces por segundo, así también se debe contar un una precisión de por lo menos el 70 por ciento en la detección de patrones musicales; por esto la aplicación

de software debe hacer uso intensivo de hilos para su procesamiento en paralelo.

3.11. Notación musical de tablatura utilizada

Se seleccionó la notación musical de tablatura debido a la complejidad de otras notaciones como la notación de pentagrama en donde se expresan tiempos, duración, acordes y otros aspectos musicales que por la naturaleza borrosa de la redes neuronales es mucho más conveniente utilizar la notación de tablatura.

La notación de tablatura presenta un conjunto limitado de símbolos y colocaciones para su interpretación, además de esto es completamente dependiente del instrumento musical que se esté ejecutando. Además de todo esto, la relativa sencillez de las tablaturas las hace populares y fáciles de leer.

En una tablatura un sonido musical está compuesto por el número de cuerda a tocar y el número de traste en el que se pulsa tal cuerda, esto debe reflejarse en la interfaz gráfica en tiempo real.

3.12. Entrada de audio

La entrada de audio será proveída por Java Sound API, en donde se escoge por defecto la primera línea de entrada de audio del computador que ejecuta la aplicación.

El formato del audio está proveído por la clase AudioFormat de Java Sound API que contiene y abstrae toda la información que soporta la tarjeta de sonido del computador que ejecuta la aplicación. Esta clase contiene

información acerca del número de muestras por segundo, el tamaño de muestra en bits, y enumero de canales, el signo o su carencia de los datos y por último el formato de almacenamiento de los datos en memoria.

3.13. Arquitectura del Software

La arquitectura es un nivel de diseño de más alto nivel de la estructura de un sistema, más allá de los algoritmos y estructuras de los datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema.

Existen varios métodos para documentar la arquitectura, uno de los más aceptados es el modelo 4+1 vistas de Kruchten que comprende las siguientes vistas:

3.13.1. Vista Lógica (la descomposición orientada a objetos)

Esta vista esta principalmente orientada a la funcionalidad del sistema. Que es lo que el sistema debería de hacer en términos de servicios a los usuarios. El sistema es descompuesto en un conjunto de abstracciones tomadas del dominio del problema en forma de objetos y clases.

Esta vista puede incluir diagramas de clases, de comunicación o de secuencia en lenguaje UML.

3.13.2. Vista de despliegue (descomposición en subsistemas)

Esta vista se enfoca en la organización de módulos de software actual en el entorno de desarrollo de software. El software es empaquetado en pequeñas

porciones que pueden ser desarrollados uno o un número pequeño de desarrolladores. Los subsistemas están organizados en capas, cada capa provee de interfaces bien definidas con otras capas.

En esta vista se pueden incluir diagramas de componentes y de paquetes.

3.13.3. Vista de procesos

Esta se encarga de algunos requerimientos no funcionales como desempeño y disponibilidad. Sus principales enfoques son la concurrencia y distribución, integración de sistema, tolerancia a fallos y como la abstracciones principales de la vista lógica encajan en la vista de procesos (o en que procesos o hilo de control deben ser ejecutados).

En esta vista se pueden incluir diagramas de actividad con anotaciones para mejorar su legibilidad. Debido a las características de las redes neuronales esta vista es muy importante ya que expresa la conexión entre neuronas y la forma asíncrona en la que funciona la red neuronal.

3.13.4. Vista física mapear el software en el hardware

La vista física toma compañía principalmente de los requerimientos no funcionales del sistema como disponibilidad, tolerancia a fallos, desempeño y escalabilidad.

El software se ejecuta en una red de computadoras o nodos de procesamiento. Los elementos identificados necesitan ser mapeados en varios nodos.

En esta vista se pueden incluir diagramas de despliegue.

3.13.5. Escenarios (poniendo todo junto)

Los elementos de las cuatro vistas se muestran trabajando juntos, utilizando un pequeño grupo de importantes escenarios, para los que se describen las correspondientes secuencias de interacciones entre objetos y procesos. Los escenarios son en algún nivel de abstracción los más importantes requerimientos.

Esta vista es redundante a las otras, pero es muy importante por dos principales propósitos:

Como conductor para descubrir que los elementos arquitectónicos están bien diseñados y dispuestos.

Como validador e ilustrador que el diseño de la arquitectura esta completado. En esta vista se pueden incluir diagramas de caso de uso, además se pueden adjuntar historias de usuario como complemento a los diagramas.

3.14. Objetivos de la arquitectura y limitaciones

La arquitectura de la aplicación debe coincidir con los requerimientos funcionales y no funcionales, de la aplicación. Debido a que se utilizaran redes neuronales para el procesamiento de información, se hará uso intensivo de hilos en la aplicación; esto también contribuirá a que la aplicación no sea una aplicación sincronizada, dando como resultado una arquitectura más resistente a fallos.

