UNIVERSIDAD NACIONAL DE INGENIERIA

FACULTAD DE INGENIERIA ELECTRICA Y ELECTRÓNICA

RECONOCIMIENTO DE CARACTERES MANUSCRITOS AISLADOS EN CAMPOS DE FORMULARIOS UTILIZANDO REDES NEURONALES

TESIS

PARA OPTAR EL GRADO DE MAESTRO EN CIENCIA
MENCIÓN : TELEMÁTICA

PRESENTADA POR:
ROXANA FLORES
QUISPE

LIMA - PERU

2006

TABLA DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

EXTRACTO

El presente trabajo de investigación intitulado "**Reconocimiento de caracteres manuscritos aislados en campos de formularios utilizando redes neuronales**", tiene por objetivo desarrollar un prototipo de sistema para el reconocimiento de caracteres manuscritos aislados en campos de formularios predefinidos utilizando Redes Neuronales, ya que este campo de la Inteligencia Artificial ha demostrado ser el más adecuado para el desarrollo de este tipo de aplicaciones.

El reconocimiento de caracteres manuscritos es una tarea relativamente sencilla para nosotros los humanos, puesto que reconocer un carácter manuscrito es una tarea que hemos desarrollado desde niños cuando aprendemos a leer y escribir, esto podría hacer pensar que enseñar a una computadora a reconocer caracteres manuscritos sea también una tarea sencilla; pues no lo es y la razón radica en que nosotros tenemos la habilidad de mirar subconscientemente las diferentes facetas de lo que estamos leyendo y usamos un multinivel de pensamiento para reconocer cada palabra; es decir que nosotros no solo tomamos en cuenta la forma individual de cada letra o caracter, sino que nuestro cerebro tiene la habilidad de juntar patrones y darle el significado correspondiente teniendo en cuenta el contexto.

Es así que surge la presente tesis, cuya importancia radica en la construcción de una herramienta de software desarrollada bajo el lenguaje de modelamiento unificado "UML" e implementada en Visual Basic, para la captura de caracteres manuscritos provenientes de los campos de formularios de datos, almacenando además toda la información reconocida en una base de datos para su posterior tratamiento y uso.

ABSTRACT

The present investigation "Recognition of isolated handwritten characters in fields from forms using Neuronal Networks", have as purpose to develop a prototype of system for the recognition of isolated handwritten characters in predefined fields from forms using Neuronal Networks, because the Artificial Intelligence has demonstrated to be the most suitable for these applications.

The Recognition of handwritten characters is a task relatively simple for us the humans, although it is a task that we have developed from children when we learn to read and to write, then it could make to think that to teach to a computer to recognize handwritten characters is also a simple task; but that is not true because we have the ability to watch subconsciously the different facets of the text that we are reading and we use a multilevel of thought to recognize each word; then we not only consider the individual form of each character, but also our brain has the ability to join patterns and to give them the corresponding meaning accord the context.

For that reason the present thesis has achieved to develop a software´s tool, using the Unified Modeling Language "UML" and his code source has been implemented in Visual Basic, to capture of handwritten characters from fields of data forms, and also this tool store all the recognized information in a data base for his later use.

CAPITULO I

INTRODUCCIÓN

1.1 Planteamiento y Formulación del Problema

Actualmente son numerosas las instituciones que utilizan una gran cantidad de formularios predefinidos en el desarrollo de sus operaciones diarias con la finalidad de recolectar datos, debido a la forma estructurada en que éstos pueden presentar la información, puesto que un formulario consta de una plantilla que contiene pequeñas casillas en blanco en las que los usuarios y/o clientes deben escribir ya sea a mano o a máquina la información requerida, incluso para una mejor administración, dichos formularios pueden contar con un código de clasificación en la parte superior de la hoja o tomar un color diferente que les ayude a ser diferenciados a la hora de ser procesados; tal es el caso de los documentos de la declaración de Renta.

Una vez que el usuario y/o cliente ha terminado de rellenar la información en el formulario, ésta debe ser procesada manualmente; y si es el caso cada uno de estos datos llegan a conformar los datos de entrada de algún sistema de información y por ende constituir parte de una Base de Datos. Cabe mencionar que este proceso manual de introducir los datos desde el formulario hacia un sistema muchas veces es susceptible de errores ya sean de digitación o de lectura por parte del encargado y peor aun cuando se trata de procesar gran cantidad de información o cuando el formulario es bastante denso (contenga bastante información), la demanda del tiempo se torna considerable.

Para dar solución a este tipo de problemas aparecieron los llamados Sistemas de Reconocimiento Óptico "OCR", cuyo objetivo es reconocer y procesar una

imagen digitalizada que contiene caracteres escritos almacenándolos en un archivo de texto plausible de ser utilizado por algún procesador de texto, estando muchos de estos sistemas están basados en Redes Neuronales[1].

Así mismo, también existen sistemas OCR para el reconocimiento de caracteres impresos o de máquina y su uso esta muy extendido, pero no sucede así con los caracteres manuscritos debido a las infinitas irregularidades que se puedan presentar en la escritura. Esto hace que los resultados tengan un elevado porcentaje de fallos o que los recursos a utilizar tengan un elevado costo, motivo por el cual los trabajos que se han realizado en este terreno no han tenido demasiado éxito.

Considerando la importancia de la información procedente de estos formularios, así como de la creciente demanda de sistemas OCR capaces de convertir información manuscrita en datos lógicos manejables por un computador, se construyó una herramienta off-line para el reconocimiento de caracteres manuscritos aislados en campos de formularios predefinidos mediante el uso de redes neuronales competitivas, puesto que en este tipo de redes las neuronas compiten y cooperan unas con otras con el fin de llevar a cabo una tarea dada, además con este tipo de aprendizaje se pretende que cuando se presenta a la red cierta información de entrada, sólo una de las neuronas de la capa de salida de la red neuronal se activa, alcanzando su valor de respuesta máximo y el resto quedan anuladas siendo forzadas a sus valores de respuesta mínimos, lo cual constituyó una gran ayuda para alcanzar el objetivo de que la red neuronal reconozca uno a uno los caracteres manuscritos, dando como salida a la neurona ganadora que representa al carácter reconocido.

1.2 Justificación

En el mercado actual existen sistemas para el reconocimiento de caracteres impresos o a máquina quedando un vacío importante en el terreno de los caracteres manuscritos, así mismo el proceso de obtención de información de estos formularios en forma automatizada es también casi nulo y con el desarrollo de esta tesis se pretende cubrir estas carencias importantes.

[1] De acuerdo con Werbos (1998) el 50% de los sistemas OCR se basan en Redes Neuronales

Además el ingreso de información a una Base de Datos en forma manual demanda mayor tiempo comparado con el uso de una aplicación software, que inclusive puede minimizar el número de errores de digitación y lectura de datos.

Por otro lado, con el desarrollo de este software de aplicación se pretende lograr la reducción de costos mediante la utilización de la misma para con diferentes tipos de plantillas de formularios, lo que supone además una notable mejora y modernización en cuanto a la infraestructura y servicio se refiere.

Así mismo, para el desarrollo de la presente tesis, se aplicó una Metodología de desarrollo de software basado en UML, lo cual constituirá un aporte para la construcción de futuras aplicaciones de software de ingeniería.

1.3 Objetivos de la Tesis

1.3.1 Objetivo General

Desarrollar un prototipo de sistema para el reconocimiento de caracteres manuscritos aislados en campos de formularios utilizando Redes Neuronales.

1.3.2 Objetivos Específicos

Diseñar e implementar la Red Neuronal para el reconocimiento de caracteres manuscritos aislados.

Diseñar e implementar el mecanismo por el cual se determinará el posicionamiento de los caracteres en los recuadros de los campos de formularios, así como la organización de datos en el sistema.

Construir el prototipo de sistema para el reconocimiento de caracteres manuscritos en campos de formularios con una interfaz amigable.

1.4 Alcances y Limitaciones de la Investigación

El prototipo de sistema solo trabaja con plantillas de formularios que contengan casillas de texto en las que el usuario puede colocar información comprendida entre letras mayúsculas del alfabeto y los dígitos del 0 al 9.

Los formatos de los formularios deben estar en buen estado, es decir libre de marcas innecesarias y las líneas que bordean a las casillas de texto deberán ser legibles.

Los formularios a ser procesados deben cumplir con requisitos mínimos en cuanto al color, resolución, alineación, formato de imagen, nitidez e intensidad de brillo.

CAPITULO II

INTELIGENCIA ARTIFICIAL Y REDES NEURONALES

2.1 Principios de Inteligencia Artificial

La Inteligencia Artificial (IA) es una rama de las ciencias de la computación que estudia los fundamentos teóricos y prácticos del diseño de sistemas de computación inteligentes, esto es, sistemas que exhiben características inteligentes del ser humano, tales como: resolución de problemas, comprensión de lenguajes, aprendizaje, razonamiento, etc[2].

Algunos autores como Rich & Knight [1994], Stuart [1996], definen en forma general la IA como la capacidad que tienen las máquinas para realizar tareas que en el momento son realizadas por seres humanos; otros autores como Nebendah [1988], Delgado [1998], arrojan definiciones más completas y las definen cómo el campo de estudio que se enfoca en la explicación y emulación de la conducta inteligente en función de procesos computacionales basadas en la experiencia y el conocimiento continuo del ambiente.

2.2 Campos de la Inteligencia Artificial

Dentro de los principales campos de la Inteligencia Artificial tenemos: Teoría de Juegos, Visión Artificial, Robótica, Comprensión del Lenguaje, Traducción Automática, Programación Automática, Sistemas Expertos, Redes Neuronales

[2] Maynard Kong, Inteligencia Artificial 1993.

y Algoritmos Genéticos. A su vez todos estos campos se encuentran dentro de las siguientes categorías.

1. Técnicas básicas, así llamadas por encontrarse en la base de diversas aplicaciones de la IA. Entre estas se encuentran Búsqueda Heurística de Soluciones, Representación del Conocimiento, Deducción Automática, Programación Simbólica (LISP) y Redes Neuronales, las que se consideran como base de muchas aplicaciones. En su mayoría, no necesita conocerla el usuario final, sino los profesionales que se dedican a su aplicación y la generación de aplicaciones comerciales.

2. Tecnologías, o combinaciones de varias técnicas básicas, orientadas a resolver familias de problemas, son más especializadas que las técnicas básicas y están más cerca de las aplicaciones finales. Se pueden mencionar dentro de estas a la Robótica y Visión, Lenguaje Natural, Sistemas Expertos.

3. Clases o tipos de aplicaciones: Diagnóstico (sistemas que permiten encontrar causas de fallas ya sea en una línea de producción o de enfermedades en una persona.), Predicción (sistemas de autocontrol de reactores atómicos), Secuenciamiento de operaciones ("Scheduling"), Diseño, Interpretación de datos. Todas ellas son familias de problemas tipo.

4. Todos estos campos de aplicaciones de Inteligencia Artificial son aplicables a diversas áreas tales como: Ingeniería, Medicina, Sistemas de Manufactura, Administración, Apoyo a la Toma de Decisiones Gerenciales, etc. Todas caen dentro de las áreas de los sistemas computacionales, pero que se consideran como clientes de la Inteligencia Artificial.

2.3 Ramas y Tendencias de la Inteligencia Artificial

Dentro de la I.A. podemos distinguir tres grandes ramas según el método que utilizan:

Simbólica: Su característica esencial es que representa los conceptos mediante símbolos. El razonamiento consiste en manipular los símbolos de acuerdo con ciertas reglas obtenidas a partir de principios lógicos.

Probabilística: Se basa en el cálculo de probabilidades, concretamente en la aplicación del teorema de Bayes, para realizar un razonamiento. Aunque la inferencia se desarrolla a nivel numérico (cuantitativo), hay luego un paso al nivel cualitativo, pues de otro modo estaríamos en el campo de las matemáticas, no en el de la inteligencia artificial.

Conexionista: Consiste en utilizar una red de neuronas artificiales (pueden ser procesadores reales o simulados mediante ordenador). Suelen utilizarse sobre todo en tareas de clasificación. La red se programa a través de ejemplos y la información se almacena en forma de pesos (números) asociados a los enlaces entre neuronas.

En general, cuando se habla de inteligencia artificial se trata de la rama simbólica, pues las redes neuronales tienen un origen independiente (por cierto, anterior a la I.A. simbólica) y se han desarrollado tanto en los últimos años que constituyen un campo amplio y bien diferenciado, dotado de gran vitalidad. Los esfuerzos por integrar ambas corrientes aún no han dado los frutos deseados. Por otra parte, la rama probabilística permanece unida al tronco de la I.A., pues como ya hemos mencionado, necesita utilizar los métodos simbólicos para pasar del nivel cuantitativo al cualitativo.

Si en vez de considerar los métodos atendemos a los objetivos, podemos reconocer en la I.A. tres tendencias:

Psicológica: Se fundamenta en el estudio de la mente humana con el fin de reproducir sus capacidades mediante sistemas informáticos. La tendencia equivalente a ésta dentro de la computación neuronal sería la biológica, basada en el estudio de los sistemas nerviosos animales.

Teórica: Estudia los principios teóricos de la inteligencia que deben regir tanto al razonamiento humano como el automático. Se apoya sobre todo en el estudio de la lógica, la epistemología, la teoría de la probabilidad, la lingüística, etc.

Pragmática: Supone que la forma más eficaz de desarrollar la I.A. consiste en tratar de resolver problemas concretos y, a partir de la experiencia, generalizar los resultados para poder aplicarlos a áreas diferentes.

2.4 Redes Neuronales Artificiales

Las redes neuronales artificiales o modelos conexionistas como también se les conoce están inspiradas en el comportamiento del cerebro humano. Una *red neuronal* puede verse como una *máquina* diseñada originalmente para modelar la forma en que el sistema nervioso de un ser vivo realiza una determinada tarea.

Una Red Neuronal, puede ser definida como una estructura distribuida, de procesamiento paralelo, formada de neuronas artificiales (llamados también elementos de procesamiento), interconectados por un gran numero de conexiones (sinapsis), los cuales son usados para almacenar conocimiento que esta disponible para poder ser usado.

2.5 Estructura de una Red Neuronal Artificial

La estructura de una Red Neuronal, está formada por un conjunto de unidades de procesamiento interconectadas llamadas *neuronas*.

Una neurona artificial es una unidad de procesamiento de información de las redes neuronales. El modelo de neurona más conocido es de McCulloch- Pitts.

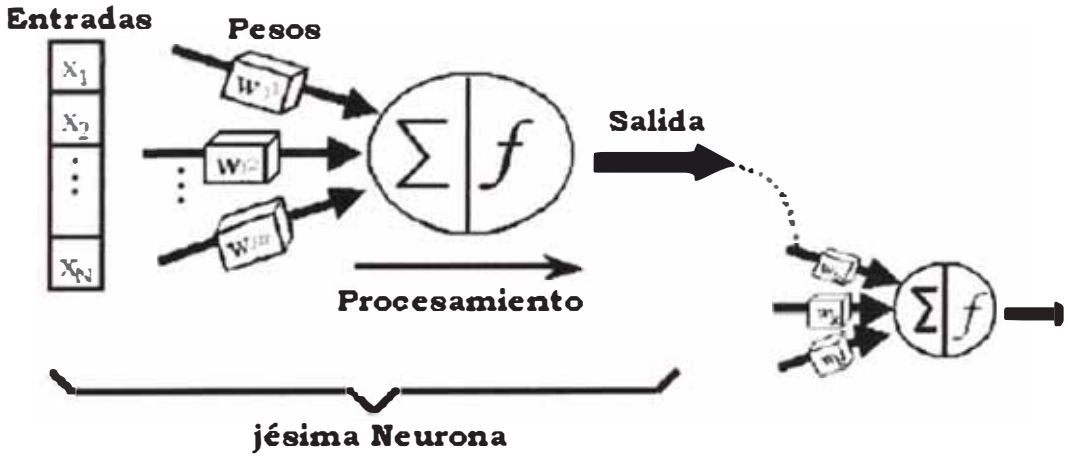


Figura 2.1 Estructura de una Neurona Artificial

Cada neurona recibe como entrada un conjunto de señales discretas o continuas, representadas por las variables x_1 x_2... x_N, las cuales están asociadas a los pesos o pesos sinápticos, que son representados por las variables w_{ji}, y son ellos en los que se guarda la mayor parte del conocimiento que la red neuronal

tiene sobre la tarea en cuestión, además determinan el nivel de influencia de la neurona j para la neurona i.

Existen dos etapas de procesamiento para cada neurona: "suma y activación".

- En la primera etapa, las señales de entrada x_j y los pesos w_{ij} son combinadas mediante la siguiente sumatoria:

$$y_i = \sum_{j=1}^{N} w_{ij} x_i$$

Donde y_i representa al estado interno de la *i-ésima* neurona.

- En la segunda etapa, la salida de la neurona es generada a través de la aplicación de una función llamada función de activación.

$$x_i = f(y_i)$$

Donde la salida de la neurona es representado por *xi* y *f* corresponde a la función de activación aplicada al estado interno de la neurona, que tiene como objetivo limitar el nivel de activación de entre [-1 ,1] o [0 1], en el caso de *xi* sea un valor continuo y si *xi* es discreto entonces el nivel de activación puede ser : {-1,1} o {0,1}

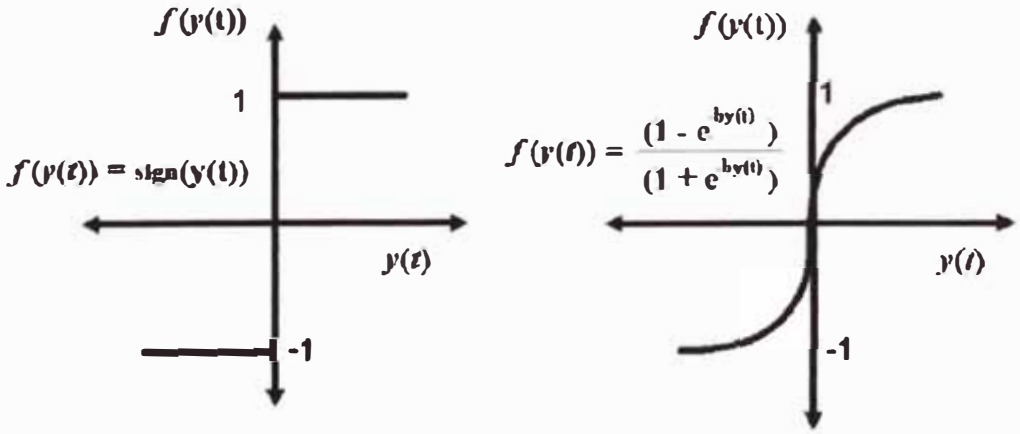


Figura 2.2 Funciones de Activación : función de grado y tangente hiperbólica

Existen varios tipos de función de activación, la figura 2.2 muestra dos funciones de activación más usadas: la función de grado y la tangente

hiperbólica. Como se vio en la primera figura 2.1, la salida de una neurona puede ser la entrada de otra. Generalmente, una red neuronal se forma por muchas neuronas de alguna forma acoplados.

2.6 Características de las Redes Neuronales Artificiales

Debido a su constitución y fundamentos, las redes neuronales artificiales presentan un gran número de características similares a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que presentan información irrelevante, etc.

Entre sus propiedades más importantes se puede citar:

Procesamiento Paralelo. Esta característica resulta esencial, ya que así como el cerebro para preprocesar una imagen compuesta por millones de píxeles (los que representan los conos y bastones de la retina), tarda un tiempo en extraer sus rasgos característicos, analizarla e interpretarla, y en todo este proceso de visión intervienen miles de millones de neuronas, operando en paralelo sobre la totalidad de la imagen. En una red Neuronal Artificial, sucede lo mismo ya que todas las neuronas trabajan en forma paralela para realizar una determinada tarea.

Memoria Distribuida. Mientras en un computador la información ocupa posiciones de memoria bien definidas, en los sistemas neuronales la información se encuentra distribuida por las sinapsis (pesos sinápticos) de la red, de modo que si una sinapsis se daña solamente perdemos una pequeña parte de la información

Adaptabilidad. Las redes neuronales son capaces de reajustar sus pesos para adaptarse a cambios en el entorno. Esto es especialmente útil cuando el entorno que suministra los datos de entrada es *no estacionario*, es decir, algunas de sus propiedades varían con el tiempo

Auto Organización. Las redes neuronales utilizan su capacidad de aprendizaje adaptativo para organizar la información que reciben durante el aprendizaje y/o la operación. Mientras el aprendizaje es la modificación de

cada elemento procesal, la autoorganización consiste en la modificación de la red neuronal completa para llevar a cabo un objetivo específico. Esta autoorganización da lugar a la generalización: facultad de responder apropiadamente cuando se les presentan datos o situaciones a los que no habían sido expuestas anteriormente. El sistema puede generalizar la entrada para obtener una respuesta. Esta característica es de especial importancia cuando se tiene que solucionar problemas para los cuales la información de entrada es poco clara; permite además que el sistema dé una solución incluso cuando la información de entrada está especificada en forma incompleta.

Tolerancia ante fallos. Una red neuronal es tolerante ante fallos en el sentido de que los posibles fallos operacionales en partes de la red solo afectan débilmente al rendimiento de esta. Esta propiedad es debida a la naturaleza distribuida de la información almacenada o procesada en la red neuronal.

2.7 Tipos de Redes Neuronales

Entre los principales tipos de Redes Neuronales podemos mencionar:

A) Perceptrón

La red tipo Perceptrón fue inventada por el psicólogo Frank Rosenblatt en el año 1957. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general.

Rosenblatt creía que la conectividad existente en las redes biológicas tiene un elevado porcentaje de aleatoriedad, por lo que se oponía al análisis de McCulloch Pitts (quien desarrollo la primera red neuronal conocida, en 1943) en el cual se empleaba lógica simbólica para analizar estructuras bastante idealizadas. Rosenblatt opinaba que la herramienta de análisis más apropiada era la teoría de probabilidades, y esto lo llevó a una teoría de *separabilidad estadística* que utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatorias.

La estructura de un Perceptrón sencillo es similar a la del elemento general de procesamiento en la que se observa la adición de una condición umbral en la salida. Si la entrada neta, a esta condición es mayor que el valor umbral, la salida de la red es 1, en caso contrario es 0, que esta definido por una función de salida de la red llamada función umbral o función de transferencia, la cual se muestra a continuación:

$$f(salida) = \begin{matrix} 1 \; si \; salida > \theta \\ 0 \; si \; salida < \theta \end{matrix}$$

El Perceptrón es aún hoy una red de gran importancia, pues con base en su estructura se han desarrollado otros modelos de red neuronal como la red Adaline y las redes multicapa.

B) Perceptrón Multicapa

El Perceptrón multicapa es una red con múltiples entradas y una entrada adicional representada por la ganancia "*b*", este esquema general se ve en la ***figura 2.3*** en donde se notan las conexiones entre sus nodos de entrada y las neuronas de salida.

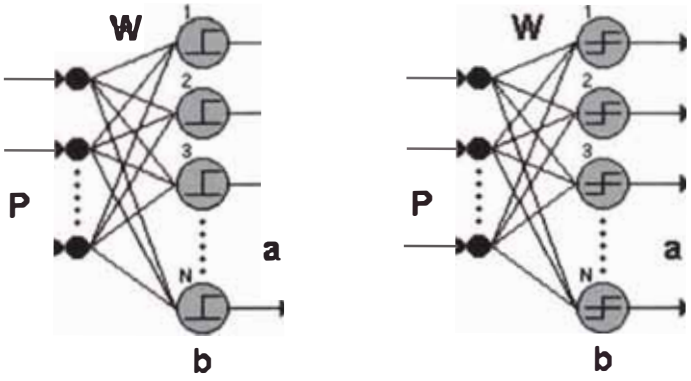


Figura 2.3 Conexiones del Perceptrón

Un Perceptrón multicapa es una red con alimentación hacia delante, compuesta de varias capas de neuronas entre la entrada y la salida de la misma, esta red permite establecer regiones de decisión mucho más complejas que las de dos semiplanos, como lo hace el Perceptrón de un solo nivel.

La salida de la red se activará sólo si las salidas de todos los nodos de la segunda capa están activos, esto equivale a ejecutar la función lógica AND en el nodo de salida, resultando una región de decisión intersección de todos los semiplanos formados en el nivel anterior. La región de decisión resultante de la intersección será una región convexa con un número de lados a lo sumo igual al número de neuronas de la segunda capa.

C) Adaline

Al mismo tiempo que Frank Rosenblatt trabajaba en el modelo del Perceptrón Bernard Widrow y su estudiante Marcian Hoff introdujeron el modelo de la red Adaline y su regla de aprendizaje llamada algoritmo LMS (Least Mean Squar).

La red Adaline es similar al Perceptrón, excepto en su función de transferencia, la cual es una función de tipo lineal en lugar de un limitador fuerte como en el caso del Perceptrón.

El elemento de procesamiento realiza la suma de los productos de los vectores de entrada y de pesos, y aplica una función de salida para obtener un único valor de salida, el cual debido a su función de transferencia lineal será +1 si la sumatoria es positiva o −1 si la salida de la sumatoria es negativa. En términos generales la salida de la red está dada por:

$$a = W^T p$$

En este caso, la salida es la función unidad al igual que la función de activación; el uso de la función identidad como función de salida y como función de activación significa que la salida es igual a la activación, que es la misma entrada neta al elemento.

La red Adaline presenta la misma limitación del Perceptrón en cuanto al tipo de problemas que pueden resolver, ambas redes pueden solo resolver problemas linealmente separables, sin embargo el algoritmo LMS es más potente que la regla de aprendizaje del Perceptrón ya que minimiza el error medio cuadrático, la regla sirvió de inspiración para el desarrollo de otros algoritmos, este es el gran aporte de esta red.