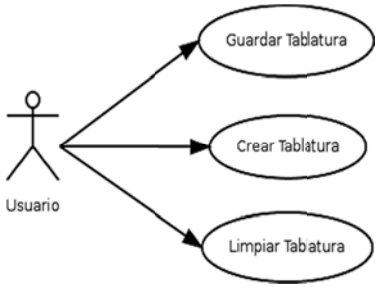
Por otro lado tenemos como objetivo desarrollar una aplicación cercana a tiempo real, para lograr esto las neuronas tendrán que tener un procesamiento casi completamente en paralelo; el número de buffers intermedio debe de ser el mínimo, y por diferencia de velocidad el volumen de datos que se almacena en el disco duro debe ser prácticamente nulo.

Respecto a la portabilidad y interoperabilidad se debe escoger un lenguaje de programación que sea posible de compilar y ejecutar en los sistemas operativos objetivo, entre estos se tiene un conjunto limitado como Python o Java; y por popularidad se escoge el lenguaje de java debido a que los programas codificados en este lenguaje pueden ejecutarse en multitud de sistemas operativos y dispositivos móviles.

3.14.1. Casos de uso

Es un diagrama que muestra la estructura de un sistema de forma estática, este muestra sus clases, aquí se muestran solo los requerimientos funcionales del sistema, todos los requerimientos nos funcionales no se incluyen en este diagrama.

Figura 7. **Casos de uso de alto nivel**



Fuente: elaboración propia.

- Guardar tablatura

    - El usuario puede en cualquier momento guardar la tablatura generada en tiempo real dando eligiendo la opción guardar tablatura, esto deberá generar un archivo de texto plano en el disco duro que contenga la representación de la tablatura.

- Crear tablatura

    - El usuario puede crear una tablatura reproduciendo sonidos musicales con el instrumento musical por la entrada del micrófono, estos sonidos musicales deberán ser traducidos a notación de tablatura en una latencia baja para aproximar a tiempo real.

- Limpiar tablatura

    - El usuario puede limpiar la tablatura que se genera en tiempo real de la interfaz gráfica al escoger esta opción.

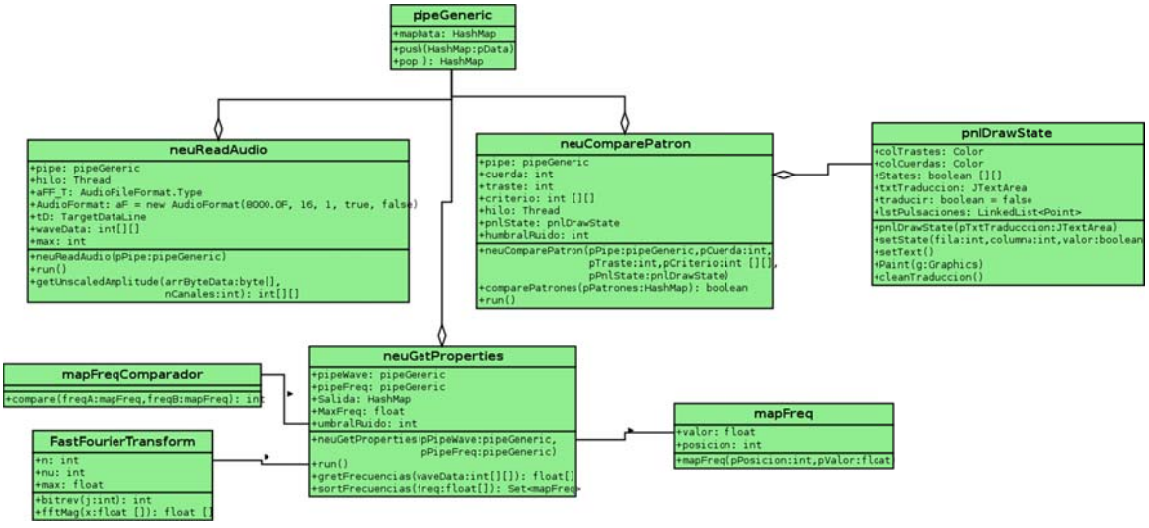
3.14.2. Vista lógica clases, (comunicación, secuencia)

La vista lógica expresa la arquitectura en forma de es tructura(los componentes del sistema que incluye el diagrama de clases) y su comportamiento (que expresa la interacción y dinamismo de sus componentes, aquí se incluyen los diagramas de comunicación y de secuencia).

3.14.2.1. Diagrama de clases

Este diagrama describe la estructura del sistema mostrando sus clases con sus respectivos métodos y propiedades. Es uno de los diagramas básicos para describir un sistema informático orientado a objetos

Figura 8. **Diagrama de clases**



Fuente: elaboración propia.

3.14.2.2. Especificación de clases

A continuación se hace una descripción rápida de cada clase del sistema Neuro-Transmisor.

3.14.2.2.1. neuReadAudio

Clase que implementa una neurona que lee audio de la tarjeta de sonido utilizando las librerías de java y las envía por una tubería hacia la neurona que

extrae las propiedades. Tiene un hilo propio para trabajar en paralelo con otros objetos.