D) Backpropagation

El primer algoritmo de entrenamiento para redes multicapa fue desarrollado por Paul Werbos en 1974, este se desarrolló en un contexto general, para cualquier tipo de redes, siendo las redes neuronales una aplicación especial, razón por la cual el algoritmo no fue aceptado dentro de la comunidad de desarrolladores de redes neuronales. Fue solo hasta mediados de los años 80 cuando el algoritmo Backpropagation o algoritmo de propagación inversa fue redescubierto al mismo tiempo por varios investigadores, David Rumelhart, Geoffrey Hinton y Ronal Williams, David Parker y Yann Le Cun. El algoritmo se popularizó cuando fue incluido en el libro "Parallel Distributed Processing Group" por los sicólogos David Rumelhart y James McClelland. La publicación de este libro trajo consigo un auge en las investigaciones con redes neuronales, siendo la Backpropagation una de las redes más ampliamente empleadas, aun en nuestros días.

Uno de los grandes avances logrados con la Backpropagation es que esta red aprovecha la naturaleza paralela de las redes neuronales para reducir el tiempo requerido por un procesador secuencial para determinar la correspondencia entre unos patrones dados. Es decir que la red Backpropagation es capaz de examinar todos los patrones en paralelo, con la capacidad de adaptarse así mismo para aprender la relación entre un conjunto de patrones dado como ejemplo y ser capaz de aplicar la misma relación a nuevos patrones de entrada.

La Red Backpropagation emplea un ciclo propagación – adaptación de dos fases, es decir una vez que se ha aplicado un patrón a la entrada de la red como estímulo, este se propaga desde la primera capa a través de las capas superiores de la red, hasta generar una salida. La señal de salida se compara con la salida deseada y se calcula una señal de error para cada una de las salidas. Las salidas de error se propagan hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa oculta que contribuyen directamente a la salida.

Sin embargo las neuronas de la capa oculta solo reciben una fracción de la señal total del error, basándose aproximadamente en la contribución relativa que haya aportado cada neurona a la salida original. Este proceso se repite,

capa por capa, hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total. Basándose en la señal de error percibida, se actualizan los pesos de conexión de cada neurona, para hacer que la red converja hacia un estado que permita clasificar correctamente todos los patrones de entrenamiento.

La importancia de este proceso consiste en que, a medida que se entrena la red, las neuronas de las capas intermedias se organizan a sí mismas de tal modo que las distintas neuronas aprenden a reconocer distintas características del espacio total de entrada.

E) Aprendizaje Asociado

El aprendizaje asociativo fue inicialmente estudiado por escuelas de Sicología, las cuales se dedicaron a estudiar las relaciones entre el comportamiento humano y el comportamiento animal.

Basado en este tipo de comportamiento, Donald Hebb postuló el siguiente principio conocido como la regla de Hebb:

" Cuando un axón de una celda A está lo suficientemente cerca de otra celda B como para excitarla y repetidamente ocasiona su activación, un cambio metabólico se presenta en una o ambas celdas, tal que la eficiencia de A, como celda excitadora de B, se incrementa". Con el término celda, Hebb se refería a un conjunto de neuronas fuertemente conexionadas a través de una estructura compleja, la eficiencia podría identificarse con la intensidad o magnitud de la conexión, es decir el peso.

Este postulado aplicado a redes asociativas, marcó el inicio del aprendizaje no supervisado. Un gran número de investigadores ha contribuido al aprendizaje asociativo, en particular Tuevo Kohonen, James Anderson y Stephen Grossberg. Anderson y Kohonen desarrollaron independientemente el asociador lineal a finales de los años 60's y Grossberg introdujo la red asociativa no lineal durante este mismo periodo.

F) Redes competitivas

En las redes con aprendizaje competitivo (y cooperativo), suele decirse que las neuronas compiten (y cooperan) unas con otras con el fin de llevar a cabo una tarea dada. Con este tipo de aprendizaje se pretende que cuando se presente a la red cierta información de entrada, sólo una de las neuronas de la capa de salida de la red neuronal (o una por cierto grupo de neuronas) se active, alcanzando su valor de respuesta máximo. Por tanto las neuronas compiten para activarse quedando finalmente una (o una por grupo) como neurona vencedora y el resto quedan anuladas y siendo forzadas a sus valores de respuesta mínimos.

La competición entre neuronas se realiza en todas las capas de la red, existiendo en estas redes neuronas con conexiones de autoexitación (signo positivo) y conexiones de inhibición (signo negativo) por parte de neuronas vecinas.

El objetivo de este aprendizaje es categorizar los datos que se introducen en la red, de esta forma las informaciones similares son clasificadas formando parte de la misma categoría y por tanto deben activar la misma neurona de salida. Las clases o categorías pueden ser creadas por la propia red o por eventos externos, puesto que se trata de un aprendizaje no supervisado en el primer caso y aprendizaje supervisado en el segundo.

Otra forma de aplicar este tipo de aprendizaje fue propuesta por Rumelhart y Zisper en 1985, quienes utilizaban redes multicapa dividiendo cada capa en grupos de neuronas, de tal forma que éstas disponían de conexiones inhibitorias con otras neuronas de su mismo grupo y conexiones excitadoras con las neuronas de la siguiente capa; en una red de este tipo, después de recibir diferentes informaciones de entrada, cada neurona en cada grupo se especializa en la respuesta a determinadas características de los datos de entrada. En este tipo de redes cada neurona tiene asignado un peso total (suma de todos los pesos de las conexiones que tiene a su entrada), el aprendizaje afecta sólo a las neuronas ganadoras (activas), en las que se redistribuye el peso total entre sus conexiones y se sustrae una porción de los pesos de todas las conexiones que llegan a la neurona vencedora, repartiendo esta cantidad por igual entre todas las conexiones procedentes

de unidades activas, por tanto la variación del peso de una conexión entre una unidad *i* y otra *j* será nula si la neurona *j* no recibe excitación por parte de la neurona *i* (no vence en presencia de un estímulo por parte de *i*) y se modificará (se reforzará) si es excitada por dicha neurona.

G) Redes Recurrentes

En la década de los 80´s con el fin de estudiar procesos que involucran sistemas gobernados por ecuaciones diferenciales no lineales surge la teoría clásica de control geométrico basada en la geometría diferencial; y simultáneamente renace el estudio de las Redes Neuronales debido al redescubrimiento del algoritmo Backpropagation, este hecho sumado al fracaso de las metodologías tradicionales aplicadas a la inteligencia artificial y a la disponibilidad de herramientas computacionales de bajo costo permitieron el desarrollo las redes neuronales recurrentes cuya principal aplicación es el control e identificación de sistemas no lineales. Este desarrollo es posible debido a que las propiedades matemáticas de las redes recurrentes están enmarcadas en las mismas propiedades que fundamentan el control geométrico, la primera red neuronal recurrente de naturaleza dinámica fue propuesta por Hopfield en 1984 bajo el contexto de las memorias asociativas.

2.8 Campos de Aplicación de las Redes Neuronales

Los campos de aplicación de las redes neuronales son habitualmente todos aquellos en los que se utilizan o pueden utilizarse modelos estadísticos y/o lineales. Entre algunos campos de aplicación se pueden mencionar:

Finanzas

- Predicción de índices.
- Detección de fraudes.
- Riesgo crediticio, clasificación.
- Predicción de la rentabilidad de acciones.
- Previsión de la evolución de los precios.
- Valoración del riesgo de los créditos.
- Identificación de las falsificaciones.

Negocios

- Marketing.
- Venta cruzada.
- Campanas de venta.

Tratamiento de textos y proceso de formas

- Reconocimiento de caracteres impresos mecánicamente.
- Reconocimiento de gráficos.
- Reconocimiento de caracteres escritos a mano.
- Reconocimiento de escritura manual cursiva.

Alimentación

- Análisis de olor y aroma.
- Perfilamiento de clientes en función de la compra.
- Desarrollo de productos.
- Control de Calidad.

Energía

- Predicción consumo eléctrico.
- Distribución recursos hidráulicos para la producción eléctrica.
- Predicción consumo de gas ciudad.

Empresa

- Explotación de base de datos (minería de datos).
- Optimización de plazas y horarios en líneas aéreas.

Industria manufacturera

- Control de procesos, control de calidad, control de robots.
- Sistemas de control automatizado (visión artificial, sensores de temperatura, presión, gas, etc.).
- Control de producción en líneas de proceso.

Medicina y salud

- Analizadores del habla para la ayuda en la audición.
- Diagnostico y tratamiento a partir de síntomas y/o datos analíticos (electrocardiograma, encefalograma, análisis sanguíneo).
- Monitorización en cirugía.

- Predicción a reacciones adversas a medicamentos.
- Lectores de Rayos X.
- Obtención del Modelo de Retina.
- Desarrollo de medicamentos.
- Distribución de recursos.

Ciencia e Ingeniería

- Análisis de datos y clasificación.
- Ingeniería Química.
- Ingeniería Eléctrica.
- Climatología, previsión del tiempo.

Transportes y Comunicaciones.

- Optimización de rutas.
- Optimización en la distribución de recursos.

Militar

- Clasificación de señales por radar.
- Optimización del uso de recursos escasos.
- Creación de armas inteligentes.

2.9 Algoritmos de Aprendizaje

El proceso mediante el cual se ajustan los pesos para lograr un determinado objetivo se denomina *aprendizaje* o *entrenamiento* y el procedimiento concreto utilizado para ello se conoce como *algoritmo de aprendizaje* o *algoritmo de entrenamiento*, es decir los procedimientos para determinar las conexiones entre neuronas reciben el nombre de algoritmos de aprendizaje, ya que es en los pesos donde reside el "conocimiento" de una red neuronal. Así este proceso de aprendizaje en la red neuronal puede verse como el problema de modificar la arquitectura de la red y el aprendizaje de los pesos de las conexiones, de tal manera que la red pueda realizar eficientemente una tarea específica.

Ahora bien, todos los métodos de aprendizaje pueden ser clasificados en las siguientes categorías:

2.9.1 Aprendizaje supervisado:

A la red se le ofrece una respuesta correcta para cada patrón de entrada. Los pesos se ajustan para aproximar la respuesta de la red a la respuesta correcta conocida.

En este tipo de aprendizaje se suelen considerar, a su vez, tres formas de llevarlo a cabo: aprendizaje por corrección de error, aprendizaje por refuerzo y aprendizaje estocástico.

2.9.1.1. Aprendizaje por corrección de error.

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida de la red; es decir, en función del error cometido en la salida [Figura 2.4].

Una regla o algoritmo simple de aprendizaje por corrección de error podría ser el siguiente:

$$\Delta w_{ji} = \alpha y_i (d_j - y_j)$$

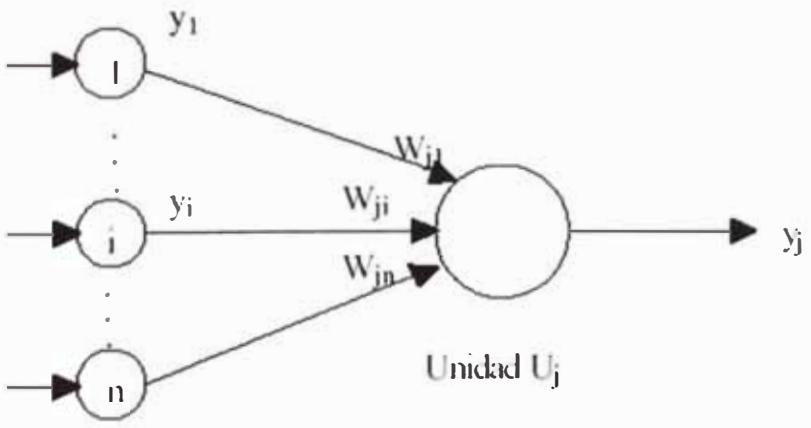


Figura 2.4 Aprendizaje por Corrección de error

Donde:

Δwij : Es la variación en el peso de la conexión entre las neuronas i y j ($\Delta Wij = Wij^{actual} - Wij^{anterior}$).

yi Valor de salida de la neurona i.
dj : Valor de salida deseado para la neurona j.
yj : Valor de salida obtenido en la neurona j.
α : Factor de aprendizaje ($0<\alpha<=1$) que regula la velocidad del aprendizaje.

2.9.1.2. Aprendizaje por refuerzo.

Se trata de un aprendizaje supervisado, más lento que el anterior, que se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado; es decir, de no indicar durante el entrenamiento exactamente la salida que se desea que proporcione la red ante una determinada entrada.

En el aprendizaje por refuerzo la función del supervisor se reduce a indicar mediante una señal de refuerzo si la salida obtenida en la red se ajusta a la deseada (éxito =+ 1 o fracaso =- 1), y en función de ello se ajustan los pesos basándose en un mecanismo de probabilidades. Se podría decir que en este tipo de aprendizaje la función del supervisor se asemeja más a la de un crítico (que opina sobre la respuesta de la red) que a la de un maestro (que indica a la red la respuesta concreta de que debe generar), como ocurriría en el caso de supervisión por corrección de error.

2.9.1.3. Aprendizaje estocástico.

Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidades.

En el aprendizaje estocástico se suele hacer una analogía con la termodinámica. Se asocia a la red neuronal con un sólido físico que tiene un cierto estado energético. En el caso de la red, la energía representa el grado de estabilidad de la misma, de tal forma que el estado de mínima energía correspondería a una situación en la que los pesos de las conexiones consiguen que su funcionamiento sea el que más se ajuste al comportamiento deseado.

Según lo anterior, el aprendizaje consistiría en realizar un cambio aleatorio de los valores de los pesos y determinar la energía de la red. Si la energía es menor después del cambio: es decir, si el comportamiento de la red se acerca al deseado, se acepta el cambio. Si, por el contrario, la energía no es menor, se acepta el cambio según una distribución de probabilidades.

2.9.2 Aprendizaje no-supervisado:

Las redes con aprendizaje no supervisado (también conocido como autosupervisado) no requieren influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello, suele decirse que estas redes son capaces de autoorganizarse.

Estas redes deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presenten en su entrada. Puesto que no hay un supervisor que indique a la red la respuesta que debe generar ante una entrada concreta, cabe preguntarse qué es lo que la red genera en estos casos. Existen varias posibilidades en cuanto a la interpretación de la salida de estas redes, que dependen de su estructura y del algoritmo de aprendizaje empleado.

En algunos casos, la salida representa el grado de familiaridad entre la información que se le está presentando a la entrada y las informaciones que se le han mostrado hasta entonces. En otro caso, podría realizar una clusterización (clustering) o establecimiento de categorías, indicando la red a la salida a qué categoría pertenece la información presentada a la entrada, siendo la propia red la que debe encontrar las categorías apropiadas a partir de correlaciones entre las informaciones presentadas. Una variación de esta categorización es el prototipado. En este caso la red obtiene ejemplares o prototipos representantes de las clases a las que pertenecen las informaciones de entrada.

En general, en cuanto a los algoritmos de aprendizaje no supervisado, se suelen considerar dos tipos que dan lugar a los siguientes aprendizajes:

- Aprendizaje Hebbiano.

- Aprendizaje Competitivo y Cooperativo.

  En el primer caso, normalmente se pretende medir la familiaridad o extraer características de los datos de entrada, mientras que el segundo suele orientarse hacia la clusterización o clasificación de dichos datos.

  En el siguiente apartado se dará énfasis al estudio del aprendizaje competitivo puesto que éste corresponde a la Red Neuronal competitiva, la cual se utilizó en el desarrollo de esta Tesis.

2.9.3 Aprendizaje híbrido:

Combina los dos anteriores. Parte de los pesos se determinan mediante un proceso supervisado, mientras que el resto se obtienen mediante un aprendizaje no-supervisado.

2.10 Algoritmo de Aprendizaje Competitivo

Sin duda la cualidad más atrayente de las Redes Neuronales es su habilidad para aprender, el aprendizaje en este contexto esta definido como un cambio en los valores de los pesos de las conexiones y gracias a ello la red neuronal es capaz de capturar información.

El aprendizaje competitivo fue introducido por Grossberg (1970) y Malsburg (1973) y fue extensamente estudiado por Amari y Takeuchi (1978), Amari (1983), y Grossberg (1982), el cual es un método de creación automatizada de clases para un conjunto de patrones de entrada.

En las redes con aprendizaje competitivo (y cooperativo), suele decirse que las neuronas compiten (y cooperan) unas con otras con el fin de llevar a cabo una tarea dada. Con este tipo de aprendizaje, se pretende que cuando se presente a la red cierta información de entrada, sólo una de las neuronas de salida de la red, o una por cierto grupo de neuronas, se active, es decir alcance su valor de respuesta máximo.

Por tanto, las neuronas compiten por activarse, quedando finalmente una, o una por grupo, como neurona vencedora (winner take all unit), quedando anulado el resto de las neuronas, las que son forzadas al valor de respuesta mínimo.

El objetivo de este aprendizaje es categorizar (clusterizar) los datos que se introducen en la red. De esta forma, las informaciones similares son clasificadas formando parte de la misma categoría, y por tanto deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado, a través de las correlaciones entre los datos de entrada.

A continuación se detalla el algoritmo desarrollado en la presente tesis, para ello previamente es necesario especificar algunos términos:

- F_x : Representa a la capa de entrada de la red neuronal.
- F_y : Representa a la capa de salida de la red neuronal, que a la vez representa una clase dada.
- W : Es el peso de la conexión entre los elementos de procesamiento de la capa de entrada y la capa de salida.
- EP : Elemento de procesamiento
- A_k : Patrón de entrada, por consiguiente es el patrón a reconocer.

El algoritmo de entrenamiento se resume en los siguientes pasos:

Paso 1 : Determinar el elemento de procesamiento "EP" ganador de la capa de salida "F_y".

Un patrón de entrada "A_k", el cuál esta conformado por el conjunto de EPs pasa a través de las conexiones desde la capa de entrada "F_x" hacia la capa de salida "F_y" mediante un encadenamiento hacia adelante, y cuyos EPs correspondientes de F_y se determinan mediante la siguiente ecuación:

$$Y_j = \sum_{i=1}^{n} X_i W_{ij}$$

donde:

Y_j : Es el j-ésimo EP de F_y, para un $j = 1, 2, \ldots, p$.

X_i : Es el i-ésimo EP de F_x, para un i = 1, 2, . . . , n.
W_{ij}: Es el valor del peso de la conexión entre X_i y Y_j

Cada conjunto de conexiones alrededor de F_y, se representa por un vector de referencia W_j, el cual está conformado por el conjunto de pesos $w_{1j}, w_{2j}, \ldots w_{nj}$. los mismos que corresponden a la clase j. Ahora bien, es este vector referencia W_j, quien debe prever que el EP de F_y con el valor más alto se active, el cual representará al EP ganador de F_y.

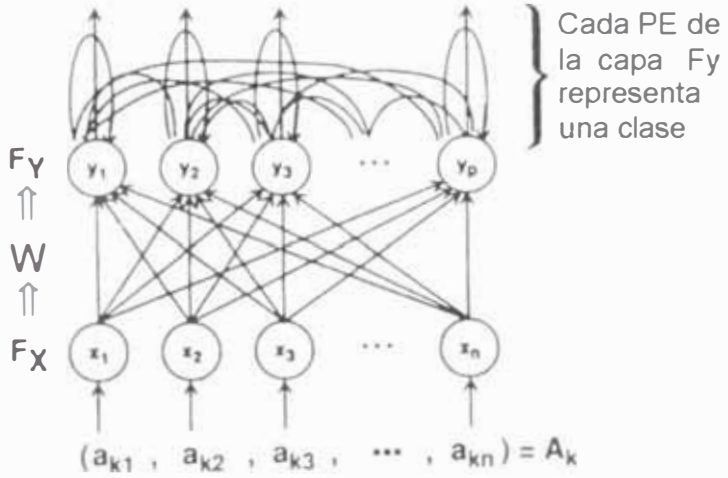


Figura. 2.5 Red Neuronal Competitiva

Además para un patrón de entrada A_k, *donde k va desde* 1,2,..., m, y sus vectores de referencia W_j, *donde j va desde* 1,2,...., p; se normalizarán sus correspondientes EPs de F_y, para tener un valor entre 0 y 1 según la siguiente ecuación:

$$0 \le \left(Y_j = A_k W_j = \sum_{i=1}^{n} a_{ki} w_{ij} \right) \le 1$$

Los valores del producto Y_j son usados como los valores iniciales para el EP ganador toma todo, en las interacciones competitivas. El resultado de estas interacciones es idéntico a la búsqueda de un EP de F_y con el valor más alto, usando la siguiente ecuación :

$$Y_j = \begin{cases} 1 & si \quad Y_j > Y_k, para todo \ j \neq k \\ 0 & en \ otro \ caso \end{cases}$$

Paso 2: Ajustar los valores de conexión de los EPs de F_y, en el aprendizaje competitivo con la dinámica el ganador toma todo, por lo tanto hay solo un conjunto de los pesos en las conexiones que se ajustan que son los pesos en las conexiones del vector de referencia ganador y esta dado por:

$$W_{ij}^{nuevo} = W_{ij}^{antiguo} + \alpha(t)Y_j(a_{ki} - W_{ij})$$

Donde $\alpha(t)$ es un valor diferente de cero, que se decrementa en función del tiempo. Los resultados de esta operación es el movimiento hacia adelante del vector de entrada.

CAPITULO III

RECONOCIMIENTO OPTICO DE CARACTERES

3.1. Introducción

Un OCR (Optical Character Recognition), esta definido como un software que reconoce y procesa una imagen digitalizada que contiene caracteres escritos, almacenándolos en un archivo de texto plausible de ser utilizado por algún procesador de texto.

Los programas de OCR diseñados sobre bases algorítmicas (heurísticos) son los más populares hasta el momento, pero poseen un grado apreciable de inexactitud cuando el texto presenta "ruido", es decir cuando el original contiene manchas como las producidas al fotocopiar una página o símbolos mezclados con el texto un dibujo por ejemplo.

El proceso de convertir una imagen escaneada en texto editable no es una tarea sencilla. Así muchos documentos que fácilmente pueden ser leídos por nuestros ojos, no pueden ser convertidos con precisión en texto editable por un programa OCR ordinario.

La razón por la que un OCR no puede reconocer caracteres tan bien como la gente lo hace radica en que nosotros tenemos la habilidad de mirar subconscientemente las diferentes facetas de que es lo que estamos leyendo y usamos un multinivel de pensamiento para reconocer cada palabra. Es decir que nosotros no solo tomamos en cuenta la forma individual de cada letra, sino que nuestro cerebro tiene la habilidad de juntar patrones y darle el significado correspondiente teniendo en cuenta el contexto.

Entre las características más importantes de los OCRs, se pueden mencionar:

- El reconocimiento óptico de caracteres puede convertir una gran cantidad de información en un corto período de tiempo.
- Puede leer diferentes tipos de letras impresas o mecanografiadas y diferentes clases de caracteres.
- Permite la entrada directa de información sin necesidad de digitar la información.
- Generalmente presentan entornos amigables para los usuarios, por lo cual el personal que los utilice necesita poco entrenamiento.

Una extensión de los OCRs, lo constituyen los ICR (Intelligent Character Recognition) que permiten reconocer caracteres escritos a mano, y los OMR (Optical Mark Recognition) que reconoce marcas impresas.

3.2 Aplicaciones Generales de Reconocimiento de Caracteres Ópticos

En términos generales, el reconocimiento de caracteres ópticos es aplicado en actividades relacionadas con cuentas de cobro/pago y ordenamiento de correspondencia, específicamente en esta última el interés esta en el uso de reconocimiento de caracteres ópticos y su tecnología relacionada para leer y ordenar la correspondencia.

Los bancos usan el reconocimiento de caracteres ópticos para procesar los cheques. En las compañías comerciales, el reconocimiento de caracteres ópticos es usado para registrar las cuentas de las tarjetas de crédito. Así mismo, hay lectores de reconocimiento de caracteres ópticos que son capaces de leer 2.400 cuentas de tarjetas de crédito por minuto.

En el mundo profesional, abogados, médicos, sociólogos, psicólogos, bibliotecarios, etc. pueden usar el reconocimiento de caracteres ópticos para leer revistas, documentos profesionales y crear sus propias bases de datos textuales, también en las oficinas que necesitan aumentar la productividad y eliminar mucha papelería.

En el documento "Technology Assessment Report", los autores indican que un lector de reconocimiento de caracteres ópticos avanzado puede entrar el

equivalente de aproximadamente 300 páginas en una hora mientras que un buen mecanógrafo puede producir cerca de 6 a 10 páginas por hora. También, los más avanzados lectores de reconocimiento de caracteres ópticos han establecido un porcentaje de errores de uno en 300.000 caracteres mientras que un diestro mecanógrafo usualmente tiene un porcentaje de error de aproximadamente uno en 3.000 caracteres (1984, p. 42).

3.3 Sistemas OCR

Actualmente en el mercado internacional existen sistemas OCR e ICR para el reconocimiento de caracteres, entre los que podemos mencionar a los Sistemas Públicos y Sistemas Comerciales:

A) Sistemas Públicos

A.1) NIST : El Instituto Nacional de Estándares y Tecnología (NIST) ha desarrollado un sistema de reconocimiento basado en formas para escritura a mano. NIST ha hecho el sistema disponible al público sin cargo alguno. El sistema, que funciona bajo el sistema operativo Unix, puede usarse sin restricciones, pues fue creado con fondos del gobierno de los Estados Unidos de América.

A.2) XOCR : Fue diseñado para ser usado en máquinas PC bajo el sistema Linux; tiene un pobre desempeño. El sistema usa segmentación directamente "inundando" los símbolos a reconocer, sin reconocer la orientación de la hoja, o si lo que selecciona es texto o no. Resultó ser un sistema poco estable, además de presentar una interfaz que era difícil de manipular. El único formato aceptado por este OCR es el de archivos Bitmap de Windows (BMP).