3.14.2.2.2. neuGetProperties

Clase que implementa una neurona que extrae las propiedades de audio que se reciben por una tubería, esto consiste en calcular la transformada de Fourier, ordenar los datos de la frecuencia más intensa a la más débil y mandarla por una tubería las neuronas que comparan patrones.

3.14.2.2.3. neuComparePatron

Clase que implementa una neurona que realiza la comparación de patrones que envía la neurona que extrae las propiedades del sonido y realiza la comparación entre los patrones de sonido establecido para dicha neurona y los patrones de sonido que se extraen de la tarjeta de sonido, si el resultado es positivo se envía un mensaje a un objeto pnlDrawState para actualizar la interfaz gráfica y agregar la traducción a la tablatura.

3.14.2.2.4. pipeGeneric

Clase utilizada como tubería entre objetos, se utiliza para el paso de mensajes entre instancias de clases que utilizan hilos debido a que no es posible la comunicación inter-procesos de forma directa.

3.14.2.2.5. pnlDrawState

Panel en el que se dibuja el mástil de una guitarra indicando las cuerdas y posiciones de trastes que se pisan en tiempo real de la aplicación

3.14.2.2.6. mapFreq

Clase que se utiliza para organizar las frecuencias, contiene dos atributos, la frecuencia y la posición en la que se ubica.

3.14.2.2.7. mapFreqComparator

Clase que implementa un comparador entre clases mapFreq, esto para poder ordenar las frecuencias de acuerdo a su intensidad.

3.14.2.2.8. FastFourierTransform

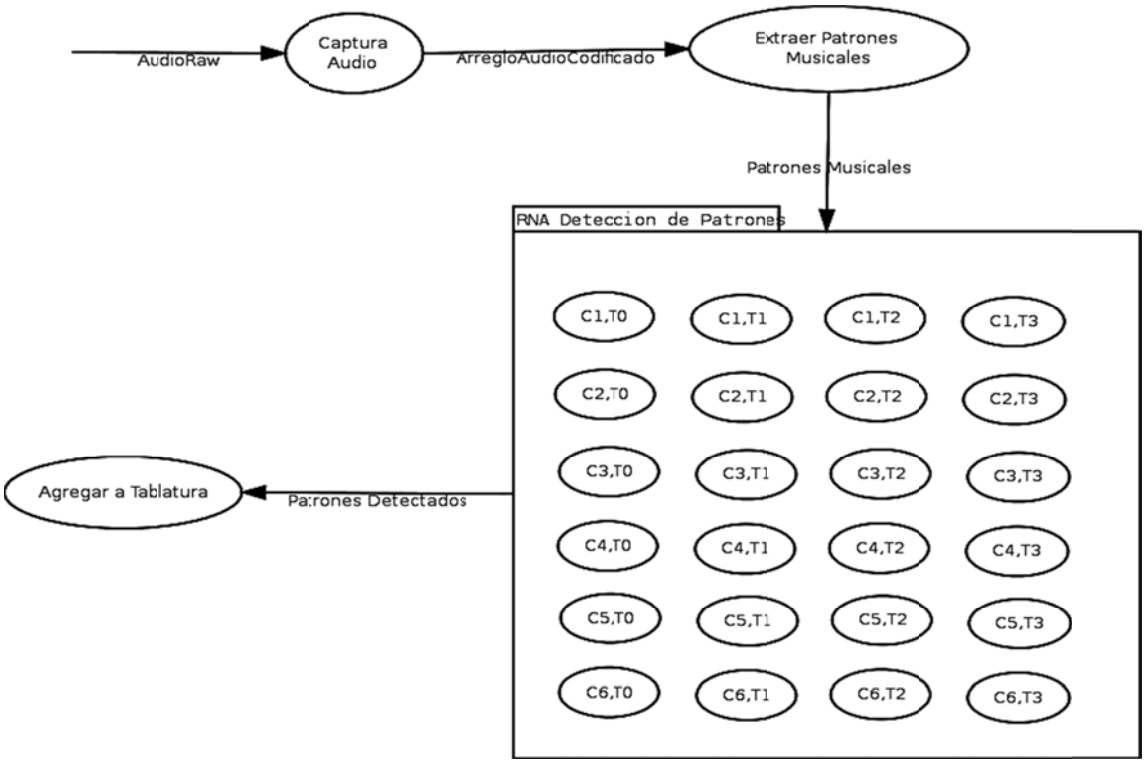
Clase que implementa la transformada de Fourier rápida, esta fue extraída la siguiente dirección bajo licencia GPL:

http://audiofingerprinting.googlecode.com/svnhistory/r101/trunk/SoundCatcher/SoundCatcher/FourierTransform.cs fecha de visita 01/06/2013.

3.14.3. Grafo dirigido RNA

El grafo indica la disposición de la red de neuronas artificiales con sus entradas y salidas indicadas. El paquete RNA detección de patrones contiene un Arreglo de neuronas que detectan los patrones musicales que tendrán un aprendizaje supervisado.

Figura 9. **Arquitectura de red neuronal como grafo dirigido**



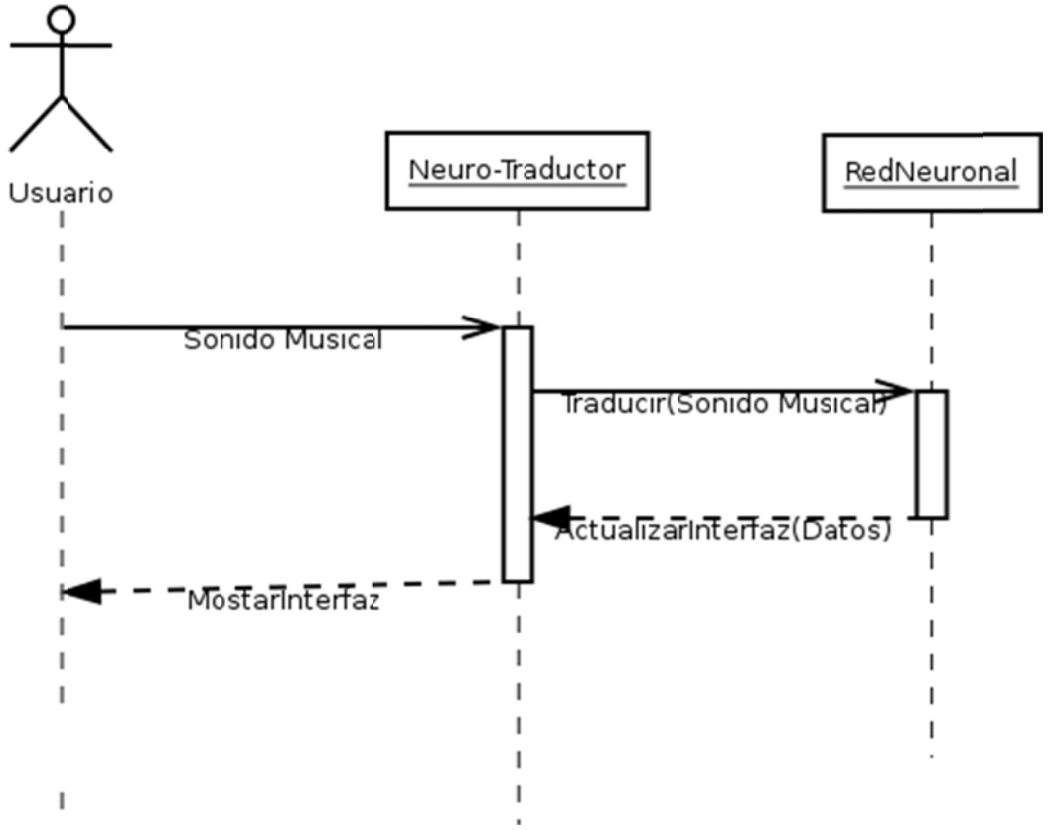
Fuente: elaboración propia.

3.14.3.1. Diagramas de Secuencia

El diagrama muestra a los objetos interactuar en un orden de tiempo con su respectivo intercambio de mensajes, es útil para expresar los objetos que se utilizan para implementar un caso de uso.

Crear Tablatura: expresa el orden de interacción entre objetos del sistema Neuro-Traductor y el usuario, así como también el paso de mensajes y las respuestas del sistema que en este caso sería mostrar la tablatura en la interfaz gráfica.

Figura 10. **Diagrama secuencia, crear tablatura**



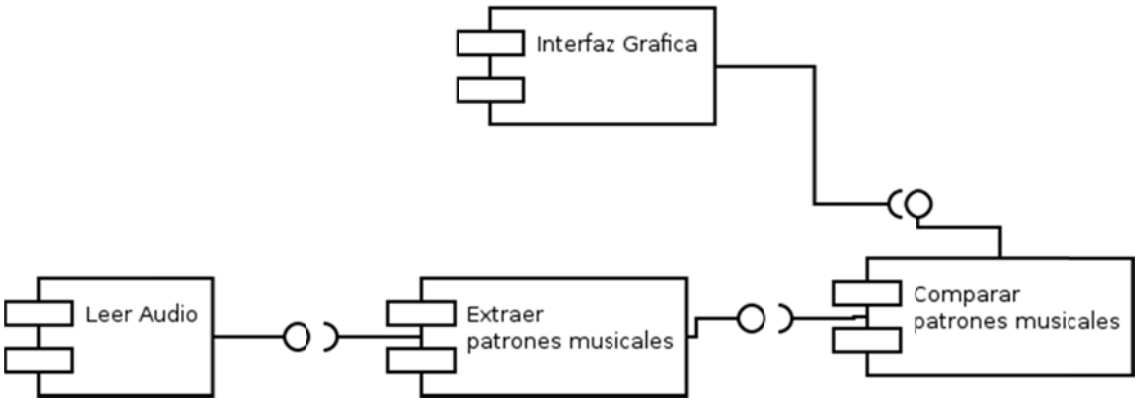
Fuente: elaboración propia.