A.3) OCRCHIE : OCRChie nació como un proyecto de Ingeniería de software en la universidad de Berkele, después se convirtió en el proyecto de Tesis de Kathey Mardsen [1996]. Este software tiene mejores características que XOCR. Puede detectar errores de alineamiento en el texto y corregirlos, segmenta basándose en palabras y admite el popular formato de imágenes TIFF.

A.4) SOCR : SOCR es un esfuerzo por construir librerías estándar de OCR, no obstante, este proyecto tiene mucho por avanzar. Es un proyecto financiado por el departamento de Ciencias de la Computación de la Universidad de Waikato, Nueva Zelandia. El fin de este proyecto es hacer este OCR disponible gratuitamente. El sistema aún no tiene una interfaz integrada para el usuario y aún no reconoce caracteres, la integración de este software está pendiente.

B) Sistemas Comerciales

B.1) OMNIPAGE PRO 12.0 : Es una herramienta de conversión de documentos de papel a formato electrónico, con la capacidad de convertir los archivos de imagen más difundidos a formatos como Microsoft Word, Excel o PDF, así como también permite la integración con las principales herramientas ofimáticas y de gestión de archivos gráficos.

B.2) EYE & HANDS : Es un sistema desarrollado por la compañía Readsoft "Teaching the World's Computers to Read", que permite transferir información automáticamente desde cualquier formato (fax, internet, papel, email), a un sistema informático. Básicamente toda la tecnología de este sistema está englobada en cuatro procesos que son: Interpretación (de texto manuscrito, impreso, código de barras), Verificación (de la interpretación por parte del usuario), Transferencia (para la integración con otros sistemas) y Target .

B.3) FINE READER PROFESSIONAL 6.0 : Es otro sistema OCR desarrollado por ABBYY, el cual es preciso para transformar documentos impresos en ficheros editables, además de ofrecer notables mejoras en cuanto a precisión de texto, reconoce inclusive texto vertical, retención de formato, opciones de almacenamiento, conversión a PDF. También incluye la tecnología de filtrado inteligente de fondo.

B.4) Nf-OCR: Es un producto desarrollado por SANAD, y esta orientado a la automatización de la lectura de campos en documentos de imagen. Está basado en la tecnología de reconocimiento de CAERE y permite el reconocimiento de caracteres omnifont, Caracteres escritos a mano (molde), Marcas, Códigos de barra (en 1Dimensión y PDF en 2 Dimensiones).

3.4 Prototipo de Sistema de Reconocimiento de Caracteres Manuscritos.

3.4.1 Metodología de Desarrollo

Durante el desarrollo de la presente Tesis se ha utilizado una Metodología Orientada a Objetos, la cual es una simplificación del proceso propuesto por Unified Modeling Language "UML", que consta de las siguientes etapas:

1. Presentación de los requerimientos del sistema.
2. Definición del contexto y modos de utilización del sistema.
3. Diseño de la arquitectura del sistema.
4. Modelo conceptual e identificación de objetos del sistema.
5. Desarrollo de los modelos de diseño.
6. Especificación de los algoritmos.

3.4.1.1.Presentación de los Requerimientos del Sistema

Como resultado de esta etapa se muestra a continuación los dos diagramas de actividades correspondientes al conjunto de requerimientos que debe cumplir el prototipo de Sistema desarrollado en esta tesis.

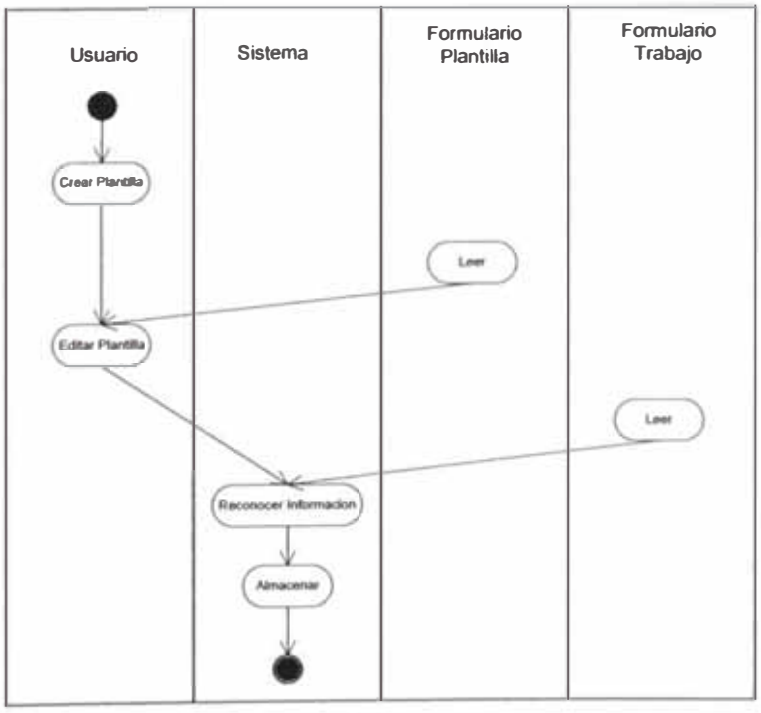


Figura 3.1 Primer Diagrama de Actividades

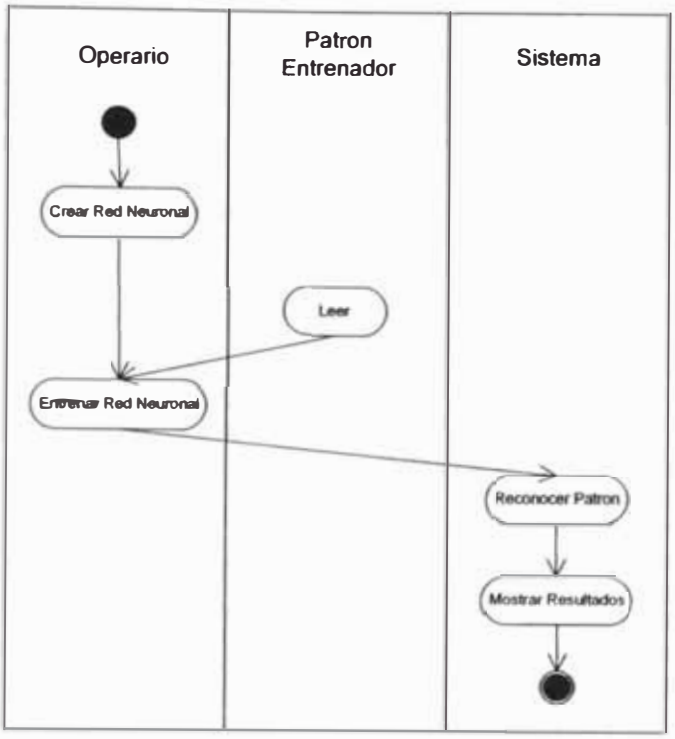


Figura 3.2 Segundo Diagrama de Actividades

Cada diagrama de actividades engloba un proceso diferente, los mismos que se detallan en el siguiente apartado.

A) Listado de las actividades:

El conjunto de actividades que se realizan en cada uno de los dos procesos son las siguientes:

Proceso 1
Crear Plantilla
Editar Plantilla
Reconocer Información
Almacenar

Proceso 2
Crear Red Neuronal
Entrenar Red Neuronal
Reconocer Patrón
Mostrar Resultados

Se han omitido las actividades "Leer Formulario Plantilla", "Leer Formulario Trabajo", "Leer Patrón Entrenador", puesto que, más que actividades son tareas propias del Sistema.

B) Caso de Uso: Primera Iteración

Cada una de las actividades mencionadas en el apartado anterior, esta asociada a un Caso de Uso donde los actores están representados por los usuarios finales del Prototipo (Usuario y Operador), documentos externos (Formulario Plantilla, Formulario Trabajo) y archivos de imágenes; todos ellos en forma conjunta interactúan con el sistema. A su vez cada Caso de Uso describe una secuencia de acciones realizadas por el Prototipo- Fig. 3.3.

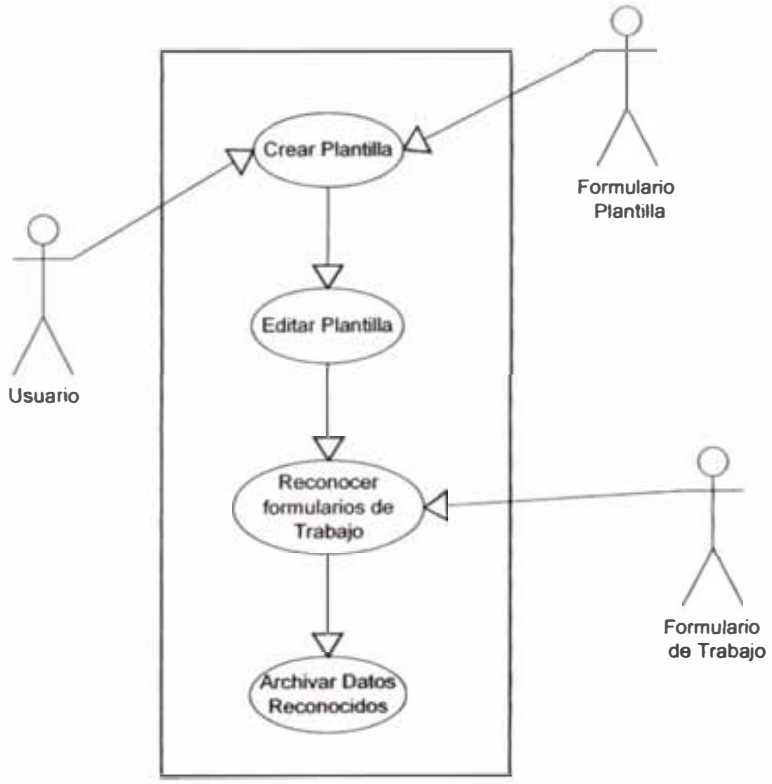


Figura 3.3 Diagrama de Caso de Uso :Primera Iteración – Proceso 1

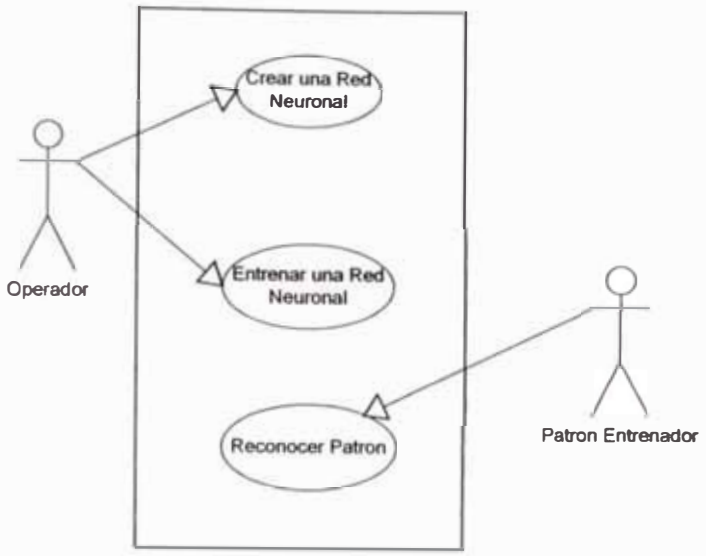


Figura 3.4 Diagrama de Caso de Uso :Primera iteración – Proceso 2

3.4.1.2.Definición del contexto y modos de utilización del sistema

El Diagrama de Caso de Uso anterior resultó muy sencillo, a continuación se presenta una descripción de cada uno de los Casos de Uso, los cuales han permitido realizar un análisis más profundo, así mismo permite comprender las relaciones entre el software que se esta diseñando y su entorno, para ello se hizo uso de la siguiente plantilla (UML no propone ningún modelo de plantilla, por tanto cada cual puede definirse su propia plantilla, siempre y cuando utilice la misma durante todo el proyecto).

Tabla 3.1 Plantilla de Definición de un Caso de Uso de Uso Crear Plantilla

Nombre	Nombre del Caso de Uso
Fecha	Fecha en que se ha elaborado la Plantilla
Descripción	Descripción Informal de los Objetivos de los Casos de Uso.
Actores	Actores que interviene en el Caso de Uso: principales y Secundarios.
Precondiciones	Condiciones que deben cumplirse para que pueda realizarse el Caso de Uso.
Flujo Normal	Secuencia de pasos necesarios para que el Caso de Uso sea desarrollado con éxito.
Flujo Alternativo	Variación en la secuencia de pasos.
Postcondiciones	Condiciones que deben cumplirse al finalizar el Caso de Uso.

Utilizando la plantilla de la figura 3.1, se muestra la documentación correspondiente a cada uno de los Casos de Uso del Primer Proceso.

A) Casos de Uso del Primer Proceso

a.1) Caso de Uso : Crear Plantilla

Tabla 3.2 Plantilla Caso de Uso Crear Plantilla

Nombre	Crear Plantilla.
Fecha	26/10/2003
Descripción: Permite crear una Plantilla, para ello es necesario cargar la imagen de un Formulario Plantilla.	
Actores: Usuario, Formulario Plantilla.	
Precondiciones: La imagen del formulario plantilla debe estar en formato BMP monocromático, además debe ser legible y estar en buenas condiciones.	
Flujo Normal: 1.El actor Usuario selecciona Plantilla / Cargar Formulario. 2.El prototipo de sistema le muestra al actor la ventana estándar Abrir. 3.El actor selecciona la imagen del Formulario Plantilla. 4.El actor visualiza en pantalla la imagen correspondiente	
Flujo Alternativo: 1.El actor no selecciona una imagen valida. 4.El actor no visualiza ninguna imagen en pantalla.	
Postcondiciones: La imagen del Formulario Plantilla no se altera	

a.2) Caso de Uso : Editar Plantilla

Tabla 3.3 Plantilla Caso de Uso Editar Plantilla

Nombre	Editar Plantilla.
Fecha	26/10/2003
Descripción: Le permite al actor usuario preparar una plantilla, para lo cual se enmarcan las casillas en el Formulario Plantilla que se han de reconocer.	
Actores: Usuario, Formulario Plantilla.	
Precondiciones: El actor usuario debe cargar una imagen valida para el formulario plantilla	
Flujo Normal: 1.El actor usuario elige la opción crear Plantilla. 2.Luego el prototipo le muestra el cuadro Abrir, donde debe elegir la imagen del Formulario Plantilla. 3.El prototipo activa la Barra de herramientas editar.	
Flujo Alternativo:	
Postcondiciones: La imagen del Formulario Plantilla no sufre alteraciones.	

El Caso de Uso Editar Plantilla incluye a su vez cuatro tareas esenciales:

- Enmarcar casillas, sucede cuando el usuario se dispone a identificar dentro del Formulario Plantilla las casillas que desea que el prototipo reconozca, a esta tarea se le denomina Colocar Bloque.

- Modificar posición, el usuario debe tener la Posibilidad de arrastrar un bloque y colocarlo alrededor de otra casilla, es decir cambiar la posición de un bloque, a esta tarea se la denomina Modificar Posición.

- Eliminar bloque, también es necesario contar con alguna opción donde el prototipo omita el reconocimiento de alguna casilla del Formulario Plantilla, a esta tarea se le denomina Eliminar Bloque.

- Ajustar el tamaño de un bloque, por ultimo el usuario debe tener la posibilidad de ajustar el tamaño del Bloque, de tal manera que obtenga una zona de reconocimiento más exacta. A esta tarea se le denomina Modificar Tamaño.

Teniendo en cuenta lo anterior, aparecen cuatro nuevos casos de uso:

a.2.1) Caso de Uso: Colocar Bloque

Tabla 3.4 Plantilla Caso de Uso Colocar Bloque

Nombre	Colocar Bloque.
Fecha	23/10/2003
Descripción: Permite enmarcar dentro de un recuadro la casilla del formulario plantilla que va a ser reconocida por el prototipo.	
Actores: Usuario, Formulario Plantilla.	
Precondiciones: El actor Usuario debe cargar la imagen del formulario plantilla.	
Flujo Normal: 1.El actor usuario selecciona la herramienta dibujar bloque. 2.Dirige el mouse hacia la casilla del formulario plantilla que desea enmarcar. 3.El actor usuario hace clic izquierdo sobre la coordenada superior izquierda y en seguida arrastra el mouse hacia la coordenada inferior derecha (o viceversa). 4.El actor usuario visualiza en pantalla un recuadro de color rojo alrededor de la zona enmarcada. 5.El prototipo le pide al actor ingresar el nombre de identificación para dicho bloque, el tipo de dato que contiene y el tipo de casilla.	
Flujo Alternativo:	
Postcondiciones: El prototipo almacena información sobre la ubicación del bloque dentro del formulario plantilla, además del nombre, tipo de dato que contiene y tipo de casilla.	

a.2.2) Caso de uso: Modificar posición del Bloque

Tabla 3.5 Plantilla Caso de Uso Modificar Posición Bloque

Nombre	Modificar posición Bloque
Fecha	23/10/2003
Descripción: Permite trasladar un bloque de una a otra posición.	
Actores: Usuario, Formulario Plantilla.	
Precondiciones: Existe al menos un bloque que enmarca a una casilla en el Formulario Plantilla.	
Flujo Normal: 1.El actor usuario selecciona la herramienta Seleccionar. 2.Ubica el mouse en el interior del bloque que desea mover. 3.Lo arrastra con el mouse hasta conseguir la posición que desea para este bloque. 4.El actor usuario suelta el mouse. 5.El bloque se ubica en su nueva posición.	
Flujo Alternativo: 1.El actor no ubica el mouse en el interior del bloque. 4.No se altera ninguna posición.	
Poscondiciones: El número de bloques en el Formulario Plantilla no varía.	

a.2.3) Caso de Uso : Eliminar Bloque

Tabla 3.6 Plantilla Caso de Uso Eliminar Bloque

Nombre	Eliminar Bloque
Fecha	23/01/2004
Descripción: Permite eliminar un bloque.	
Actores: Usuario, Formulario Plantilla	
Precondiciones: El Formulario Plantilla contiene al menos un bloque.	
Flujo Normal: 1.El actor usuario selecciona la herramienta eliminar. 2.Ubica el mouse en el interior del bloque que desea eliminar. 3.Hace clic. 4.El bloque desaparece del Formulario Plantilla.	
Flujo Alternativo: 2.El actor no ubica el mouse en el interior de ningún bloque. 4.Ningún bloque desaparece.	
Postcondiciones: El número de bloques dentro del Formulario Plantilla decrementa en uno.	

a.2.4) Caso de Uso : Modificar tamaño de un bloque

Tabla 3.7 Plantilla Caso de Uso Modificar tamaño de Bloque

Nombre	Modificar tamaño de un Bloque
Fecha	23/01/2004
Descripción: Permite modificar las coordenadas de un bloque.	
Actores: Usuario, Formulario Plantilla.	
Precondiciones: Existe al menos un bloque en el Formulario Plantilla.	
Flujo Normal: 1.El actor usuario selecciona la herramienta Seleccionar. 2.El actor usuario ubica el mouse sobre cualquiera de los bordes del bloque que desea modificar. 3.Hace clic y arrastra el mouse hasta conseguir el tamaño deseado. 4.El actor suelta el mouse y el bloque cambia de tamaño.	
Flujo Alternativo: 1.El actor no ubica el mouse sobre alguno de los bordes del bloque. 4.No se altera el tamaño del bloque.	
Postcondiciones: El número de bloques dentro del Formulario Plantilla no varía.	

Hasta aquí los Casos de Uso descritos, permiten al usuario Editar un Formulario Plantilla, pero también es muy importante almacenar toda esta información, en un archivo de texto, el cual toma la denominación de "Plantilla", de tal manera que pueda ser utilizada y/o modificada en lo posterior. Por lo tanto surge un nuevo Caso de Uso Guardar Plantilla.

a.2.5) Caso de Uso : Guardar Plantilla

Tabla 3.8 Plantilla Caso de Uso Guardar Plantilla

Nombre	Guardar Plantilla
Fecha	23/10/03
Descripción: Permite el almacenamiento físico de una Plantilla en el disco.	
Actores: Usuario	
Precondiciones: El usuario debe haber editado el Formulario Plantilla.	
Flujo Normal: 1. El usuario elige el Menú Plantilla / Guardar. 2. El usuario proporciona al prototipo la ubicación donde almacenar la Plantilla. 3. El usuario ingresa un nombre para la Plantilla y hace clic en aceptar. 4. El prototipo crea un archivo de texto de salida.	
Flujo Alternativo: 3. El usuario hace clic en el botón cancelar. 4. El prototipo no crea ningún archivo de texto de salida.	
Postcondiciones: El prototipo de Sistema crea un archivo de texto que contiene el nombre de la imagen del Formulario Plantilla, la posición, el nombre, el tipo de dato y tipo de casillas que enmarca a los campos del Formulario Plantilla a reconocer.	

a.3) Caso de Uso : Reconocer Caracteres

Tabla 3.9 Plantilla Caso de Uso Reconocer Caracteres

Nombre	Reconocer Caracteres
Fecha	26/11/2003
Descripción: Permite que el prototipo realice el reconocimiento de un carácter dado. ya sea letra, digito o símbolo.	
Actores: Formulario de Trabajo	
Precondiciones: La imagen del carácter no debe contener efectos no deseados que la distorsionen, ni puntos aislados (ruido).	
Flujo Normal: 1.El prototipo toma la zona de la imagen del carácter a reconocer. 2.El prototipo encarga a la red neuronal (RN) la tarea del reconocimiento. 3.La RN determina si la imagen del carácter corresponde a uno de los patrones entrenados.	
Flujo Alternativo: 3.En caso contrario mostrara un mensaje indicando que la imagen es difusa y si se desea entrenar o agregar dicho imagen como un nuevo patrón	
Poscondiciones: La imagen del carácter es reconocida por la RN.	

El Caso de Uso anterior es complejo y a su vez constituye el corazón del Prototipo de Sistema, ya que debe realizar el Reconocimiento de los datos contenidos en los Formularios de Trabajo, por ello se le ha dividido en 5 subtareas: Extraer campos

del Formulario de Trabajo, Segmentar Campos, Filtrar Ruido, Preprocesar, Componer los caracteres reconocidos y Almacenar la información reconocida.

a.3.1) Caso de Uso : Extraer Campos

Cuadro 3.10 Plantilla Caso de Uso Extraer Campos

Nombre	Extraer campos.
Fecha	26/11/2003
Descripción: Permite recortar solo la zona de la imagen correspondiente a cada campo a reconocer del Formulario de Trabajo	
Actores: Formulario de Trabajo.	
Precondiciones: Debe existir al menos un bloque en la Plantilla.	
Flujo Normal: 1.El prototipo identifica las coordenadas de cada bloque de la plantilla en los formularios de trabajo. 2.El prototipo recorta esta zona para su posterior reconocimiento.	
Flujo Alternativo:	
Postcondiciones: La imagen de los formularios de trabajo y de la Plantilla no se altera.	

a.3.2) Caso de Uso : Segmentar Campos

Tabla 3.11 Plantilla Caso de Uso Segmentar Campos

Nombre	Segmentar campos
Fecha	26/11/2003
Descripción: Permite dividir la imagen de un campo en pequeñas zonas correspondientes a cada carácter, para ello se elimina los bordes y líneas de división que pudieran existir en la imagen del campo.	
Actores:	
Precondiciones: La imagen del campo debe ser legible.	
Flujo Normal: 1.El prototipo toma la imagen del campo a reconocer. 2.El prototipo elimina los bordes y líneas divisorias si las hubiera de la imagen, aplicando técnicas apropiadas.	
Flujo Alternativo:	
Postcondiciones: La imagen del campo no presenta bordes ni líneas.	

a.3.3) Caso de Uso : Filtrar Ruido

Tabla 3.12 Plantilla Caso de Uso Filtrar Ruido

Nombre	Filtrar ruido
Fecha	26/11/2003
Descripción: El prototipo se encarga de eliminar aquellos puntos o pequeños trazos que no forman parte de la imagen de un carácter.	
Actores:	
Precondiciones: La imagen corresponde a un solo carácter . La imagen no contiene bordes ni líneas divisorias.	
Flujo Normal: 1.El prototipo toma la imagen del carácter. 2.Verifica punto por punto y elimina aquellos puntos que se encuentran aislados y que no son parte de la imagen del carácter.	
Flujo Alternativo:	
Postcondiciones: La imagen del carácter no presentan puntos aislados y esta lista para ser reconocida por la Red Neuronal.	

a.3.4) Caso de Uso : Componer Caracteres Reconocidos

Tabla 3.13 Plantilla Caso de Uso Composición de Caracteres

Nombre	Componer Caracteres Reconocidos
Fecha	26/11/2003
Descripción: Permite componer los caracteres reconocidos en cadenas de letras o dígitos formando palabras (letras) o cantidades (números).	
Actores:	
Precondiciones: Cada carácter debe haber pasado por la fase de reconocimiento por la red neuronal.	
Flujo Normal: 1.El prototipo toma uno a uno los caracteres reconocidos y los une para formar palabras si fueran letras o cantidades si fueran dígitos. 2.Si estos fuesen letras ubica un espacio en blanco que indica el fin de una palabra y el comienzo de la siguiente.	
Flujo Alternativo:	
Postcondiciones: Las cadenas de caracteres tienen significado.	