3.14.4. Despliegue (componentes y paquetes)

Este diagrama se utiliza para expresar los componentes físicos (hardware) y artefactos (componentes de software) así como la comunicación entre los mismos.

Este diagrama será especialmente útil para localizar los puntos de comunicación inter-proceso del sistema, aunque no todas las interfaces tendrán comunicación inter-proceso es un excelente indicador.

Figura 11. **Diagrama de despliegue**



Fuente: elaboración propia.

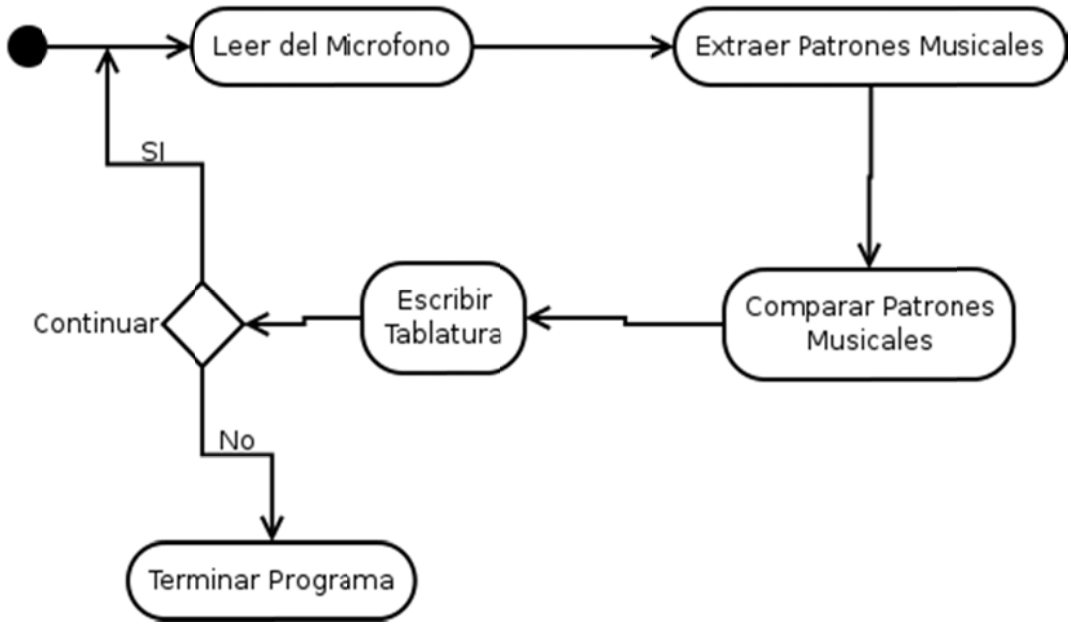
3.14.5. Procesos (Actividad)

Este diagrama expresa las operaciones de los componentes del sistema, básicamente es un diagrama de flujo.

Estos diagramas son de gran ayuda para expresar algoritmos o soluciones de un sistema ya que pueden expresar la lógica y flujo de un sistema de forma gráfica.

Aquí se expresa el flujo y transformación de datos, desde su captura por el micrófono hasta mostrar el resultado en la interfaz gráfica, sin embargo resulta incómodo e impráctico expresar el recorrido de los datos por cada neurona, por lo que se elige un nivel de abstracción lo suficientemente alto para encapsular dicho proceso.

Figura 12. **Diagrama de procesos**



Fuente: elaboración propia.

Con el diagrama de procesos se termina la especificación del sistema Neuro-Traductor y se tiene una especificación suficiente para poder realizar la implementación del sistema informático.

Todos los diagramas expresan el sistema desde diferentes puntos de vista para poder tener una visión integral del sistema, y que la implementación del sistema se pueda llevar a cabo con los recursos y esfuerzo óptimo que requiere la implementación de un sistema de este tipo.

La especificación de clases es sumamente útil para que el lector pueda entender a grandes rasgos que hace cada clase del sistema sin tener que recurrir a leer el código fuente de las mismas.

4. DESARROLLO DE LA HERRAMIENTA DE SOFTWARE

En este capítulo se desarrollará la herramienta que fue propuesta y documentada en el capítulo anterior. Para el desarrollo de la aplicación se utilizará una metodología iterativa incremental.

Este capítulo es importante debido a que en él se mostrará el enfoque del desarrollo utilizado, así como las decisiones de implementación, uso de bibliotecas y todo el proceso de desarrollo de software.