E

El siguiente Caso de Uso cumple la tarea de almacenar la información reconocida por el Prototipo.

a.3.5) Caso de Uso : Almacenar la información reconocida

Tabla 3.14 Plantilla Caso de Uso Almacenar información reconocida

Nombre	Almacenar la información reconocida
Fecha	28/12/2003
Descripción: Esta es la etapa final del Prototipo y permite que las cadenas de caracteres reconocidas sean almacenadas en una tabla (Base de Datos).	
Actores:	
Precondiciones: La tabla está vacía.	
Flujo Normal: 1.El prototipo crea una tabla cuyos campos son definidos de acuerdo a los nombres y tipos de cada campo del Formulario Plantilla especificados al momento de dibujar los bloques que enmarcan a los correspondientes campo. 2.El prototipo almacena las cadenas de caracteres reconocidas en la tabla.	
Flujo Alternativo:	
Postcondiciones: La tabla contiene la información reconocida de los campos de los formularios de Trabajo.	

B) Casos de Uso del Segundo Proceso

b.1) Caso de Uso : Entrenar la Red Neuronal

Tabla 3.15 Plantilla Caso de Uso Entrenar la Red Neuronal

Nombre	Entrenar la Red Neuronal
Fecha	06/12/2003
Descripción: En esta etapa el prototipo permite entrenar a la Red Neuronal con patrones que representan letras o dígitos que son entregados como entradas a la Red Neuronal. El Entrenamiento puede ser supervisado y No supervisado.	
Actores: Operador	
Precondiciones: Las imágenes de entrada correspondientes a los patrones entrenadores deben estar en formato BMP monocromatico:	
Flujo Normal: 1. la RN mediante el empleo del algoritmo de entrenamiento supervisado o no supervisado entrenará a un patrón especifico utilizando el patron entrenador correspondiente.	
Flujo Alternativo:	
Postcondiciones: La imagen correspondiente a los patrones entrenadores no se alteran. Los patrones entrenados se alteran de acuerdo a las variaciones que pueda provocar el patrón entrenador.	

Tal como se indica en la descripción de este proceso el entrenamiento de la Red Neuronal puede ser Supervisado y No Supervisado. En consecuencia surgen dos nuevos Casos de Uso.

b.2) Caso de Uso : Entrenamiento Supervisado

Tabla 3.16 Plantilla Caso de Uso Entrenamiento Supervisado

Nombre	Entrenamiento Supervisado.
Fecha	06/12/2003
Descripción: El actor operario es el encargado de entrenar a la Red Neuronal utilizando un patrón entrenador, es decir es el que asume las tareas de añadir un nuevo patrón, eliminar un patrón o entrenar un patrón especifico.	
Actores: Operario	
Precondiciones: La Red Neuronal debe contener al menos un patrón entrenado y otro patrón entrenador.	
Flujo Normal: 1.En el caso de eliminar, añadir o entrenar un patrón entrenado, es el actor quien lo elige.	
Flujo Alternativo:	
Poscondiciones: El número de patrones entrenados se altera según sea el caso de eliminar entonces disminuye en uno y se incrementará en uno si el operario eligió añadir o permanecerá igual si eligió entrenar.	

b.3) Caso de Uso : Entrenamiento no Supervisado

Tabla 3.17 Plantilla Caso de Uso Entrenamiento No Supervisado

Nombre	Entrenamiento no supervisado
Fecha	06/12/2003
Descripción: En esta fase es la misma Red Neuronal la que sigue un auto entrenamiento, es decir se encarga de tomar las decisiones de añadir o entrenar un patrón específico, teniendo en cuenta el porcentaje de aceptación, el cual debe ser definido previamente.	
Actores:	
Precondiciones: Definir con anterioridad el Porcentaje de Aceptación.	
Flujo Normal: 1.Se ingresa un patrón entrenador, si la red Neuronal reconoce una similitud igual o mayor al porcentaje de aceptación con un patrón entrenado, entonces esta decidirá entrenar a dicho patrón. 2.En caso contrario decidirá añadir un nuevo patrón a la Red Neuronal.	
Flujo Alternativo:	
Postcondiciones: El porcentaje de Aceptación no se altera.	

Una vez que se ha detallado los Casos de Uso y sus relaciones entre ellos, ha sido posible construir el siguiente diagrama de Casos de Uso:

3.4.1.3 Diseño Arquitectónico

La información emergente de las interacciones entre el Prototipo de Sistema y su entorno, ha sido utilizada como base para diseñar la Arquitectura del Prototipo de Sistema.

Por otro lado, debido a la Complejidad del Prototipo de Sistema de Reconocimiento de Caracteres Manuscritos Aislados en Campos de Formularios utilizando Redes Neuronales, antes de definir la Arquitectura del Sistema, se ha diseñado una posible Arquitectura representada como un Modelo de Capas, en la cual se ha especificado los principales componentes del Sistema así como su respectiva descripción.

Además como resultado del estudio de los procesos anteriores se ha identificado dos Subsistemas: Un Subsistema de Reconocimiento propiamente dicho, al que se denominará **"RECAM"**, y un Subsistema de Red Neuronal al que denominará **"NEURON"**.

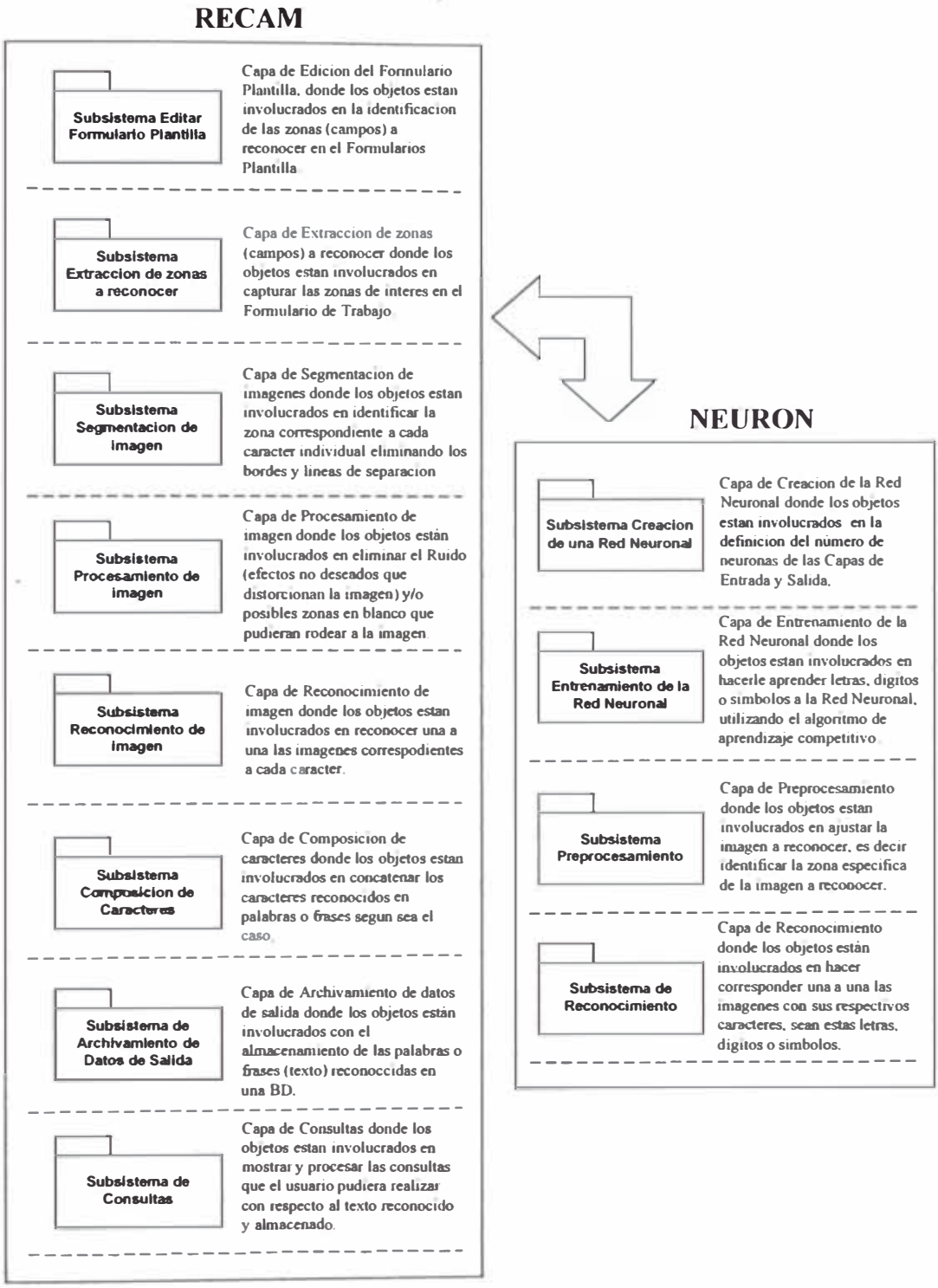


Figura 3.6 Arquitectura del Modelo de Capas

Esta Arquitectura basada en capas ha constituido la base para definir la Arquitectura propiamente dicha del Sistema, puesto que algunos de los componentes en realidad vienen a formar parte de otros o existe una dependencia entre ellos:

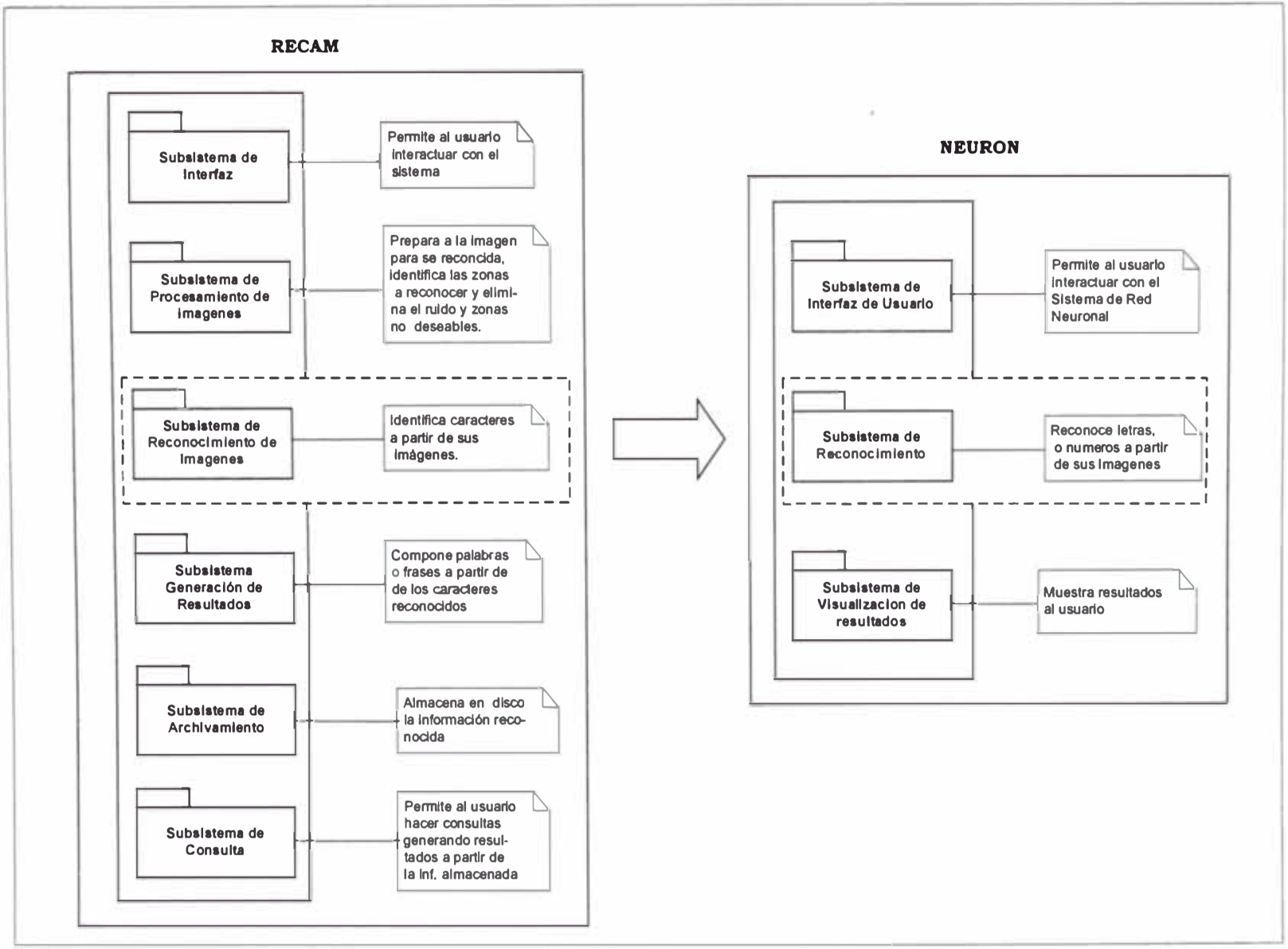


Figura 3.7 Arquitectura del Sistema

3.4.1.4 Modelo conceptual e identificación de Objetos del Sistema.

En esta etapa se ha construido el modelo conceptual a partir de los objetos, que son instancias que poseen características propias, los cuales han surgido de la descripción de los Casos de Uso.

Se ha utilizado la técnica de identificación de nombres, y además al tener en cuenta que se necesita al menos un objeto en cada uno de los niveles de la Arquitectura del Sistema, se ha encontrado los siguientes objetos:

> Formulario Plantilla, Bloques, Formulario de Trabajo, Plantilla, Campos, Red Neuronal, Patrón Entrenador, Patrón Entrenado, Datos reconocidos.

Según las especificaciones anteriores la Red Neuronal puede reconocer Letras y Números, lo cual indica la aparición de dos nuevas categorías: Red Neuronal de Números y Red Neuronal de Letras. Finalmente nos quedan los siguientes objetos:

> Formulario Plantilla, Plantilla Bloques, Formulario de Trabajo, Campos, Plantilla, Red Neuronal de Letras, Red Neuronal de Números, Patrón Entrenador, Patrón Entrenado, Datos reconocidos.

Ahora una vez que se ha identificado los objetos, es posible hallar las *asociaciones* entre ellos. A este nivel de análisis no es tan importante el encontrar todas las asociaciones; por lo tanto se identifico sólo las que se "necesitan conocer", las cuales se listan a continuación.

- Plantilla - contiene – Bloques
- Formulario de Trabajo – contiene – campos.
- Red Neuronal – es una – Red Neuronal Números.
- Red Neuronal – es una – Red Neuronal Letras.
- Red Neuronal, esta compuesto por – Patrón Entrenado.
- El operador – utiliza - Patrón Entrenador – para entrenar - Red Neuronal.
- Operador – crea - Red Neuronal.

- Operador – entrena - Red Neuronal.
- Usuario – edita – Plantilla
- Red Neuronal – reconoce – Campos – de un - Formulario de Trabajo – genera – Datos de Salida.
- Patrón Entrenado – le corresponde – Patrón Entrenador.

Ahora bien, cada uno de estos objetos tiene sus propios atributos, por lo tanto se presenta a continuación la lista de objetos identificados con sus respectivos principales atributos.

- Formulario Plantilla : nombre, ubicación, formato, ancho, alto
- Plantilla : nombre, formato, ancho, alto, imagen.
- Bloque : nombre, tipo, coordenadas.
- Formulario de Trabajo : nombre, formato, ancho, alto, imagen.
- Campos: nombre, ancho, alto, imagen
- Red Neuronal: nombre.
- Patrón Entrenador: nombre, formato, alto, ancho.
- Patrón Entrenado: nombre, elemento de procesamiento.
- Datos reconocidos: nombre, tipo, valor.

El resto de atributos irán apareciendo durante las siguientes fases del desarrollo.

A su vez, para algunos de estos objetos se ha identificado también sus respectivos métodos, (un método es un operación que permite manipular a un objeto por medio de sus atributos).

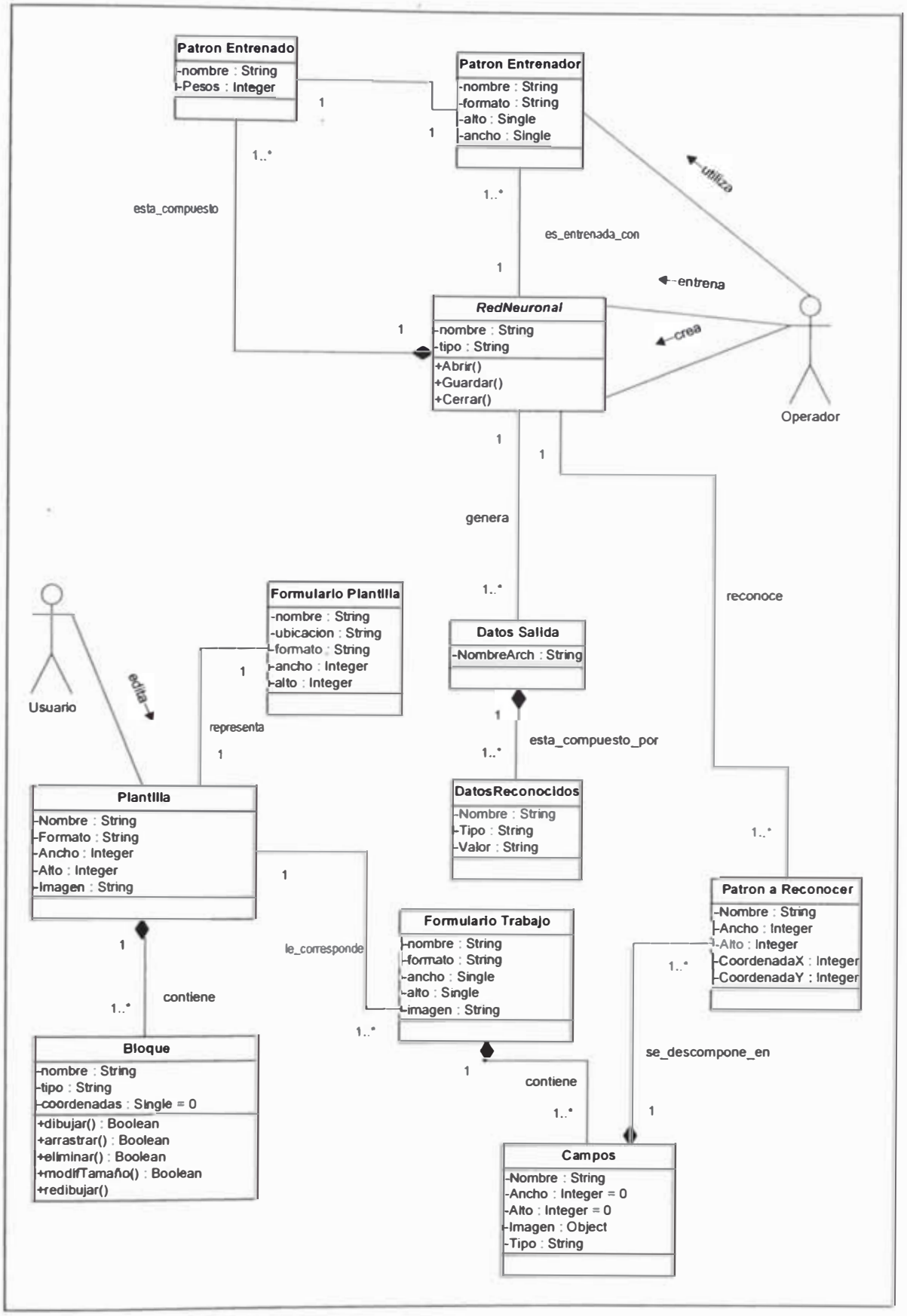


Figura 3.8 Modelo Conceptual

3.4.1.5 Desarrollo de los Modelos del Sistema

Existen dos Modelos de Diseño para describir un diseño orientado a Objetos:

- Modelos Estáticos : Son aquellos que describen la estructura estática del Sistema, es decir describen al sistema en términos de las clases de Objetos y sus relaciones.

  Entre las relaciones más importantes que se documentan en este modelo se tiene : de generalización, utiliza/utilizado por, y de composición.

- Modelos Dinámicos : Describen la estructura dinámica del Sistema, dando énfasis a las interacciones entre los objetos que conforman el Sistema

En esta etapa de desarrollo se ha aplicado ambos modelos, específicamente en el primer caso se utilizo el Modelos de los Subsistemas y en el segundo caso se utilizó el Modelo de Secuencias del UML.

A) Modelo Estático

Como se percibe durante todo el Proceso de Modelamiento, nuestro Sistema consta de dos Subsistemas principales, a los cuales se ha denominado, Subsistema de Reconocimiento **"RECAM"** y Subsistema de Red Neuronal **"NEURON"**.

El siguiente esquema muestra gráficamente el Modelo Estático correspondiente a ambos subsistemas:

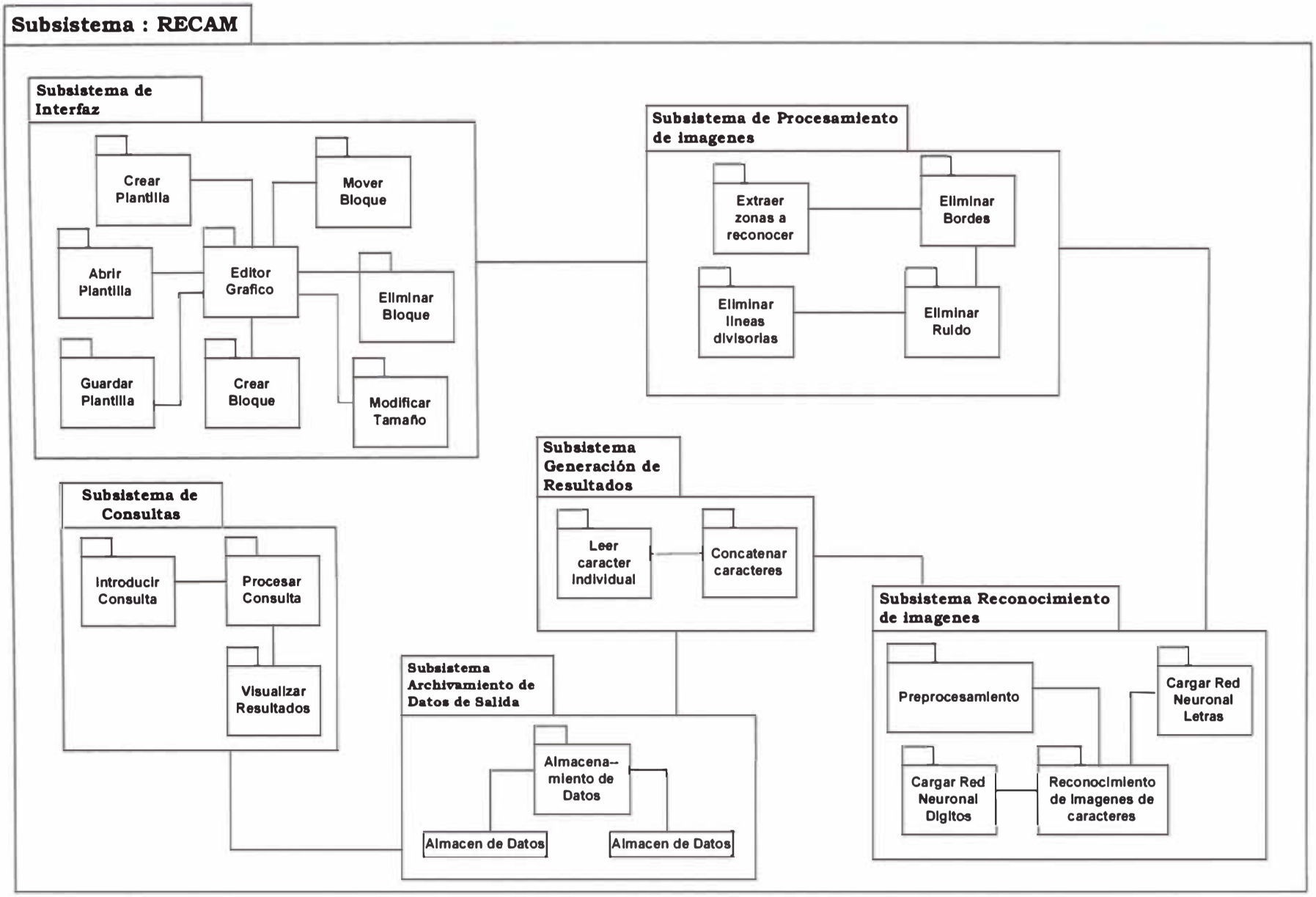


Figura 3.9 Modelo Estático del Subsistema RECAM

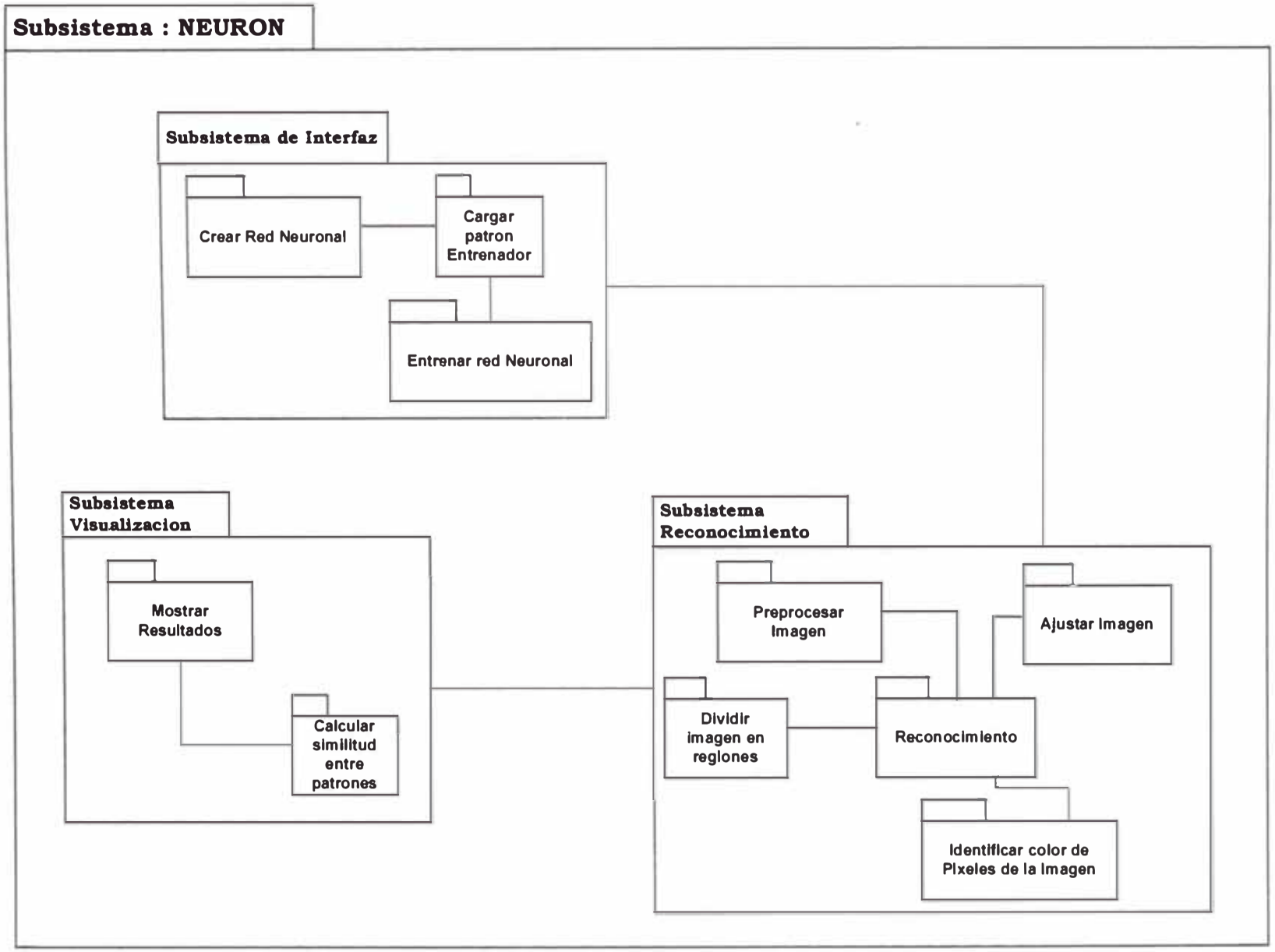


Figura 3.10 Modelo Estático del Subsistema NEURON

B) Modelo Dinámico del Sistema

Mediante este modelo se ha representado tanto las interacciones entre los diferentes objetos del sistema, así como los eventos que envía el actor al Sistema. Para ello UML propone el uso de los diagramas de secuencia siendo necesario la elaboración los Contratos, los que han permitido establecer las responsabilidades de cada operación.

A continuación se presenta la descripción del contenido de un Contrato.

Tabla 3.18 Plantilla de Contratos

Nombre	Signatura de la operación
Responsabilidades	Responsabilidades de la operación
Tipo	Clase que contiene la operación
Notas	Notas, comentarios y/o aclaraciones
Excepciones	Excepciones que pudiera presentar una operación.
Salida	Salida que no sea por pantalla
Precondiciones	Condiciones que han de cumplirse antes de ejecutar la operación.
Postcondiciones	Estado del sistema después de la ejecución del método.
Manejo de Errores	Detalle de errores que pudieran ocurrir.

a) Contrato Crear Plantilla

- **Nombre**: Crear Plantilla
- **Responsabilidades**: Crea una nueva plantilla
- **Tipo**: Interfaz Grafico.
- **Precondiciones**: Debe existir la imagen de un Formulario Plantilla.
- **Postcondiciones**: El objeto Plantilla es creado y esta listo para ser editado.
- **Manejo de Errores :** Sino existe imagen de un Formulario Plantilla, se cancela la operación.

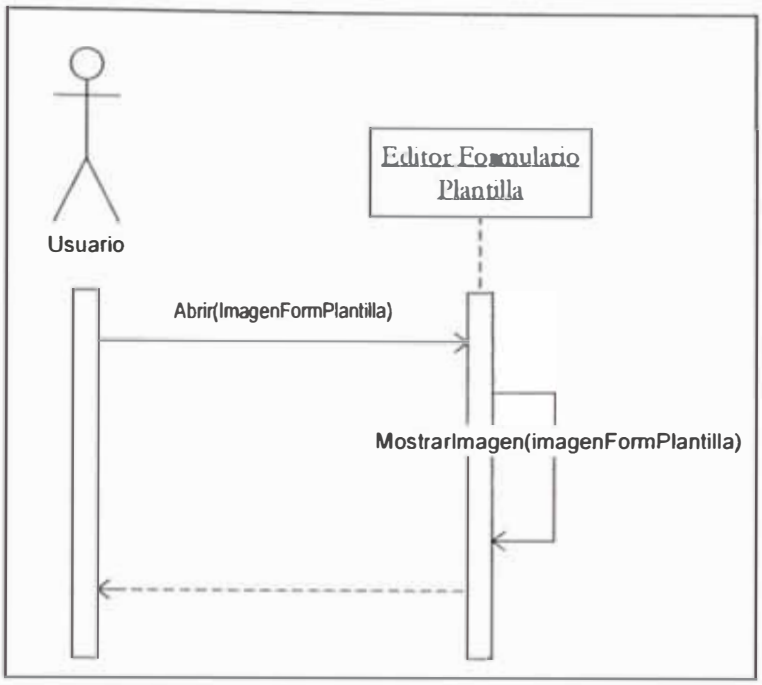


Figura 3.11 Diagrama de Secuencia "Crear Plantilla"

b) Contrato Abrir Plantilla

- **Nombre**: Abrir Plantilla
- **Responsabilidades**: Permite cargar y visualizar en pantalla una plantilla ya creada.
- **Tipo**: Interfaz Grafico.
- **Precondiciones**: Debe existir un archivo de extensión "pla", correspondiente a la plantilla que se desea abrir.
- **Postcondiciones**: Los objetos bloque que enmarcan a las zonas a reconocer se visualizan en pantalla.
- **Manejo de Errores :** Sino existe un Archivo Plantilla, se cancela la operación y no se creará ningún objeto Bloque.

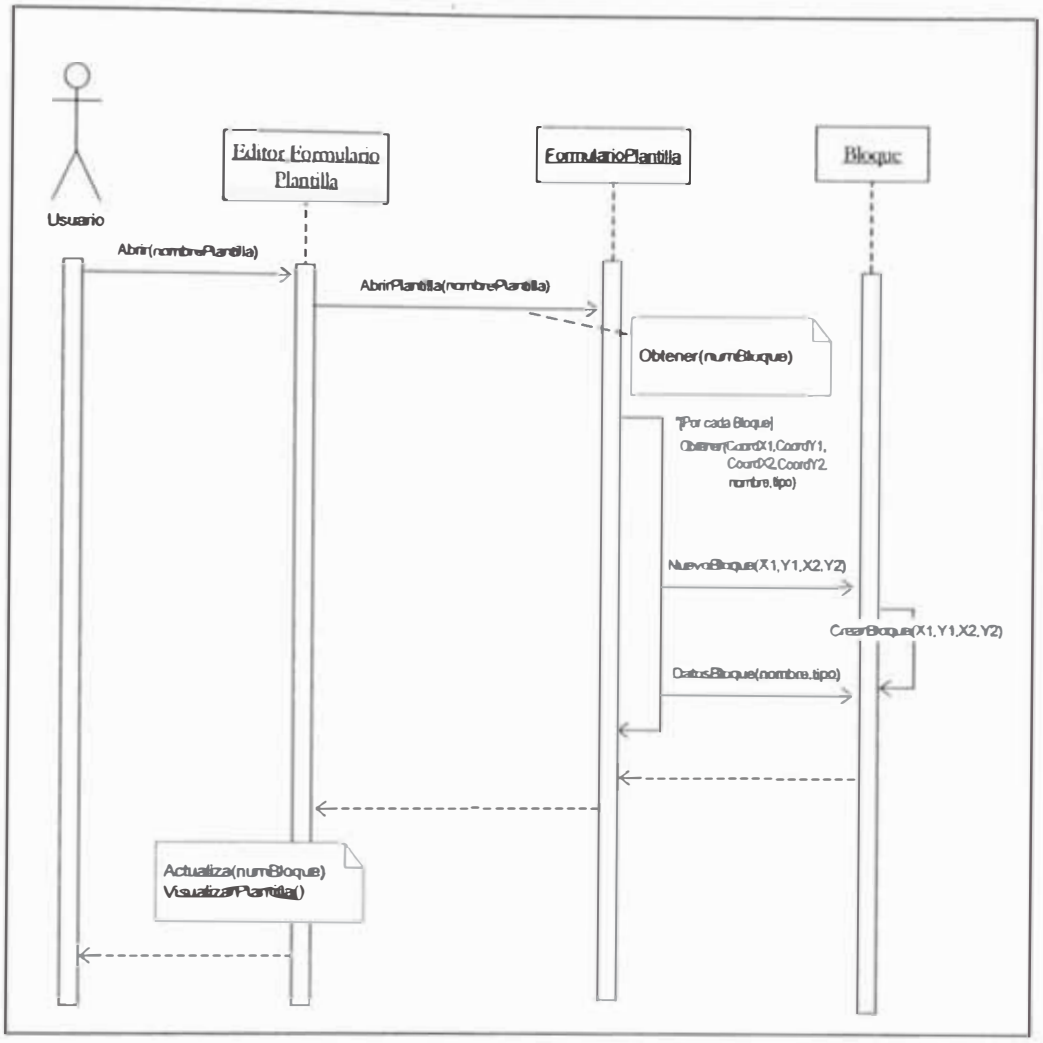


Figura 3.12 Diagrama de Secuencia "Abrir Plantilla"

c) Contrato Guardar Plantilla

- **Nombre**: Guardar Plantilla
- **Responsabilidades**: Permite almacenar fisicamente una plantilla ya creada.
- **Tipo**: Interfaz Grafico.
- **Precondiciones**: Un nombre de archivo para la plantilla a guardar ha sido elegido por el usuario.
- **Manejo de Errores :** Si el nombre de archivo ya existe, mostrar al usuario un mensaje en pantalla.

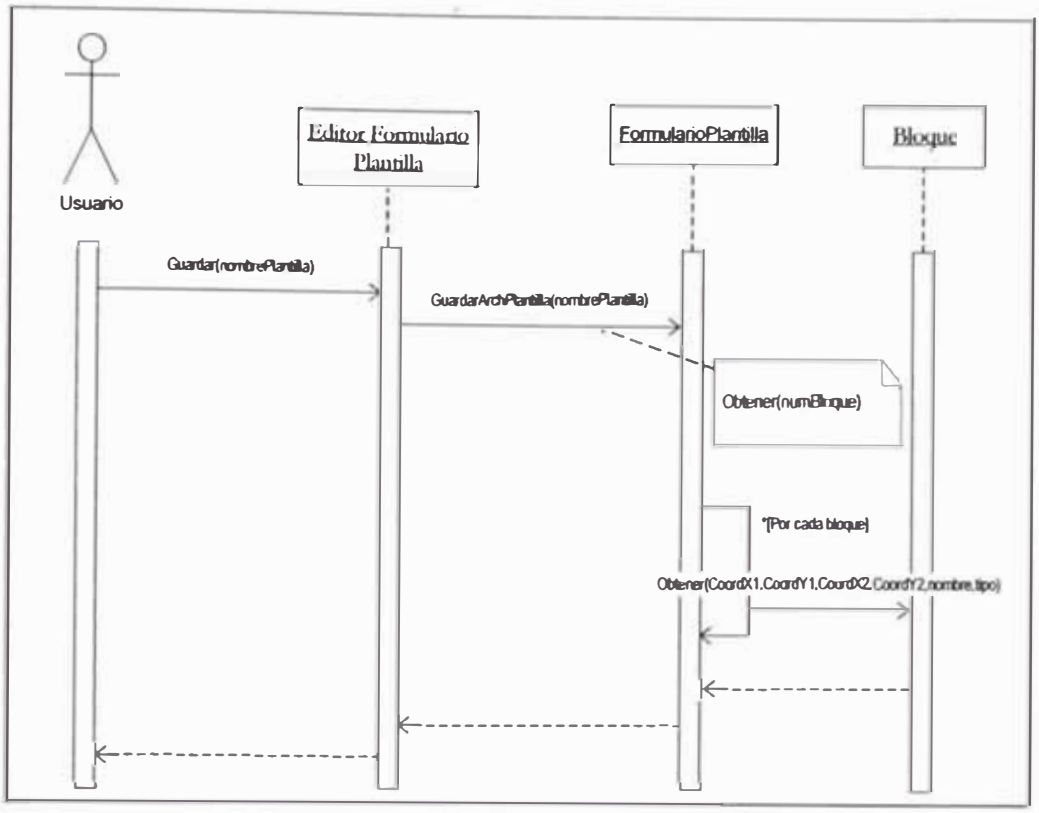


Figura 3.13 Diagrama de Secuencia "Guardar Plantilla"

d) Contrato Crear Bloque

- **Nombre**: Crear Bloque.
- **Responsabilidades**: Permite al usuario crear un nuevo bloque para enmarcar un campo del formulario que se desea reconocer.
- **Tipo**: Interfaz Grafico.
- **Precondiciones**: El usuario debe proporcionar tanto el nombre, como el tipo de dato que contiene el bloque, tipo de rejilla y la posición para cada bloque.
- **Postcondiciones**: Un objeto Bloque ha sido creado.
- **Manejo de Errores :** Si el usuario no ingresa el nombre del bloque el sistema colocara por defecto el nombre "noname", el tipo de dato por defecto es "numérico", y el tipo de rejilla por defecto es "líneas divisorias".

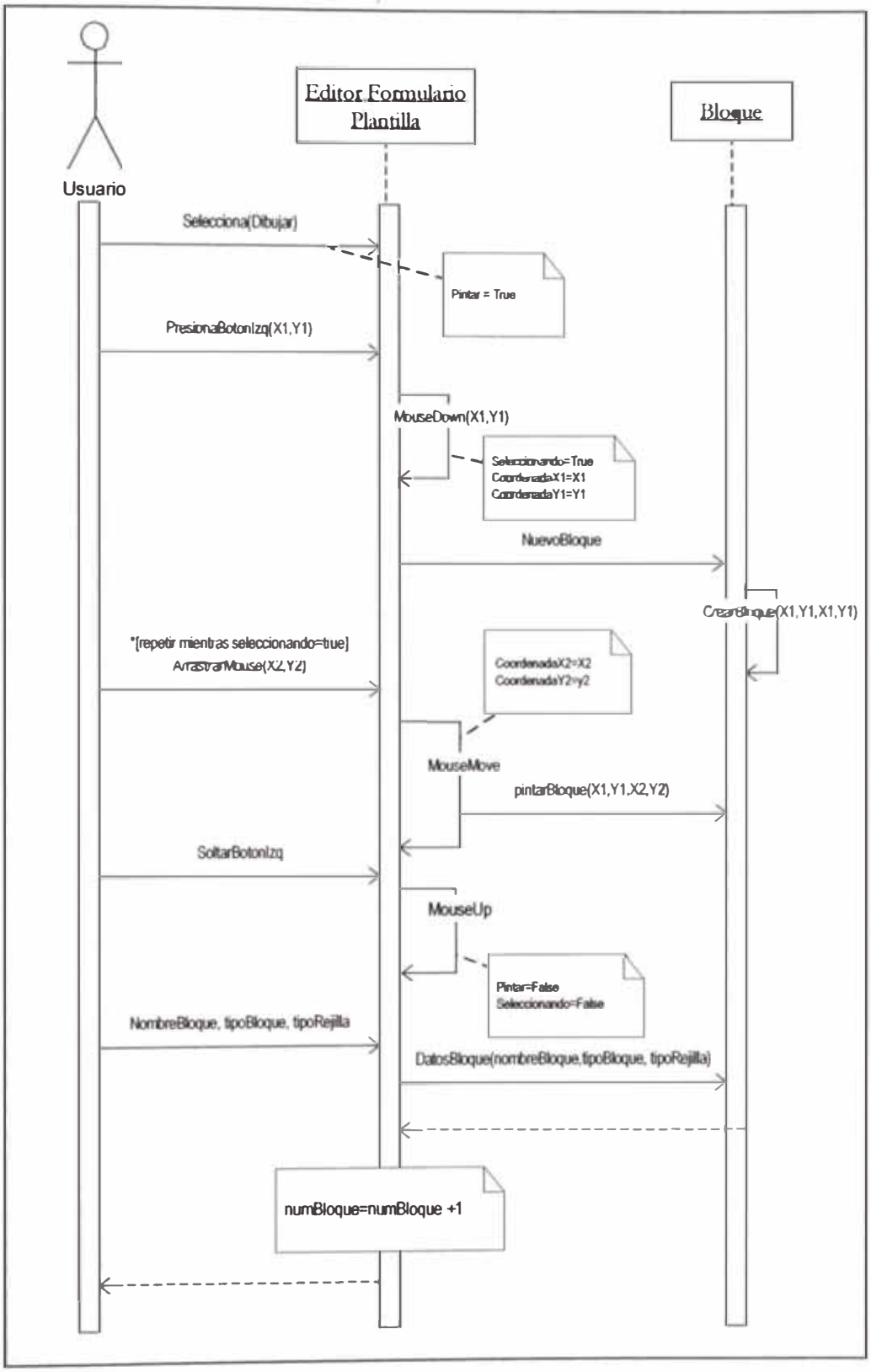


Figura 3.14 Diagrama de Secuencia "Crear Bloque"

e) Contrato Modificar Tamaño de Bloque

- **Nombre**: Modificar Bloque.

- **Responsabilidades**: Permite al usuario redimensionar el tamaño de un bloque ya creado.
- **Tipo**: Interfaz Grafico.
- **Precondiciones**: El objeto bloque debe existir.
- **Postcondiciones**: Una o mas de las coordenadas del objeto Bloque varían.

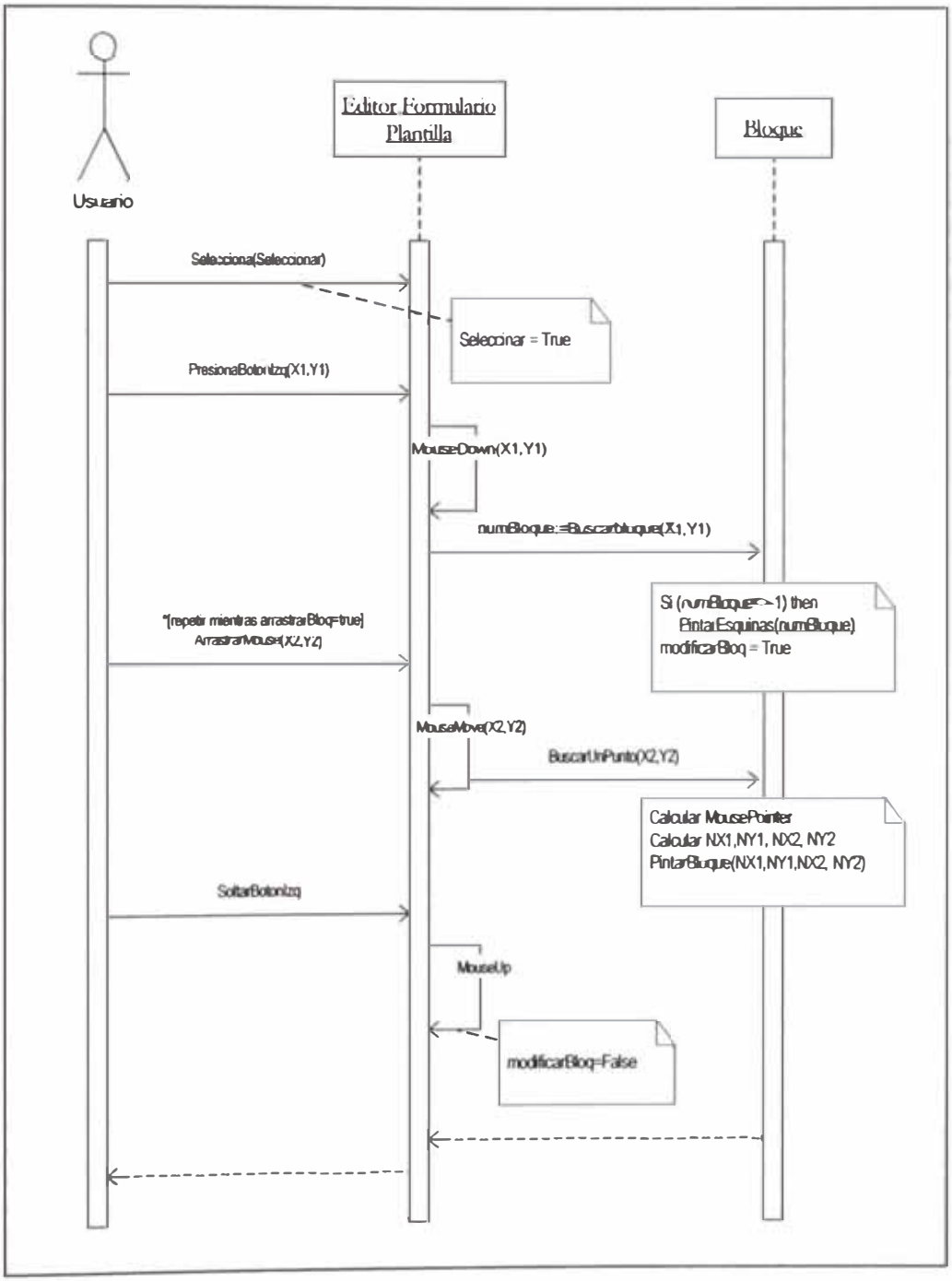


Figura 3.15 Diagrama de secuencias "Modificar Tamaño de Bloque"

f) Contrato Eliminar Bloque

- **Nombre**: Eliminar Bloque.
- **Responsabilidades**: Permite al usuario eliminar un bloque ya creado.
- **Tipo**: Interfaz Grafico.
- **Precondiciones**: El usuario debe seleccionar el bloque a eliminar, por lo tanto el objeto bloque debe existir.
- **Postcondiciones**: Un objeto bloque es eliminado.

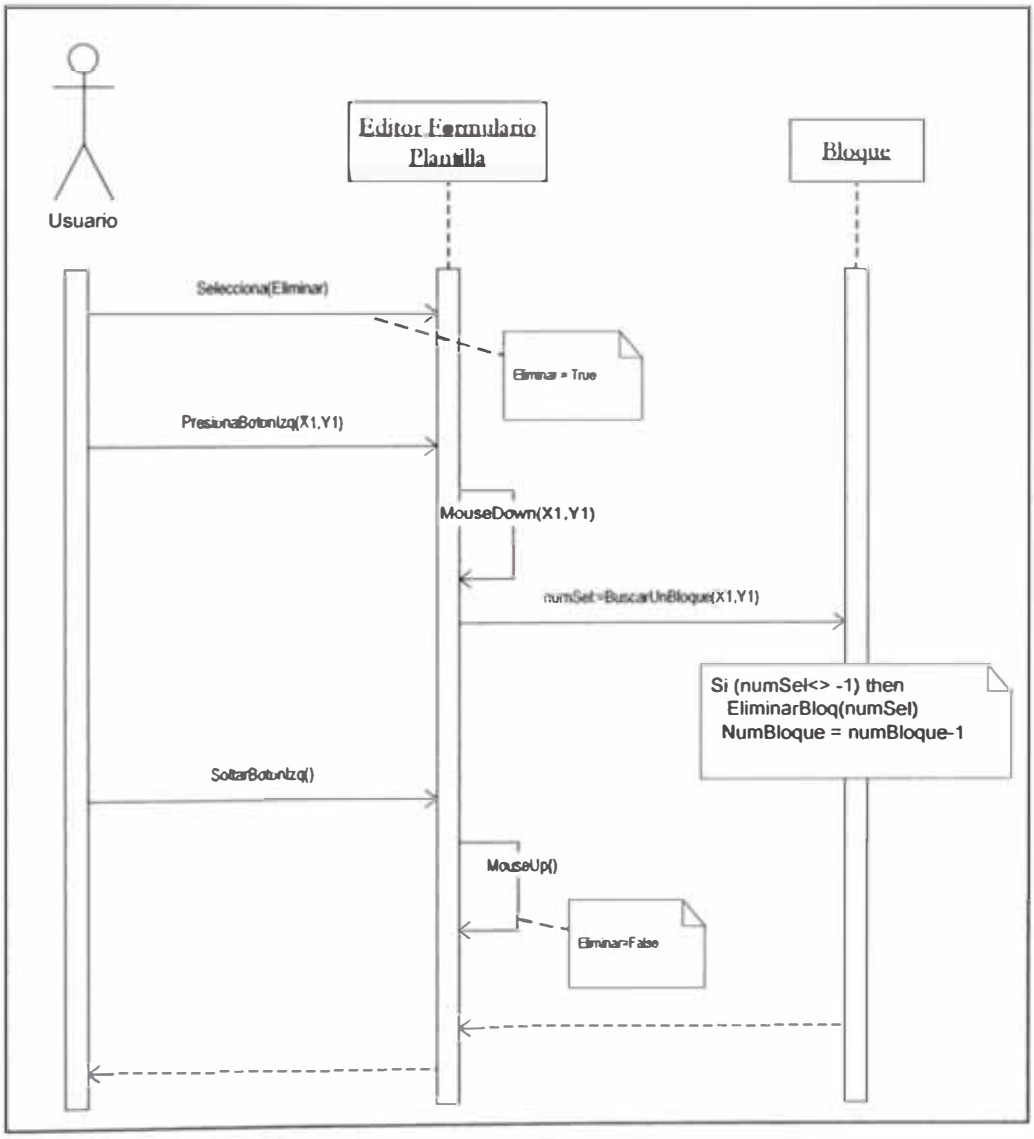


Figura 3.16 Diagrama de Secuencia "Eliminar Bloque"

g) Contrato Arrastrar Bloque

- **Nombre**: Arrastrar Bloque.
- **Responsabilidades**: Permite al usuario cambiar de posición a un bloque ya creado.
- **Tipo**: Interfaz Grafico.
- **Precondiciones**: El objeto bloque debe existir.
- **Postcondiciones** : Solo varia las coordenadas del Bloque.
- **Manejo de errores** : No se permite arrastrar el bloque fuera de la zona de trabajo.