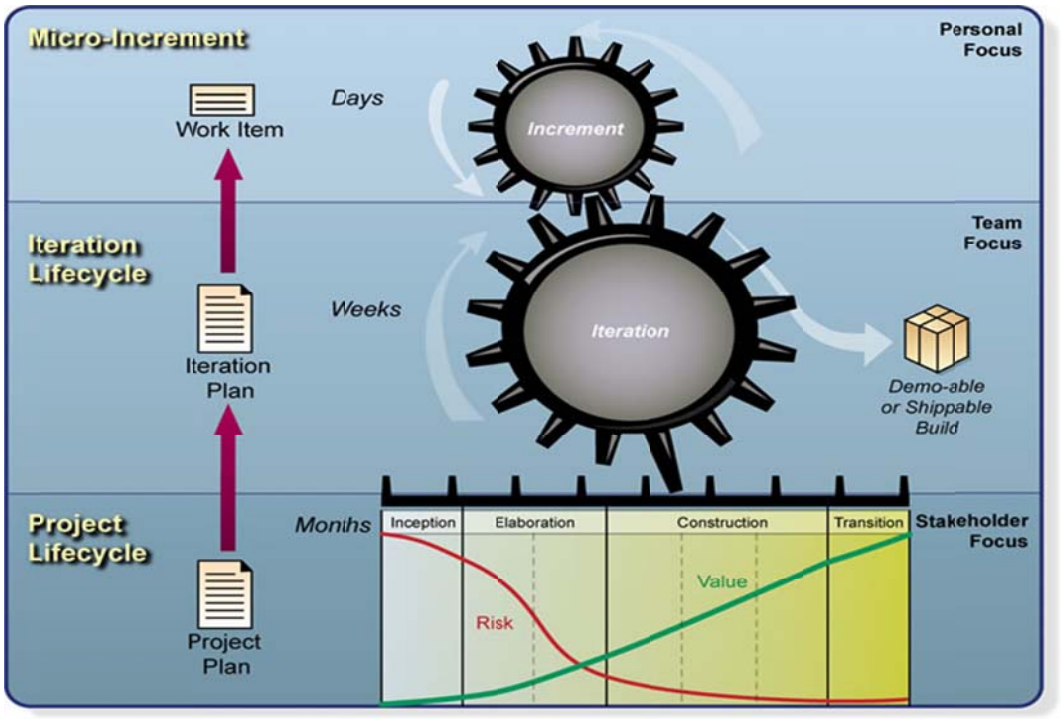
4.1. Enfoque y metodología del desarrollo

Debido a las restricciones de tiempo, el número de funcionalidades se utilizó el método y proceso de desarrollo de software conocido como OpenUp el cual es un método mínimo y suficiente para el desarrollo de proyectos de software.

La metodología OpenUp es una metodología derivada de RUP pero adaptada para el desarrollo ágil de aplicaciones, por lo que fue recortada y solo conserva las partes necesarias para llevar un desarrollo ordenado con una documentación mínima.

OpenUp es una metodología ampliamente utilizada en el software libre y es utilizada por proyectos como Eclipse que también contiene complementos para poder administrar el software desde el mismo entorno de programación integrado Eclipse.

Figura 13. **Diagrama de la metodología de software OpenUp**



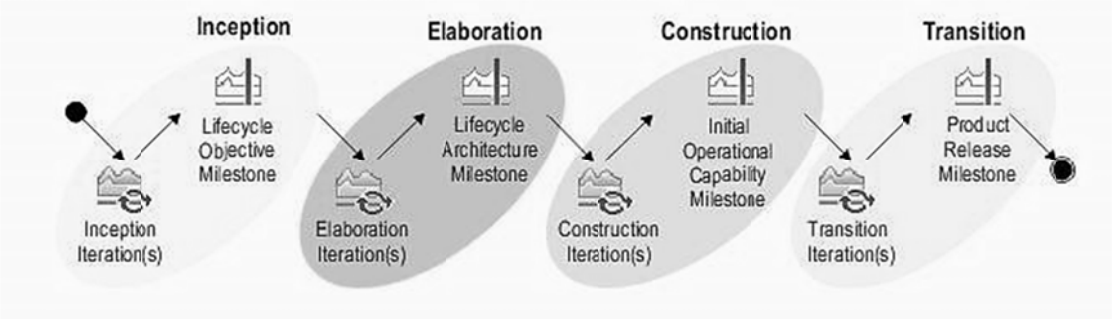
Fuente: http://epf.eclipse.org/wikis/openup. Consulta: 01 de junio de 2014.

En OpenUp el proyecto es organizado en microincrementos, estos representan pequeñas unidades de trabajo que producen un resultado medible para el progreso del proyecto. El proceso está enfocado en la colaboración intensiva y autoorganización del equipo.

OpenUp divide el proyecto en iteraciones, planeadas, dentro de un marco de tiempo, regularmente medida en semanas. Estas iteraciones se enfocan en entregar a los interesados funcionalidades de manera predictiva. El plan de iteración define que se entregará en cada iteración; el resultado son entregables de software con funcionalidades nuevas y completas.

La estructura de OpenUp se enfoca en cuatro fases principales: concepción, elaboración, construcción y transición. El ciclo de vida provee a los interesados y miembros del equipo la visibilidad necesaria para la toma de decisiones a lo largo del proyecto.

Figura 14. **Diagrama del ciclo de vida de un proyecto de software en OpenUp**



Fuente: http://epf.eclipse.org/wikis/openup. Consulta: 01 de junio de 2104.