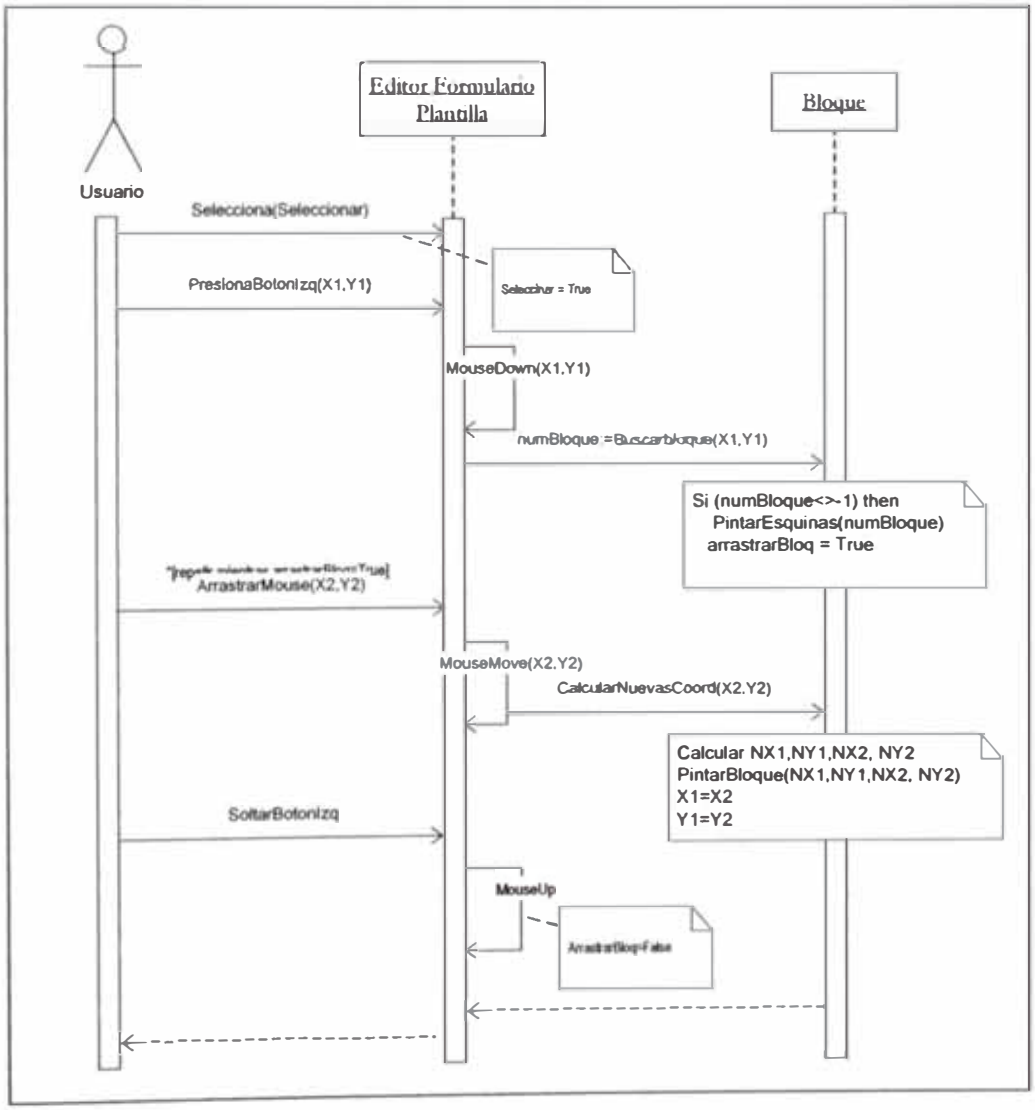


Figura 3.17 Diagrama de Secuencia "Arrastrar Bloque"

h) Contrato Filtrar Ruido en Imagen

- **Nombre**: Filtrar Ruido.
- **Responsabilidades**: Proceso por el cual se elimina manchas y/o puntos aislados en la imagen del campo a reconocer.
- **Tipo**: Procesamiento de Imagen.
- **Precondiciones** : La imagen solo debe contener la zona del campo a reconocer.
- **Postcondiciones** : La imagen del Formulario no se altera, sino mas bien todo el proceso se hace en una imagen temporal.

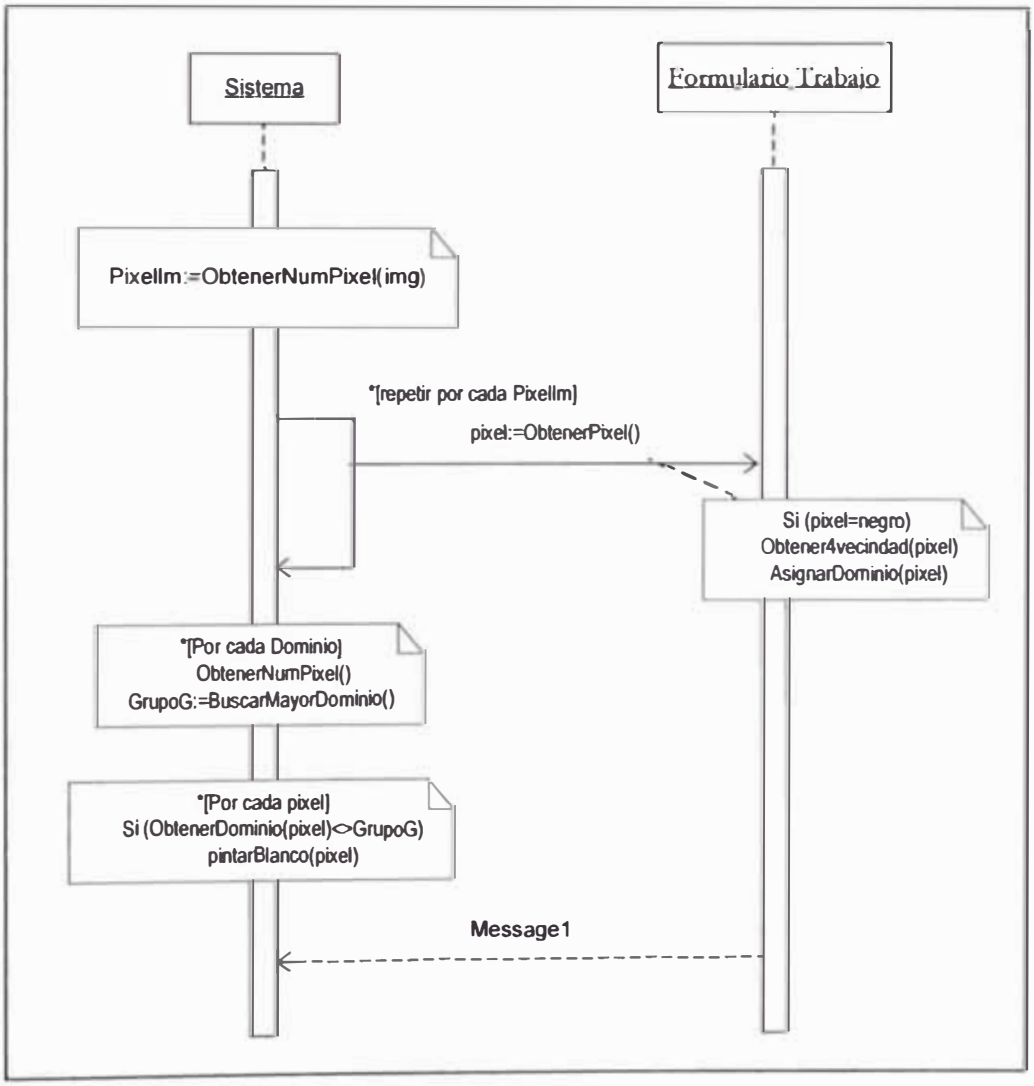


Figura 3.18 Diagrama de Secuencia "Filtrar Ruido"

i) Segmentar Imágenes

- **Nombre**: Segmentar.
- **Responsabilidades**: Permite dividir la imagen temporal de un campo que contiene una palabra o frase en las imágenes de sus caracteres correspondientes.
- **Tipo**: Procesamiento de Imagen.
- **Precondiciones**: La imagen temporal debe estar libre de ruido.
- **Condiciones de Error** : Si la imagen temporal esta en blanco no se producirá ninguna imagen a reconocer.

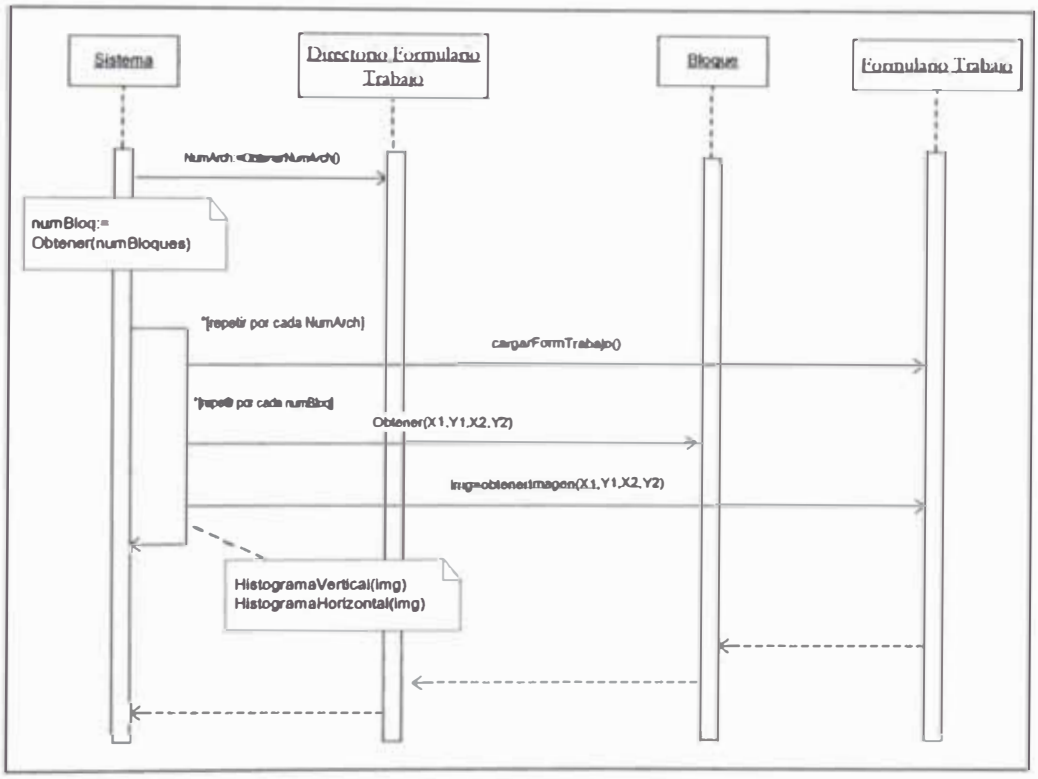


Figura 3.19 Diagrama de Secuencia "Segmentar Imágenes"

j) Contrato Preprocesamiento de Imagen

- **Nombre**: Preprocesamiento.
- **Responsabilidades**: Proceso que se encarga de preparar a la imagen del caracter para su reconocimiento y calcula a la vez el peso de cada neurona de la capa de entrada que representa al carácter a reconocer.

- **Tipo**: Procesamiento de Imagen.
- **Precondiciones** : La imagen del carácter no esta en blanco.
- **Postcondiciones** : La imagen del carácter no se altera.

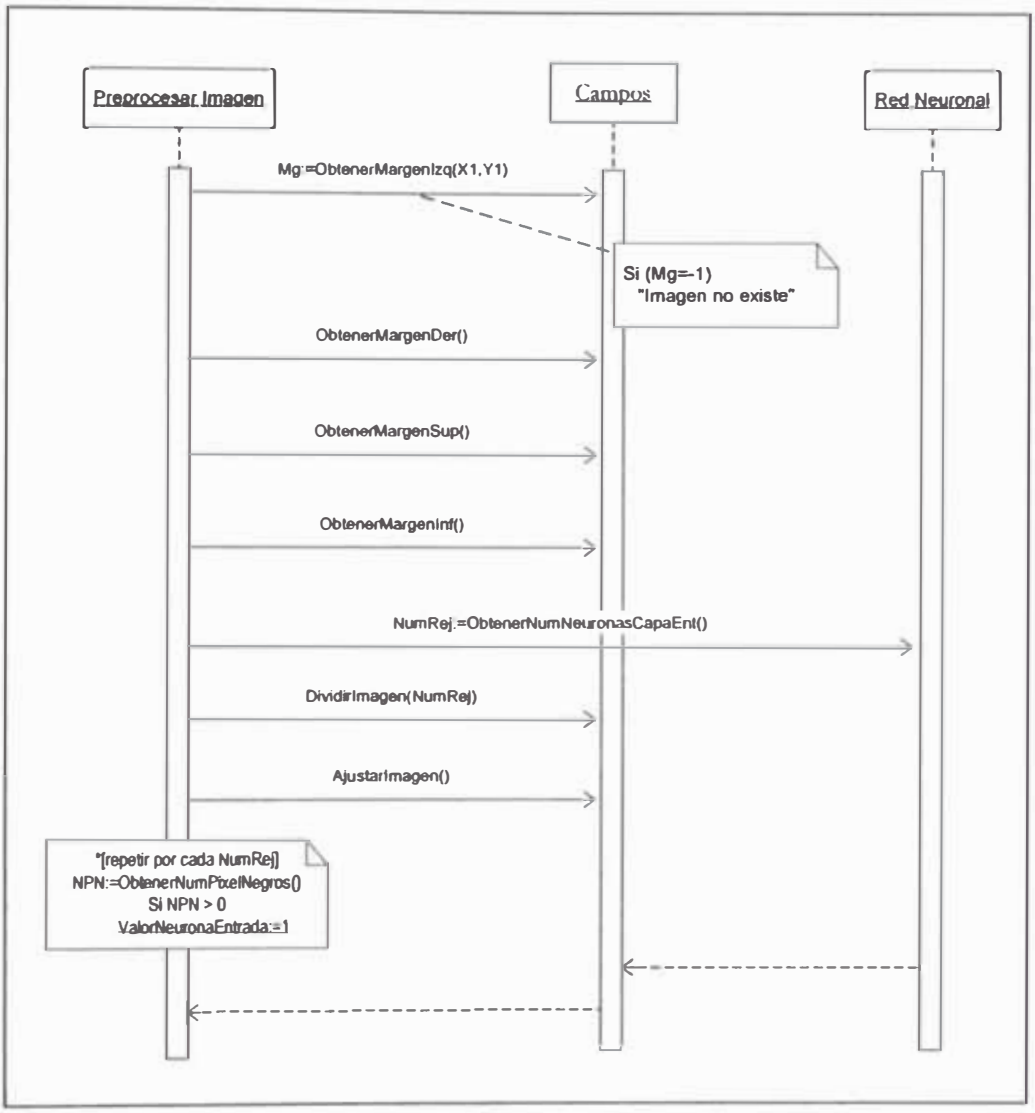


Figura 3.20 Diagrama de Secuencia "Preprocesamiento"

k) Contrato Reconocimiento de Imagen

- **Nombre**: Reconocimiento.
- **Responsabilidades**: Proceso mediante el cual la red Neuronal reconoce uno a uno las imágenes de los caracteres ingresados.
- **Tipo**: Red Neuronal.
- **Precondiciones**: El objeto Red Neuronal debe existir.

- **Postcondiciones** : Por cada imagen de caracteres solo uno de los patrones de la Red Neuronal se activa, y se le reconoce como ganador.

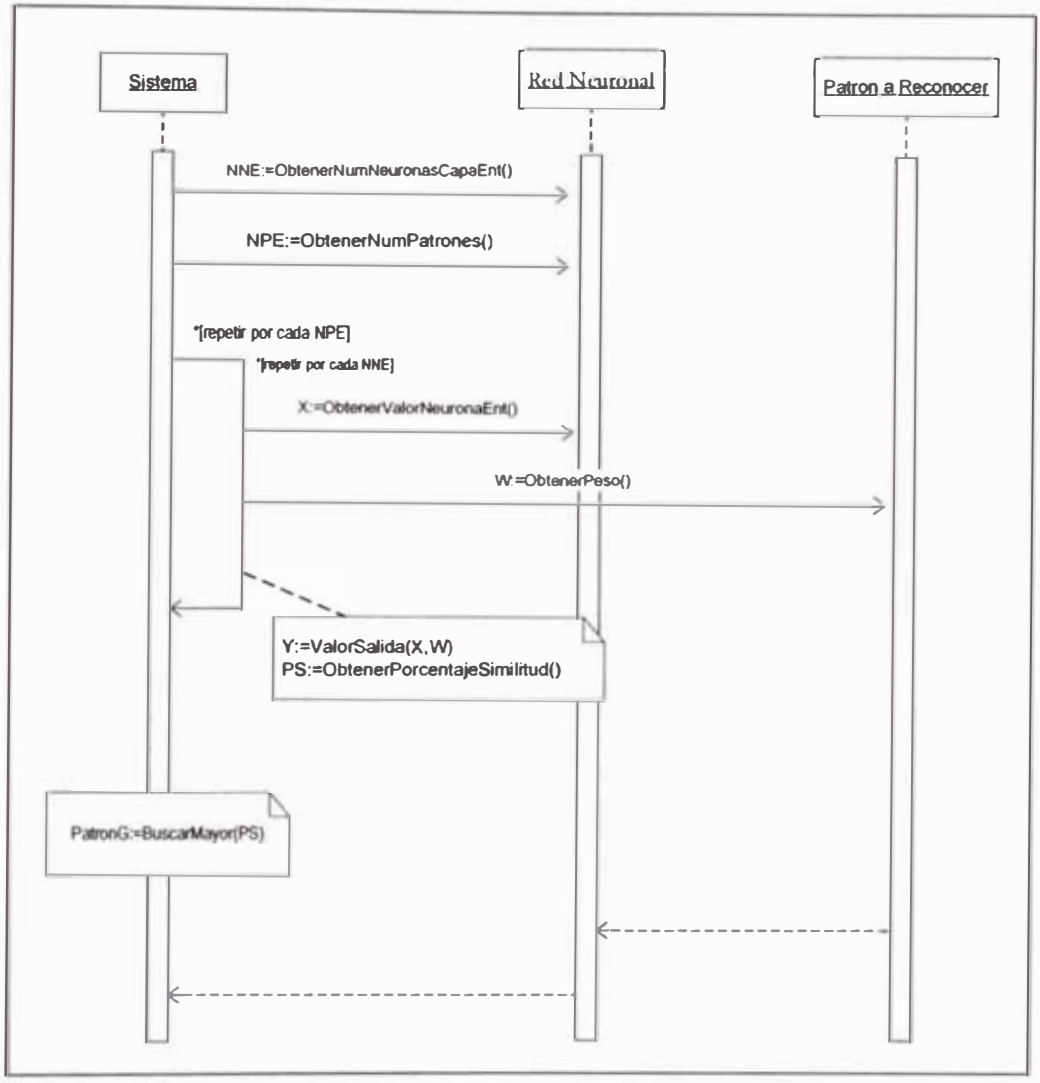


Figura 3.21 Diagrama de Secuencia "Reconocimiento de Imágenes"

l) Contrato Entrenamiento de la Red Neuronal

- **Nombre**: Entrenamiento.
- **Responsabilidades**: Proceso mediante el cual la Red Neuronal se entrena a si misma.
- **Tipo**: Red Neuronal.
- **Precondiciones**: El objeto Red Neuronal debe existir.
- **Postcondiciones** : El patrón entrenado de la Red Neuronal se altera.

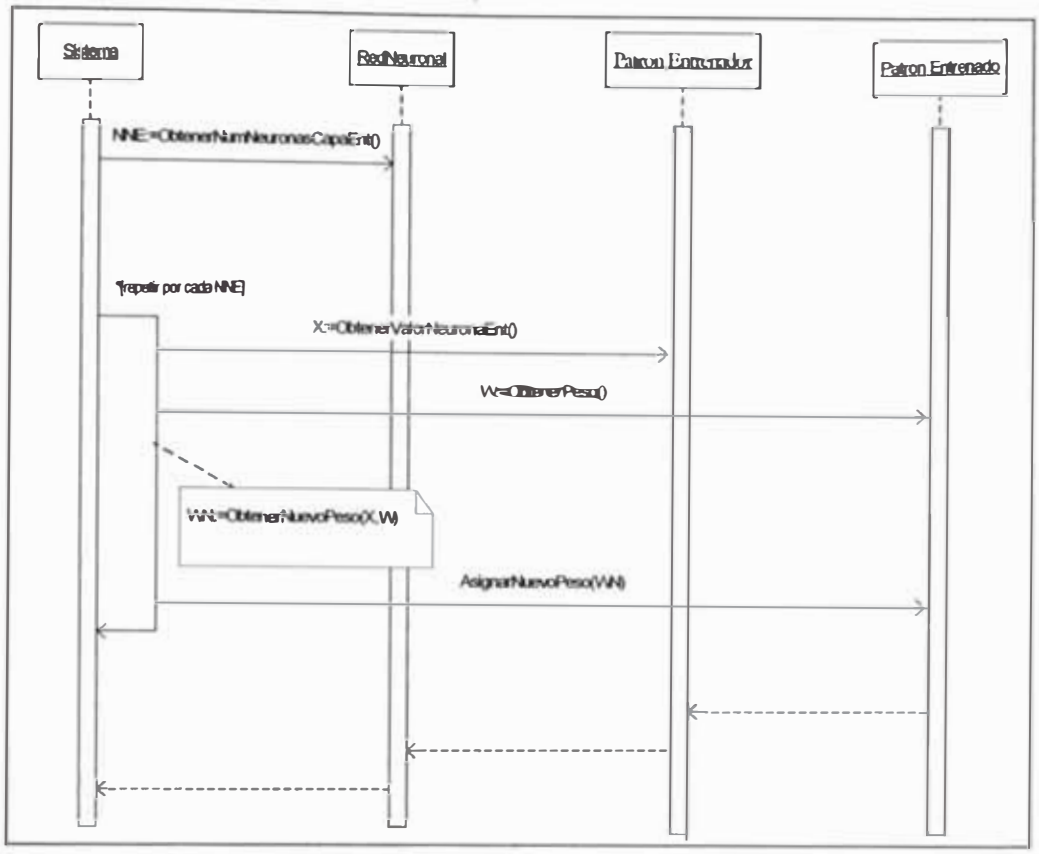


Figura 3.22 Diagrama de Secuencia "Entrenamiento de la Red Neuronal"

3.4.1.6 Diagrama de Clases

Esta es la fase final de Diseño, en la cual se ha concluido con el desarrollo del Diagrama de Clases en base al Modelo Conceptual y a los Diagramas de Colaboración presentados en los subcapítulos anteriores.

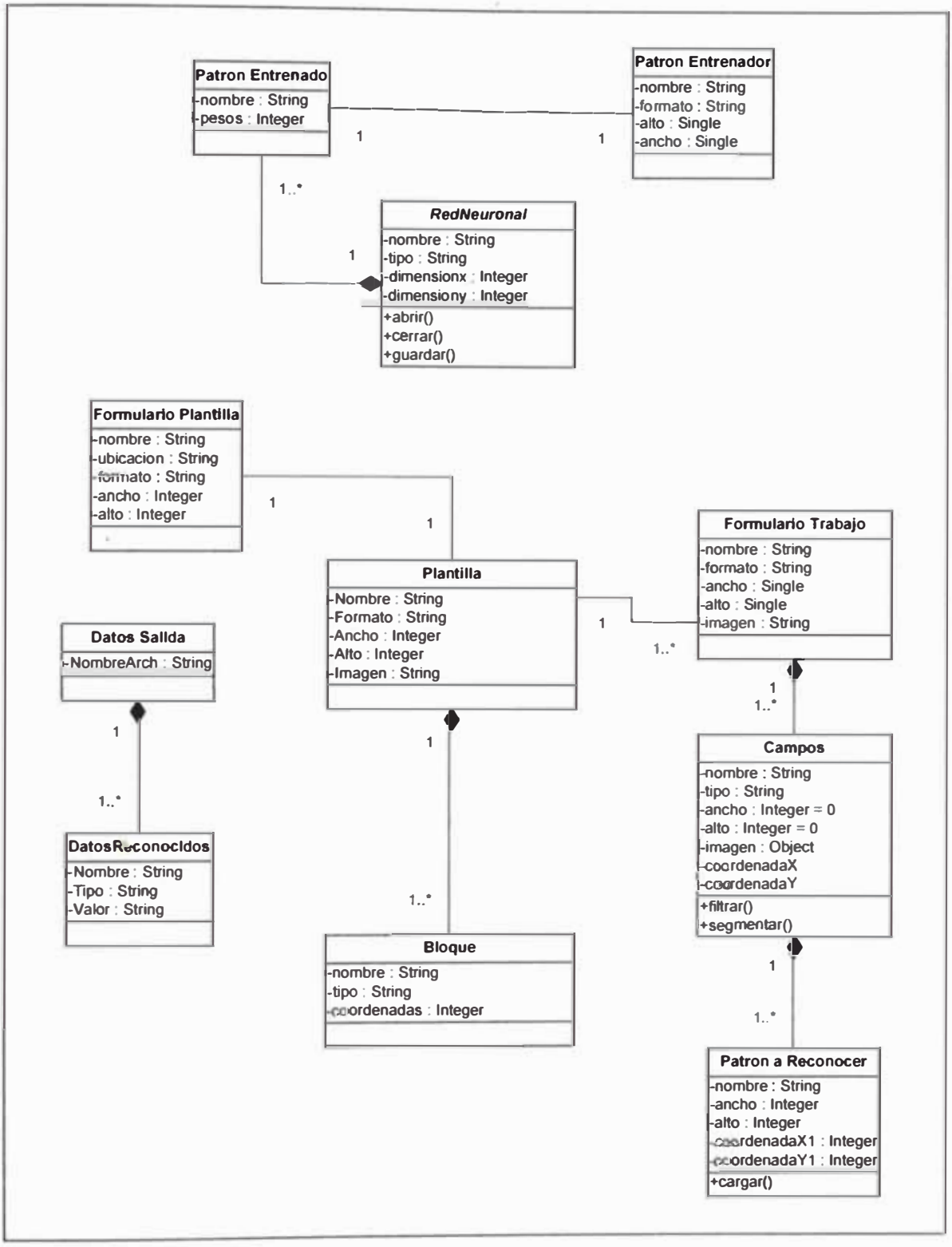


Figura 3.23 Diagrama de Clases

3.4.1.7 Subsistema RECAM y sus Principales Algoritmos.

3.4.1.7.1 Algoritmo de Segmentación

Entrada : Imagen del campo Im_c

Salida : Imágenes de caracteres Ims_{cs}

Descripción : Algoritmo que permite dividir la imagen de un campo que se encuentran enmarcado por un bloque en imágenes de sus caracteres componentes; es decir extrae uno a uno los caracteres de una cadena de caracteres partiendo desde su propia imagen.