OpenUp se centra en cuatro principios básicos que son:

- Balancear las prioridades para maximizar el valor de los interesados
- Colaborar para alinear los intereses y compartir su conocimiento
- Enfocarse en la arquitectura tempranamente para minimizar riesgos y desarrollo organizativo.
- Incorporar continuamente la retro-alimentación para mejorar continuamente.

Además se considera que OpenUp es una metodología:

Mínima: es considerada ágil, los procesos ligeros promueven el desarrollo del software con los principios del agile manifiesto.

Completa: cubre todas las disciplinas esenciales del desarrollo de software esenciales.

Extensible: la metodología puede ser adoptaba por muchos tipos de proyecto; se pueden adaptar e incorporar aspectos a la metodología.

OpenUp: por ejemplo se pueden agregar roles, agregar artefactos o plantillas, guías tecnologías y otros.

4.2. Plan del proyecto

A continuación se presenta el plan de proyecto para el desarrollo de la aplicación para la detección de patrones musicales; este es una aproximación a priori de los recursos necesarios para llevar a cabo la implementación de la misma.

4.2.1. Introducción

La presente planificación del proyecto muestra las prácticas y medidas necesarias para completar el proyecto de forma uniforme y eficiente; este plan de proyecto contiene una planificación de alto nivel con hitos del proyecto, objetivos y fechas de entrega.

4.2.2. Organización del proyecto

Debido a las restricciones de personal se cuenta con poco personal de desarrollo y supervisión. A continuación se describen los roles principales del proyecto.

Roll	**Encargado**
Analista y desarrollador	Marvin Hernández Alonzo
Asesor	Luis Fernando Espino Barrios
Tutor	Luis Fernando Quiñónez López

4.2.3. Prácticas y medidas del proyecto

Para las iteraciones 2 y 3 se realizará un programa de demostración para comprobar que se han cumplido con los objetivos de las iteraciones.

Para las iteraciones 3, 4 y 5 se harán integración de las iteraciones y pruebas generales del sistema para comprobar que se han cumplido con los objetivos de las iteraciones.

La principal medida del proyecto se llevará a cabo midiendo el porcentaje de completado de los hitos de cada iteración.

4.2.4. Lecciones aprendidas

Cada problema se deberá documentar con su respectiva solución e impacto en el desarrollo del sistema, para esto cada iteración deberá tener su propia sección de problemas.

4.2.5. Hitos y objetivos del proyecto

Los objetivos principales del proyecto de software son los siguientes:

Iteración	Objetivos primarios	Fecha Inicio	Duración
I1	Definición de la arquitectura temprana.	11/06/2012	2d
I2	Captura de audio.	13/06/2012	5d
I3	Extracción de patrones musicales.	19/06/2012	4d
I4	Comparación de patrones.	28/06/2012	3d
I5	Desarrollo de la interfaz de usuario.	30/06/2012	2d
I6	Optimización general.	04/07/2012	2d
			18d

4.3. Plan de Iteración No 2: captura de audio

La meta de esta interacción es poder capturar audio del micrófono cercano a tiempo real para poder analizar dicho audio posteriormente.

4.3.1. Hitos claves

Los hitos o puntos clave de la interacción No 2 se describen a continuación:

Hito	Duración
Obtener Información de formato de audio soportado	8h
Capturar audio en tiempo real	15h
Colocar audio en buffer	5h
Demo: grabar un sonido en formato wav.	3h
Refactorizar y optimizar	2h

4.3.2. Objetivos de alto nivel

- Obtener audio directamente de la tarjeta de sonido sin utilizar buffer intermedias.

4.3.3. Problemas en el desarrollo

- Se presenta la complejidad de capturar audio sin grabar al disco duro, esto retrasa el desarrollo del sistema 5 horas más. La solución su utilizar un arreglo de bytes directamente al método que captura audio de java.

4.4. Plan de Iteración No 3: extracción de patrones musicales

Una vez que se captura el sonido el siguiente paso es analizarlo, en la iteración No 3 se analiza el sonido capturado y se extraen los patrones musicales si los tuviera; en esta etapa de desarrollo se utiliza el algoritmo de la transformada de Furier rápida.

4.4.1. Hitos claves

Los hitos o puntos clave de la iteración No 3 se describen a continuación:

Hito	Duración
Obtener la forma de onda del sonido	7h
Obtener la frecuencia del sonido	15h
Demo: mostrar la frecuencia del sonido de entrada en la consola	2h
Refactorizar y optimizar	2h

4.4.2. Objetivos de alto nivel

Obtener los patrones musicales del sonido que es capturado.

4.4.3. Problemas en el desarrollo

- La frecuencia del sonido capturado tiene una complejidad muy alta, por lo que se utiliza una librería de licencia GPL que implementa la transformada de Fourier rápida obtenida de googlecode.

- Debido a que dibujar la forma de onda del sonido que es capturado es solo cuestión estética se remueve del desarrollo para utilizar ese tiempo en la funcionalidad.