Paso 1: Binarizar Im_c donde un valor cero denota un píxel de color negro, y un valor uno denota un píxel de color blanco.

Paso 2: Eliminar los bordes superior e inferior de Im_c, mediante la Técnica de Histograma Horizontal.

Paso 3: Eliminar los bordes izquierdo, derecho y las líneas divisorias de Im_c, mediante la Técnica de Histograma Vertical.

A continuación se describirá la Técnica de Histograma Horizontal e Histograma Vertical, para una mejor compresión se muestra en la siguiente figura la imagen correspondiente a un campo extraída de un Formulario de Trabajo.

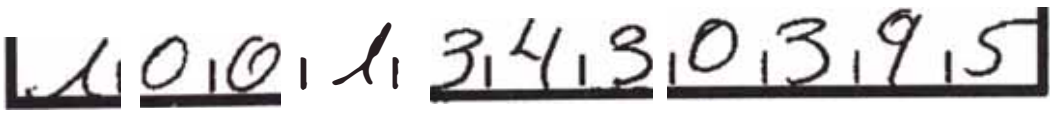


Figura 3.24 Campo a reconocer de un Formulario

a) Técnica de Histograma Horizontal

Entrada : Imagen binarizada del Campo Imb_c

Salida : Imb_c sin bordes superior ni inferior.

Paso 1: Almacenar en el array FrecuenciaFila la cantidad de píxeles de color negro que existen en cada fila de Imb_c, los cuales fueron encontrados en el proceso de barrido por filas.

Paso 2: Buscar en el array FrecuenciaFila el valor más alto, el cual será denominado como "Mayor".

Paso 3: Crear el array LineasH que almacena solo valores binarios (unos y ceros), donde

$$\text{Si} \quad \begin{cases} FrecuenciaFila[i] > Mayor/2 \Rightarrow LineasH[i] = 1 \\ FrecuenciaFila[i] < Mayor/2 \Rightarrow LineasH[i] = 0 \end{cases}$$

Paso 4 : Hacer un barrido del array LíneasH, empezando por los limite inferior y superior hasta encontrar el valor "1", en ambos extremos.

Paso 5: Eliminar la fila de píxeles de color negro de Imb_c, donde el valor almacenado en su correspondiente LineasH[] sea igual a 1, marcando así dos líneas divisorias superior e inferior.

Paso 6: Eliminar los píxeles de color negro de Im_c, que se encuentran por encima de la línea divisoria superior, y por debajo de la línea divisoria inferior.

Continuando con la imagen presentada en la figura 3.23, en la siguiente figura se muestra su histograma horizontal correspondiente.

Figura. 3.25. Histograma Horizontal de la figura 3.24

En la cual se nota claramente que la frecuencia de píxeles de las líneas superiores e inferiores es muy alta, y es que corresponden a los bordes superior e inferior de la imagen de la figura 3.23.

b) Técnica de Histograma Vertical

Entrada : Imagen binarizada del Campo Imb_c sin bordes superior ni inferior.

Salida : Imb_c sin bordes izquierdo, derecho ni líneas divisorias.

Paso 1: Almacenar en el array FrecuenciaCol la cantidad de píxeles de color negro que existen en cada columna de Imb_c encontrados en el proceso de barrido por columnas.

Paso 2: Buscar en el array FrecuenciaCol el valor más alto, denominado como "Mayor"

Paso 3: Crear el array LineasV donde se almacenarán solo valores binarios (unos y ceros), donde

Si

$$FrecuenciaCol[i] \rangle\ Mayor/2 \Rightarrow LineasV[i] = 1$$

$$FrecuenciaCol[i] \langle\ Mayor/2 \Rightarrow LineasV[i] = 0$$

Paso 4: Hacer un barrido del array LíneasV, empezando por el lado izquierdo y terminando por el lado derecho.

Paso 5: Eliminar la columna de píxeles de color negro de Imb_c, donde el valor almacenado en su correspondiente LineasV[] sea igual a 1, marcando así dos líneas divisorias izquierda y derecha.

Paso 6: Eliminar los píxeles de color negro de Imb_c, que se encuentran al lado izquierdo de la línea divisoria izquierda, y a la derecha de la línea divisoria derecha.

Paso 7: Para eliminar las líneas divisorias, es necesario tener en cuenta la proporción del ancho de un carácter con respecto a una línea divisoria, en nuestro caso se ha determinado que si la cantidad de píxeles de color negro continuos es mayor a 10 entonces se trata de un carácter en caso contrario se trata de una línea divisoria.

Continuando con la imagen del ejemplo, en la siguiente figura se muestra su histograma vertical correspondiente:

Figura. 3.26. Histograma Vertical del Campo a Reconocer

3.4.1.7.2 Algoritmo de Eliminación de Ruido

Entrada : Imagen del Carácter a reconocer Ic_c.
Salida : Imagen del Carácter a reconocer Ic_c sin ruido.
Descripción : Mediante este procedimiento se elimina el ruido que pudiera contener la imagen del carácter a reconocer.

Para ello se ha desarrollado un algoritmo basado en el concepto de vecindad de puntos, entendiéndose por esta la sucesión de puntos que se encuentran alrededor de un píxel "Píxeles vecinos", y el concepto de mallado de 4-vecindad, es decir se considera solo cuatro píxeles vecinos para identificar si un píxel de color negro forma parte de una determinada imagen, en caso contrario será aislado y conformara el conjunto de píxeles de otra imagen.

Antes de empezar a detallar el algoritmo, es necesario mencionar que la imagen correspondiente a cada carácter es tratada como una matriz bidimensional donde cada píxel tiene ocho píxeles vecinos a excepción de aquellos píxeles que se encuentran en los extremos superior, inferior, izquierdo y derecho, tal como se muestra en la figura siguiente:

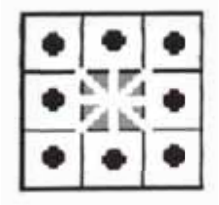


Figura 3.27 Matriz Bidimensional de Píxeles Vecinos

Para optimizar el tiempo de procesamiento (verificación de los ocho píxeles vecinos), solo se ha tenido en cuenta cuatro píxeles vecinos, los cuales están ubicados de la siguiente manera:

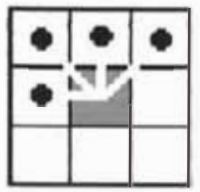


Figura. 3.28 Mallado 4 - Vecindad

A continuación se detalla el algoritmo de filtrado de Ruido implementado en la presente tesis:

Paso 1: Hacer un barrido de la imagen Ic_c empezando en la parte superior y de izquierda a derecha, reconociendo la ubicación de los píxeles de color negro.

Paso 2: Cada vez que se ubica un píxel de color negro PixelN, revisar cada uno de los píxeles que forman parte de su espacio 4-vecindad, en el siguiente orden : píxel vecino izquierdo Pvi, píxel vecino superior izquierdo Pvsi, píxel vecino superior Pvs y píxel vecino superior derecho Pvsd.

Paso 3: Si ninguno de los píxeles (Pvi, Pvsi, Pvs y Pvsd) vecinos a PixelN tuviese asignado algún valor, entonces se asume que PixelN es un píxel aislado y/o forma parte de una nueva imagen, y se le asigna un nuevo valor.

Paso 4: En el caso que los píxeles (Pvi, Pvsi, Pvs y Pvsd) vecinos a PixelN tuviesen valores diferentes, entonces PixelN tomará el valor mínimo, denominándosele "valor PíxelGanador".

Paso 5: Luego se procederá a reemplazar el valor de los otros píxeles vecinos PíxelPerdedor con el valor del PixelGanador y se inicia una búsqueda en retroceso de todos los otros píxeles con el mismo valor "PíxelPerdedor", para realizar el consiguiente

reemplazo, con lo cual se habrá logrado unir dos partes de una misma imagen.

Paso 6 : Una vez examinado todos los píxeles de la imagen, se realiza un conteo de todos los píxeles con el mismo valor y el grupo con mayor frecuencia "frecMayor", será considerada como los píxeles que conforman la imagen del carácter a reconocer libre de ruido.

Paso 7: Como ultimo paso, se procederá a eliminar los píxeles negros de los grupos con menor frecuencia "frecMenor", puesto que estos se les considera como puntos o manchas que no pertenecen a la imagen del carácter a reconocer.

3.4.1.8 Subsistema NEURON y sus Principales Algoritmos.

El subsistema Neuron es el responsable de que las Redes Neuronales realicen la tarea especifica del reconocimiento de caracteres (dígitos o letras); tarea que como se sabe es no es sencilla por lo cual comprende varias subetapas, las cuales de detallan a continuación..

3.4.1.8.1 Diseño de la Red Neuronal

Como se mencionó en el subcapitulo [2.9] "Algoritmo de Aprendizaje Competitivo", para el desarrollo de esta tesis se ha utilizado el Modelo de Red Neuronal Competitiva Simple de dos capas (una capa de entrada y una capa de salida).

Donde, las neuronas de la capa de entrada representan a la imagen del carácter a reconocer (caso de reconocimiento) y a aprender (caso del entrenamiento), por ello, las neuronas de la capa de entrada se encuentran distribuidas en forma bidimensional de $F_{xm}*F_{xn}$., y las neuronas de la Capa de Salida representan al carácter aprendido o entrenado por la Red Neuronal.

Cada una de las neuronas de la capa de entrada están enlazadas a todas las neuronas de la capa de salida, tal como se muestra en la

figura 3.29, por razones de simplicidad y para una mejor comprensión, la Red Neuronal mostrada tiene en su capa de entrada "Fx" 48 neuronas (6 * 8), mientras que en la capa de salida "Fy" tiene 10 neuronas.

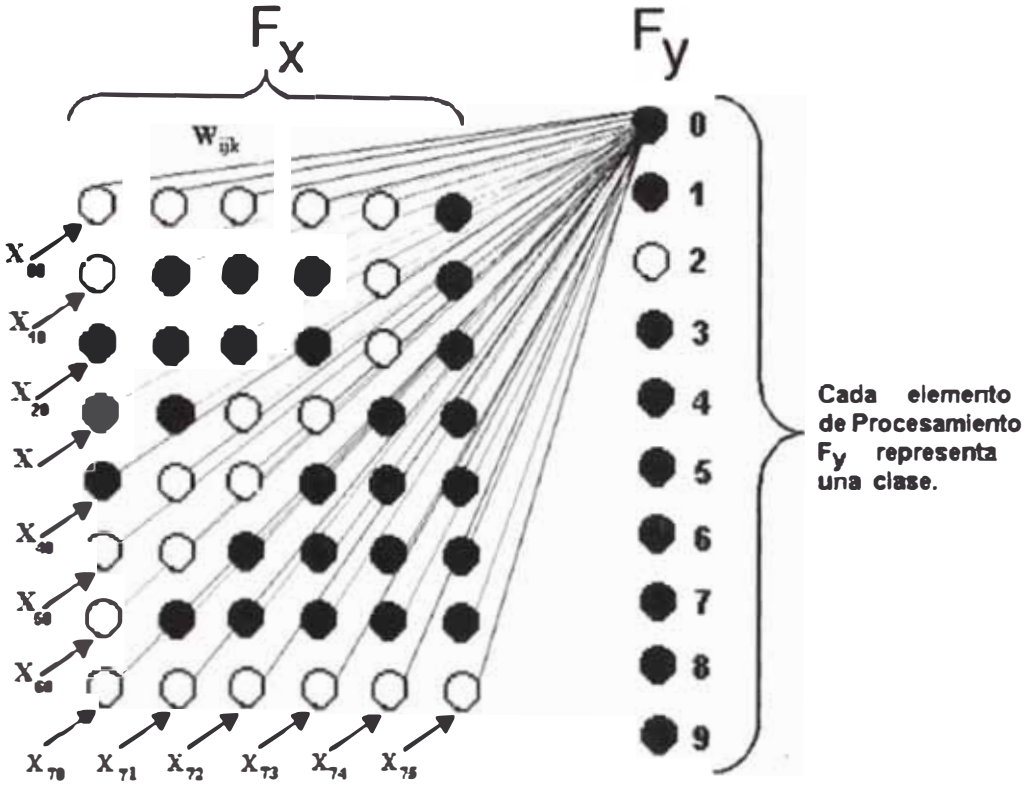


Figura 3.29 Diseño de la Red Neuronal

Donde :

F_x Representa a la capa de entrada de la red neuronal.

F_y : Representa a la capa de salida de la red neuronal, que a su vez representa una clase dada (dígitos o letras).

W_{ijk} : Representa al peso asociado al ijk-ésimo enlace entre las neuronas de capa de Entrada y las neuronas de la capa de Salida, los cuales varían constantemente, de acuerdo al aprendizaje efectuado sobre la Red Neuronal. Inicialmente todos estos pesos tiene el valor "cero".

X_{ij} : Es el valor de entrada asociado a la neurona ij-ésima, de la capa de entrada, que a su vez representa al conjunto de píxeles que conforman la imagen del carácter a reconocer o a aprender, dicho valor será "1", si el píxel

es de color negro y "0" en caso de que el píxel sea de color blanco.

Y_k : Es el valor individual de neurona k-ésima de Salida.

P : Número de neuronas de la capa de salida.

Los valores de salida Y_k (EP) de la Red Neuronal, se calculan mediante la siguiente fórmula:

$$Y_k = \sum_{i=0}^{n} \sum_{j=0}^{m} X_{ij} (-1)^s W_{ijk} \qquad \ldots\ldots\ldots (1)$$

$$\text{Si} \begin{cases} W_{ijk} > 0 \Rightarrow S = 2 \\ W_{ijk} < 0 \Rightarrow S = 1 \end{cases}$$

Donde : k = 1 ... p

También a este tipo de Red Neuronal se denomina "Feedforward Pattern Classification", que en síntesis realiza la tarea de clasificación, activando una neurona de la capa de salida, la cual es la que mejor se ajusta o mejor responde a las neuronas de la capa de entrada.

Un papel importante durante la etapa de reconocimiento y aprendizaje de caracteres lo constituye definir tanto el número de neuronas de la Capa de Entrada y el número de neuronas de la Capa de Salida.

Para definir dichos parámetros se ha teniendo en cuenta la forma de los caracteres a reconocer y aprender; en este caso dígitos y letras, en ellos se ha podido observar una característica importante en cuanto a la proporción del alto y ancho, dichos resultados se muestran en el subcapitulo [3.7] - Dimensiones de la Capa de Entrada de la Red Neuronal.

Mientras más neuronas se tengan en la capa de entrada, la imagen conservará mejor sus característica, sin embargo, tanto del número de neuronas de la capa de entrada como el número de neuronas de la capa de salida son definidas y/o modificados por el usuario del sistema, y para ello no existe un rango específico de valores, pero se debe tener en cuenta que mientras más pequeños sean estos valores se genera una pérdida de las características de los caracteres.

Por otro lado, teniendo en cuenta la complejidad de las tareas de aprendizaje y entrenamiento, y las características propias diferenciales entre dígitos y letras, se han implementado dos (02) Redes Neuronales, una para el reconocimiento de dígitos y otra para el reconocimiento de letras del alfabeto.

En el caso de la Red Neuronal de Dígitos, en su Capa de Salida tiene diez (10) neuronas, y cada una corresponde a un dígito [0,1,2,3,4,5,6,7,8,9].

Y en el caso de la Red Neuronal de Letras, en su Capa de Salida tiene veintiséis (26) neuronas, cada una corresponde a las letras [A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R.S,T,U,V,W,X,Y,Z].

3.4.1.8.2 Preprocesamiento de la Imagen

Entrada : Imagen del Carácter a reconocer o aprender sin ruido Ic_c.

Salida : Imagen del Carácter a reconocer preprocesada Ic_p.

Descripción : Mediante este algoritmo se representa a los píxeles de la imagen del carácter Ic_c, en un conjunto de neuronas de la capa de entrada de la Red Neuronal; en esta fase ya se conoce el número de las neuronas de la capa de entrada, las que se encuentran distribuidas en forma bidimensional; por ello se representa a los pixeles de la imagen del caracter sobre una matriz

bidimensional de M*N elementos donde M es igual a F_{xm} y N es igual a F_{xn}.

Paso 1: Detectar el punto (x1,y1), mas cercano al margen izquierdo, si no se encuentra ningún punto entonces se concluye que no hay ninguna imagen a preprocesar.

Paso 2: Detectar el punto (x2,y2), mas cercano al margen derecho.

Paso 3: Detectar el punto (x3,y3), mas cercano al margen superior.

Paso 4: Detectar el punto (x4,y4), mas cercano al margen inferior.

Paso 5: Delimitar la zona de la imagen por las coordenadas (x1,y3) y (x2,y4).

Paso 6: De acuerdo al tamaño de la imagen y al número de neuronas de la capa de entrada, dividir la imagen uniformemente en pequeñas zonas que denominaremos "rejillas". El número de estas rejillas es igual al número de neuronas de la capa de entrada.

Paso 7: Tanto el ancho como el alto de estas rejillas son calculadas por las siguientes fórmulas:

$$Ancho\_de\_rejilla = \frac{lado\_derecho - lado\_izquierdo}{F_{xm}} \quad \ldots\ldots\ldots (2)$$

$$Alto\_de\_rejilla = \frac{lado\_\inf erior - lado\_\sup erior}{F_{xn}} \quad \ldots\ldots\ldots (3)$$

Paso 8: Si la imagen del caracter se encuentra desproporcionada se procede a ajustar tanto el alto como el ancho de la misma, es decir se debe evitar que la imagen sea muy alta y delgada y muy ancha y corta Lo ideal, como lo mencionamos antes, es que el ancho y el alto de la imagen tengan una proporción de 1 a 1.5.

Paso 9: En el primer caso, si la imagen es muy alta y delgada, calcular una proporción extra de ancho, que permite

reajustar tanto el margen izquierdo y derecho, para evitar esta anomalía.

$$Extra\_ancho = Int\left(\left(\frac{Alto\_de\_rejilla}{1.5} - Alto\_de\_rejilla\right) * \frac{F_{xm}}{2} + 0.5\right)$$

..... (4)

$$Lado\_izquierdo = Lado\_izquierdo - Extra\_ancho$$

..... (5)

$$Lado\_derecho = Lado\_derecho + Extra\_ancho$$

..... (6)

Paso 10: En el segundo caso, si la imagen es muy ancha y corta, se calculará una proporción extra de alto, que permite reajustar tanto el margen superior e inferior, para evitar esta anomalía.

$$Extra\_alto = Int\left(\left(\frac{Ancho\_de\_rejilla}{1.5} - Ancho\_de\_rejilla\right) * \frac{F_{xn}}{2} + 0.5\right)$$

..... (7)

$$Lado\_inferior\_ = Lado\_inferior + Extra\_alto$$

..... (8)

$$Lado\_superior = Lado\_superior - Extra\_alto$$

..... (9)

Paso 11: Recalcular tanto el ancho como el alto de la imagen y por consiguiente el alto y ancho de cada rejilla.

$$Ancho\_imagen = Lado\_derecho - Lado\_izquierdo$$

..... (10)

$$Alto\_imagen = Lado\_inferior - Lado\_superior$$

..... (11)

$$Ancho\_de\_rejilla = \frac{lado\_derecho - lado\_izquierdo}{F_{xm}} \quad \ldots\ldots\ldots (12)$$

$$Alto\_de\_rejilla = \frac{lado\_\inf erior - lado\_\sup erior}{F_{xn}} \quad \ldots\ldots\ldots (13)$$

Paso 12: Calcular el valor representativo de cada rejilla, como se sabe cada una de éstas rejillas agrupan una cantidad determinada de píxeles; que por razones de complejidad no pueden ser tratados uno a uno, sino mas bien todos en conjunto tendrán un valor representativo. Dicho valor será "1" si al menos uno de estos píxeles es de color negro y "0" en caso contrario, y es este valor el que finalmente representa al valor de la neurona de la Capa de Entrada correspondiente.

Paso 13: Como ultimo paso, verificar el número de píxeles relacionados a cada casilla, si este es mayor o igual a 1 entonces se procederá a pintar de color negro toda la rejilla.

Para una mejor compresión del algoritmo a continuación se presenta el siguiente ejemplo:

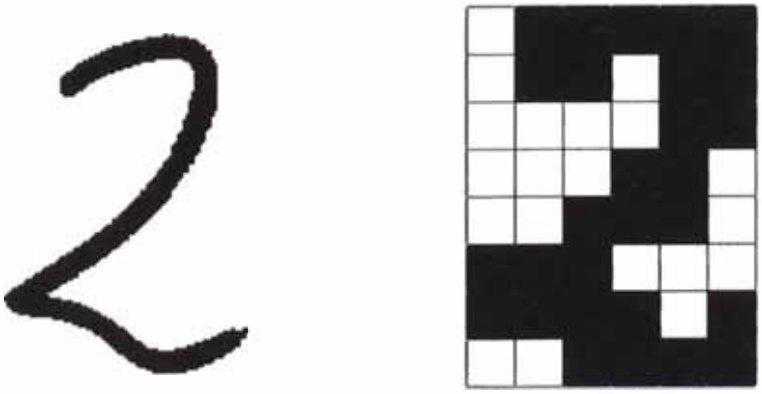


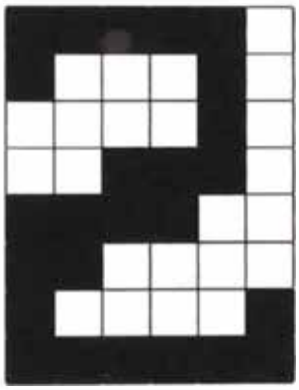
Figura. 3.30 Izquierda. Imagen manuscrita, Derecha. Imagen Preprocesada

3.4.1.8.3 Reconocimiento de la Imagen del carácter.

Entrada : Imagen preprocesada del Carácter Ip_c a reconocer o aprender.

Salida : Carácter reconocido.

Descripción : Mediante este algoritmo se describe la forma como la Red Neuronal reconoce los caracteres, para lo cual tomaremos como ejemplo un caso particular en el que la Red Neuronal contiene 48 neuronas en su capa de entrada, y una distribución bidimensional de 6*8.



1	1	1	1	1	0
1	0	0	0	1	0
0	0	0	0	1	0
0	0	1	1	1	0
1	1	1	1	0	0
1	1	0	0	0	0
1	0	0	0	0	1
1	1	1	1	1	1

Figura. 3.31 Izquierda. Imagen preprocesada, Derecha. Matriz que representa a los valores de las neuronas de la Capa de Entrada de la RN.

La matriz que se representa a la derecha de la figura 3.31, contiene los valores de las neuronas de la Capa de Entrada "X_{ij}", la cual será comparada con cada uno de los patrones definidos (patrones entrenados) en la Red Neuronal, para este caso, se le va ha comparar con el patrón entrenado "Dos" de la Red Neuronal de Dígitos, dicho patrón debe tener asignado en una matriz bidimensional, los pesos de los enlaces "W_{yk}" entre las neuronas de la Capa de entrada y las neuronas de las capas de Salida de la Red Neuronal.

2	2	2	2	2	-2
2	-2	-2	0	2	-2
-2	-2	-2	0	2	-2
-2	-2	2	2	0	-2
0	2	2	0	-2	-2
2	2	-2	-2	-2	-2
2	0	-2	-2	-2	0
2	2	2	2	2	2

1	1	1	1	1	0
1	0	0	0	1	0
0	0	0	0	1	0
0	0	1	1	1	0
1	1	1	1	0	0
1	1	0	0	0	0
1	0	0	0	0	1
1	1	1	1	1	1

Figura. 3.32 Izquierda. Matriz de los pesos W_{ijk}, del patrón entrenado "dos", Derecha. Matriz de las neuronas de la capa de entrada X_{ij} del patrón a reconocer.