4.5. Plan de Iteración No 4: comparación de patrones

Una vez que el sonido ha sido capturado y se han extraído los patrones musicales del mismo es necesario compararlos con patrones musicales previamente definidos (por lo que las neuronas tienen un aprendizaje guiado) para poder establecer a que patrón musical pertenece la captura de sonido realizada.

4.5.1. Hitos claves

Los hitos o puntos clave y las duraciones de la iteración No 4 se describen a continuación:

Hito	Duración
Creación de neuronas comparadoras patrones de audio	10h
Entrenamiento asistido de neuronas.	16h
Pruebas y ajustes.	2h
Refactorizar, integrar y optimizar.	3h

4.5.2. Objetivos de alto nivel

- Crear y entrenar neuronas para que puedan diferenciar el tono y frecuencia de un sonido musical y así puedan diferencias un sonido de otro.

4.6. Plan de Iteración No 5: desarrollo de la interfaz de usuario

Una vez que se ha determinado que patrón musical o nota musical se ha capturado mediante el micrófono es necesario mostrar dicho resultado de alguna forma, por lo que la iteración No 5 es la creación de una interfaz de usuario grafica útil y agradable al usuario.

4.6.1. Hitos claves

Los Hitos o puntos clave y las duraciones de la iteración No 4 se describen a continuación:

Hito	Duración
Mostrar la cuerda y traste que se pulsa.	6h
Escribir la tablatura.	4h
Limpiar tablatura.	1h
Guardar tablatura.	1h
Refactorizar, integrar y optimizar.	3h

4.6.2. Objetivos de alto nivel

- Crear una interfaz de usuario amigable que escriba la tablatura de la cuerda y traste que se pulsan en la guitarra.

4.7. Plan de Iteración No 6: optimización general

En esta iteración se le data una revisión general al sistema, esto con el fin de realizar cambios para mejorar su rendimiento y apariencia.

4.7.1. Hitos claves

Los Hitos o puntos clave y las duraciones de la iteración No 4 se describen a continuación:

Hito	Duración
Remover métodos no utilizados.	1h
Corrección de nombres de variables.	1h
Re-estructuración de clases y métodos.	4h
Acelerador del recolector de basura de java.	2h
Pruebas del sistema.	4h

4.7.2. Objetivos de alto nivel

Debido a que esta iteración se centra solo en la parte gráfica del sistema Neuro-Traductor los objetivos de alto nivel son:

Crear una interfaz de usuario amigable que escriba la tablatura de la cuerda y traste que se pulsan en la guitarra.

CONCLUSIONES

1. La aplicación fue probada por cuatro individuos de los cuales todos opinaron que la aplicación tenía una precisión muy buena en la detección de los sonidos de la guitarra.

2. La interfaz gráfica fue calificada por los cuatro individuos como muy fácil de utilizar e intuitiva, resaltando que no era necesario un manual de usuario para la aplicación.

3. Debido a las restricciones del proyecto la latencia de detección es baja, entre tres y cuatro detecciones de patrones musicales por segundo, esto fue comentado por los individuos de prueba; opinaron que con una latencia de 8 detecciones por segundo la aplicación tendría una calidad muy buena y en posible que se mejorara en un futuro dejando al lector la inquietud de la optimización de la aplicación.

4. El campo de las redes neuronales es muy amplio y puede ser utilizado para la detección de patrones visuales, musicales o análisis numérico entre otras áreas, y resulta muy útil en el análisis de patrones musicales en donde solo el oído humano entrenado puede reconocer dichos patrones.

RECOMENDACIONES

1. Se propone ejecutar el programa en un ambiente con un nivel de ruido bajo, debido a que esto afecta la precisión del sistema.

2. En lo que cabe lo posible no utilizar efectos o distorsiones en el sonido producido por la guitarra, esto debido a que el cambio de tono producido por los efectos y distorsiones hacen el sonido “sucio” y difícil de detectar.

3. En lo posible interpretar piezas musicales con la guitarra afinada, si no está afinada se debe afinar con la misma aplicación Neuro-Traductor.

BIBLIOGRAFÍA

1. BOSKO, B. *Lógica borrosa*. Universidad de Murcia: España, 1993. 60 p.

2. DREYFUS, G. *Applications, neural networks methodology and applications*. 3a. ed. Paris: Eyrolles, 2010. 50 p.

3. FLÓREZ LÓPEZ, Raquel; FERNÁNDEZ FERNÁNDEZ, José Miguel. *Las redes neuronales artificiales*. España: Netbiblo, 2008. 50 p.

4. GONZALO, Luis. *Inteligencia humana e inteligencia artificial*. Madrid: Palabra, 1987. 31 p.

5. MORALES ADARAGA, Pablo SANCHO; Zaccagnini. *Psicología e inteligencia artificial*. Madrid: Trotta, 1994. 100 p.

6. ROJAS, Raúl. *Neural networks: a systematich introduction springer science & business media*. berlin: Springer-Verlag, 1996. 100 p.