Paso 1: Calcular los valores de Salida Y_k, mediante la siguiente fórmula:

$$Y_k = \sum_{i=0}^{n} \sum_{j=0}^{m} X_{ij} (-1)^s W_{ijk}$$

. . . . . (14)

$$\text{Si} \begin{cases} W_{ijk} > 0 \implies S = 2 \\ W_{ijk} < 0 \implies S = 1 \end{cases}$$

Así para el ejemplo se tiene:

$Y_2 =$ 2+2+2+2+2+2
2+2+2+0+2+2
2+2+2+0+2+2
2+2+2+2+0+2
0+2+2+0+2+2
2+2+2+2+2+2
2+0+2+2+2+0
2+2+2+2+2+2 = **82**

Paso 2: Hallar el porcentaje de similitud que tiene el patrón a reconocer hacia el patrón entrenado, mediante la siguiente fórmula:

$$\% = \frac{(suma * 100)}{(\max * F_{xm} * F_{xn})} \quad \ldots\ldots\ldots (15)$$

Donde "max" es el valor absoluto del máximo valor "W_{ijk}", que para el caso del ejemplo es "2", por lo tanto el porcentaje de similitud es:

$$\% = 85.416$$

En este caso el porcentaje de similitud entre ambos patrones fue alto debido al gran parecido que muestran ambos patrones.

3.4.1.8.4 Algorítmo de entrenamiento No Supervisado de la Red Neuronal.

Entrada : Imagen preprocesada del patrón entrenador P_e.
Salida : Red Neuronal Entrenada.
Descripción : Mediante este método la red neuronal sigue un auto entrenamiento, con el objetivo de lograr cada vez un reconocimiento con menos errores.

Para ello es necesario introducir el concepto de "Porcentaje aceptable", que se refiere a aquel porcentaje de similitud entre el patrón entrenado y el patrón a reconocer, es decir en base a que porcentaje se considera que ambos patrones guardan similitud.

Cualesquier valor de porcentaje que sea menor del porcentaje aceptable entonces provocara que la red neuronal agregue un nuevo patrón a la lista de patrones entrenados, en caso contrario la Red Neuronal procederá a entrenar al patrón con mayor porcentaje de similitud.

En todo este proceso la Red Neuronal es la que toma todas las decisiones y cada vez que se ingresa un nuevo patrón a reconocer la red decidirá teniendo en cuenta el porcentaje aceptable si será necesario añadir un nuevo patrón a la Red Neuronal o se procederá a entrenar a aquel patrón que tenga el porcentaje de similitud más alto.

El método de entrenamiento es el mismo que para el Aprendizaje Supervisado, por ello será descrito en la siguiente sección.

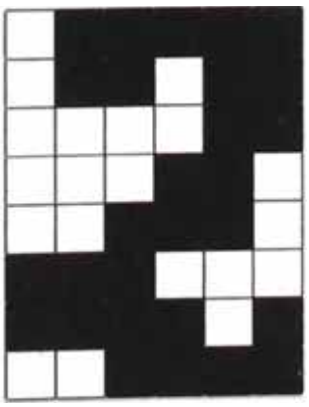
3.4.1.8.5 Algorítmo de entrenamiento Supervisado de la Red Neuronal.

Entrada : Imágenes preprocesadas del patrón entrenador P_e.

Salida : Red Neuronal Entrenada.

Descripción : Mediante este algoritmo, la red neuronal puede ser manipulada por el usuario, es decir el usuario es el que toma todas las decisiones, tales como: añadir un nuevo patrón a red neuronal, eliminar un patrón de la red neuronal, entrenar un patrón específico, etc. Teniéndose siempre el objetivo de lograr cada vez un reconocimiento más perfecto.

Paso 1: Obtener el porcentaje de similitud entre P_e y los patrones entrenados de la Red Neuronal, ello se logra cuando la Red Neuronal inicia el proceso de reconociendo sobre P_e. Sea el patrón entrenador.



0	1	1	1	1	1
0	1	1	0	1	1
0	0	0	0	1	1
0	0	0	1	1	0
0	0	1	1	1	0
1	1	1	0	0	0
1	1	1	1	0	1
0	0	1	1	1	1

Figura. 3.33 Izquierda Patrón Entrenador, Derecha Matriz de valores de las neuronas de la capa de entrada, correspondiente al patrón entrenador.

Entonces el usuario tiene la opción de elegir a uno de los patrones entrenados al cual desea entrenar, en este caso como el patrón entrenador guarda una gran similitud con el patrón entrenado "Dos", se elegirá a éste para que sea entrenado y se convirtiéndose en el patrón a entrenar.

1	3	3	3	3	-2
2	1	-2	-3	2	3
-1	-3	-3	-3	-3	3
-3	-3	-3	-1	2	3
-3	-3	-2	0	2	0
-3	-2	1	3	1	-3
-2	3	2	-1	-3	-3
3	3	3	3	3	3

Figura 3.34 Matriz de los pesos W_{ijk}, del Patrón a entrenar "2"

Paso 2: Calcular los nuevos pesos W_{ijk} del patrón a entrenar, para ello se utiliza la siguiente fórmula:

$$W_{ijk} = W_{ijk} + (-1)^{s} \qquad \ldots\ldots\ldots (16)$$

$$\text{Si} \begin{cases} X_{ij} > 0 \implies S = 2 \\ X_{ij} \leq 0 \implies S = 1 \end{cases}$$

Donde :

k : Es el patrón que se desea entrenar, en este caso "2".

W_{ijk} : Es el peso de la neurona ijk-ésima, del patrón a entrenar.

X_{ij} : Es el valor de la neurona ij-ésima del patrón entrenador

Es decir, una neurona ij-ésima con valor cero del patrón entrenador, implica restar "uno" a su correspondiente valor de la neurona ij-ésima del patrón a entrenar. Y una neurona ij-ésima con valor uno

del patrón entrenador, implica sumar "uno". Por lo cual los valores de las neuronas de nuestro patrón a entrenar serán los siguientes:

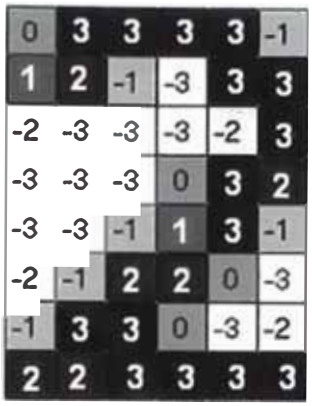


Figura 3.35 Matriz con los nuevos pesos W_{ijk}, del patrón "2"

Al finalizar esta etapa el porcentaje de similitud entre ambos patrones se ha incrementado.

3.5 Interfaz de Usuario

Esta etapa ha sido caracterizada por el desarrollo de una interfaz visual amigable, con un Menú tipo Barra Horizontal ubicado en la parte superior que contiene un conjunto de iconos que representan a las tareas más frecuentes accedidas por el usuario y contiene además Submenús desplegables.

Según el diseño de la arquitectura del Sistema de Reconocimiento de Caracteres manuscritos aislados en campos de formularios, presentado en el subcapítulo [3.4.1.3] Diseño Arquitectónico, según el cual se tiene dos subsistemas principales:

- Recam
- Neuron

Por lo tanto para cada uno de estos subsistemas se ha diseñado su propia interfaz de usuario.

3.5.1 Subsistema RECAM

3.5.1.1 Diseño de Pantallas

El conjunto de pantallas que conforman la interfaz de usuario del Subsistema Recam, están relacionadas entre si, puesto que poseen una relación de creación. A continuación se muestra el diagrama que contiene dichas relaciones:

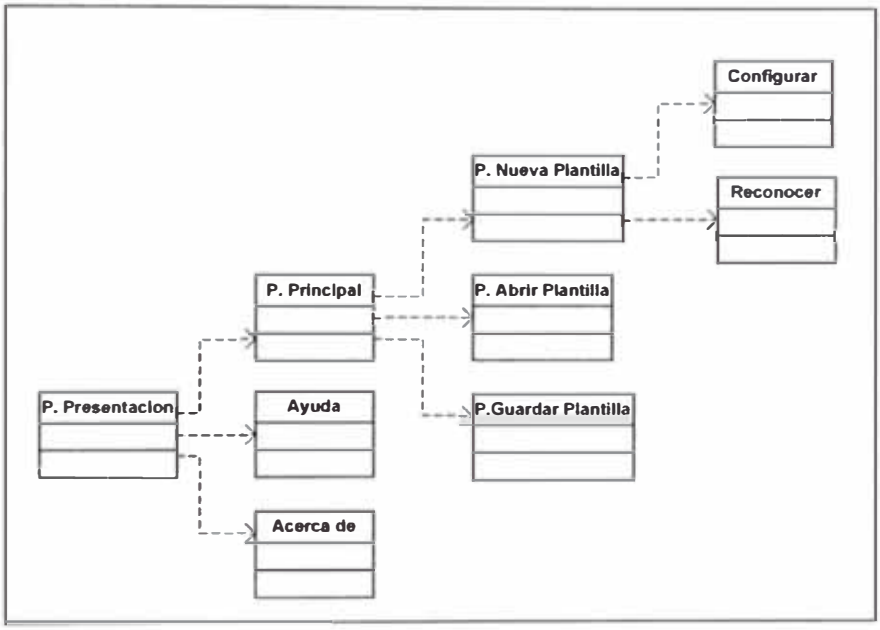


Figura. 3.36 Diagrama de pantallas del Subsistema Recam

A) Pantalla de Presentación

Esta pantalla es la que se muestra al inicio del programa.

Figura. 3.37 Pantalla de presentación del subsistema Recam

B) Pantalla Principal

Esta pantalla contiene tanto el Menú del sistema como la Barra de herramientas.

Al inicio muchos de los Submenús están desactivados puesto que estos dependen de otros o necesitan que se realicen tareas previas.

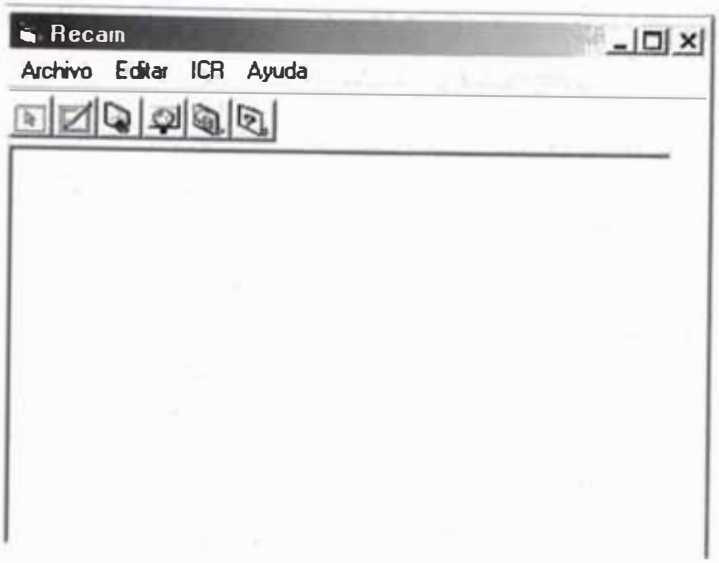


Figura 3.38 Pantalla principal del subsistema Recam

El Menú principal ha sido estructurado de la manera siguiente:

Archivo

- Nueva Plantilla
- Abrir Plantilla
- Cerrar
- Guardar
- Guardar Como
- Salir

Editar

- Seleccionar Bloque
- Insertar Bloque
- Eliminar Bloque

ICR

- Reconocer
- Configuración

Ayuda

- Ayuda
- Acerca de

Los iconos que componen la Barra de Herramientas, representan a las funciones más utilizadas del Menú del Sistema:

 Seleccionar

 Dibujar Bloque

 Eliminar Bloque

 Zoom

 Guardar Plantilla

 Ayuda

C) Pantalla Editar Plantilla

En esta pantalla, el usuario puede editar una Plantilla, lo cual implica múltiples funciones tales como: Seleccionar Bloque, Insertar bloque, Eliminar Bloque, Mover Bloque, Abrir Plantilla, Nueva Plantilla y Guardar Plantilla.

Una vez que se ha elegido el Submenú Nueva Plantilla o el Submenú Abrir Plantilla, se le muestra al usuario en pantalla la imagen del Formulario Plantilla que desea editar.

Figura. 3.39 Pantalla de Edición del subsistema Recam

Como se observa, en la pantalla de la figura 3.38, en la parte inferior y en el lado derecho, se muestran las Barras de Desplazamiento, que son de gran ayuda en caso de que las dimensiones del Formulario Plantilla superen tamaño predeterminado.

D) Pantalla Configuración

Antes de empezar con el trabajo de Reconocimiento, es necesario que el usuario especifique información de entrada, de la Red Neuronal y de Salida:

1. **Datos de Entrada:**
   - **Carpeta de Trabajo:** Ubicación exacta de la carpeta donde se encuentran almacenados los Formularios de Trabajo, es decir aquellos Formularios de los cuales se va ha extraer los caracteres ha ser reconocidos.

2. **Red Neuronal :**
   - **Red Neuronal de Números:** Mediante esta caja de texto, el usuario debe especificar la ubicación exacta del archivo "DNR", que contiene la Red Neuronal Entrenada de patrones numéricos.

- **Red Neuronal de Letras :** Mediante esta caja de texto, el usuario debe especificar la ubicación exacta del archivo "DNR", que contiene la Red Neuronal Entrenada de patrones de letras.

3. **Datos de Salida:**
   - **Base de Datos de Salida:** Ubicación exacta del archivo de Base de Datos en el cual el Sistema almacenará los resultados obtenidos en el proceso de Reconocimiento.

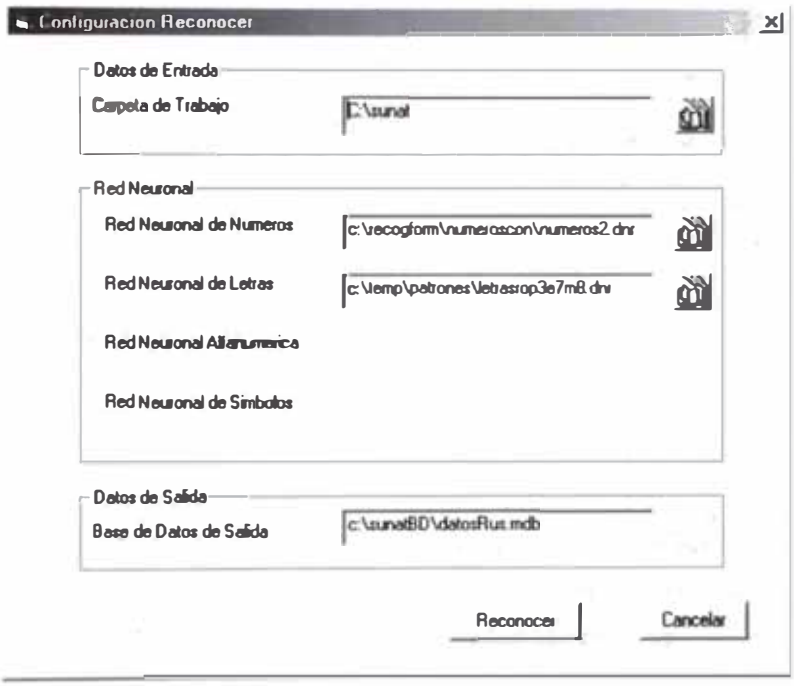


Figura. 3.40 Pantalla de Configuración del subsistema Recam

Una vez que el usuario ha proporcionado todos estos datos : de entrada, salida y Red Neuronal al Sistema, puede presionar el botón reconocer, para que el Sistema inicie el proceso de Reconocimiento, o cancelar en caso de que desee anular la operación de configuración.

E) Pantalla Ayuda

El objetivo de esta pantalla es proporcionar ayuda al usuario en el uso del subsistema Recam, para lo cual presenta un submenú con las

siguientes especificaciones: Introducción, Menú Principal, Reconocimiento y Datos Reconocidos.

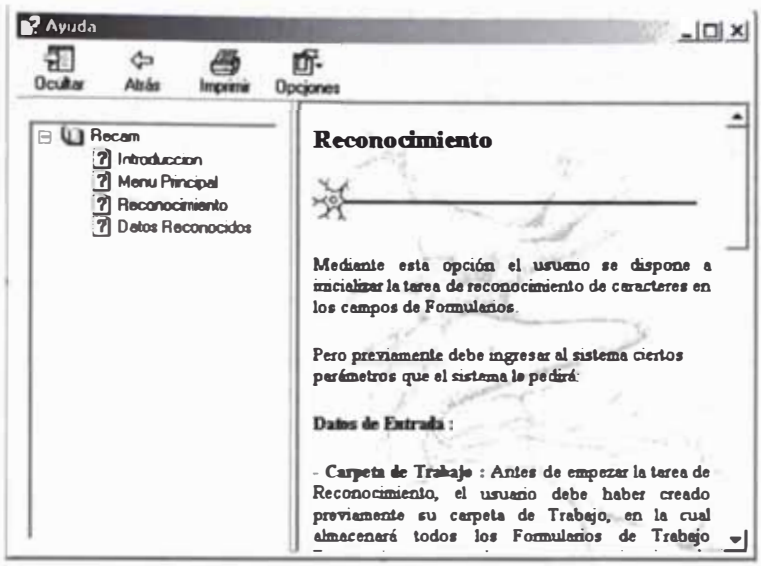


Figura. 3.41 Pantalla de Ayuda Recam del subsistema Recam

3.5.2 Subsistema NEURON

3.5.2.1 Diseño de Pantallas

El diseño de Pantallas para el subsistema Neuron, es bastante sencillo, consta básicamente de una pantalla principal en la que el usuario puede interactuar con el sistema, ya sea para crear o entrenar a la Red Neuronal, así como también puede el usuario utilizar este subsistema para Reconocer caracteres individuales.

A continuación se muestra el diagrama que contiene el conjunto de pantallas del subsistema Neuron:

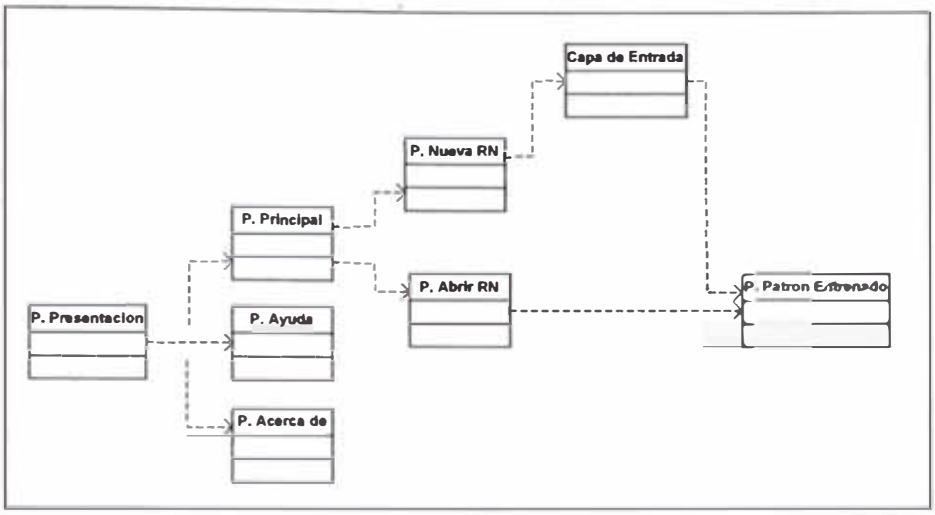


Figura 3.42 Diagrama de Pantallas del subsistema Neuron

A) Pantalla de Presentación

Esta es la pantalla que se muestra al inicio del subsistema Neuron :

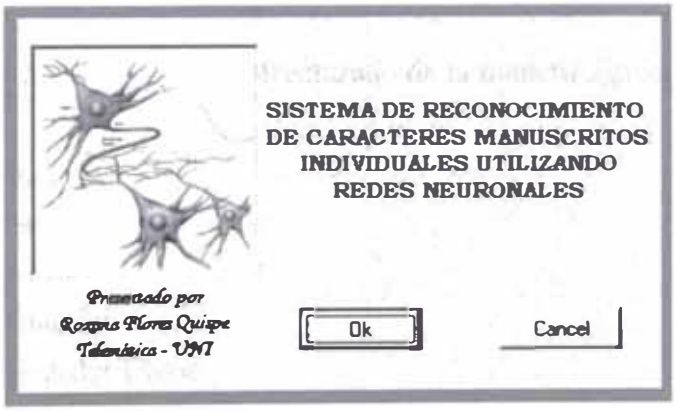


Figura 3.43 Pantalla de Presentación del subsistema Neuron

B) Pantalla Principal

Esta pantalla contiene tanto el Menú Principal y la Barra de herramientas del subsistema Neuron.

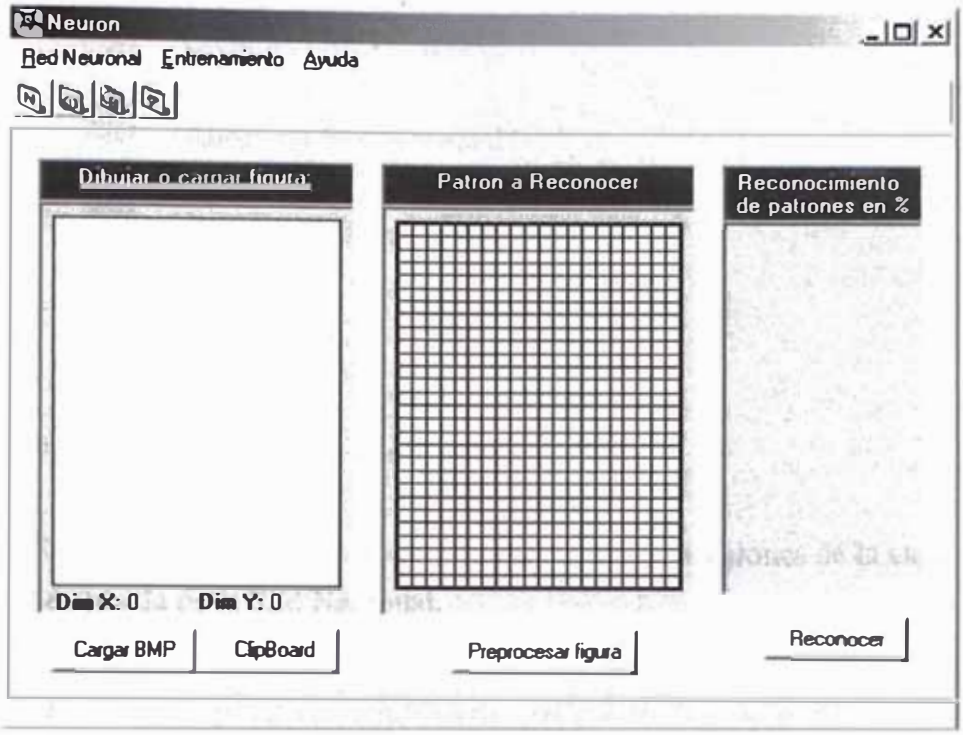


Figura. 3.44 Pantalla Principal del subsistema Neuron

El Menú principal ha sido estructurado de la manera siguiente:

Red Neuronal

- Nueva
- Abrir
- Guardar
- Guardar Como
- Salir

Entrenamiento

- Ninguno
- Supervisado
- No Supervisado

Ayuda

- Ayuda
- Acerca de

Los componentes de la Barra de Herramientas, contiene iconos con las funciones más utilizadas del Menú del subsistema Neuron:

Crear una Red Neuronal

Abrir una Red Neuronal

Guardar Red Neuronal

Ayuda

C) Pantalla Capa de Entrada

Esta pantalla le permite al usuario definir las dimensiones de la capa de entrada de la Red Neuronal.

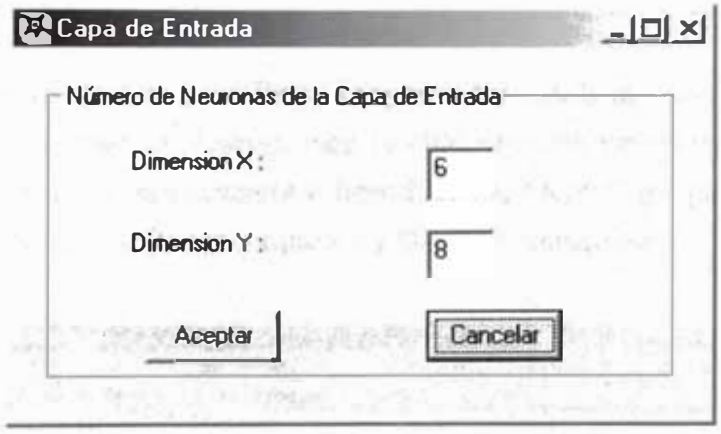


Figura. 3.45 Pantalla de la Capa de Entrada

Es decir, que la capa de entrada de la Red Neuronal estará formada por 48 neuronas o elementos de procesamiento.

D) Pantalla Patrón Entrenado

En esta pantalla, el usuario puede visualizar gráficamente la representación de un determinado patrón entrenado de la Red Neuronal.

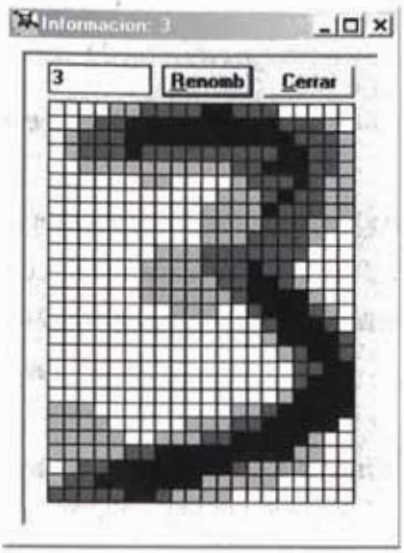


Figura. 3.46 Pantalla Patrón Entrenado

E) Pantalla Ayuda

El objetivo de esta pantalla es proporcionar ayuda al usuario en el uso del subsistema Neuron, para lo cual presenta un submenú con las siguientes especificaciones: Introducción, Menú Principal, Barra de Herramientas, Reconocimiento y Datos Entrenamiento.

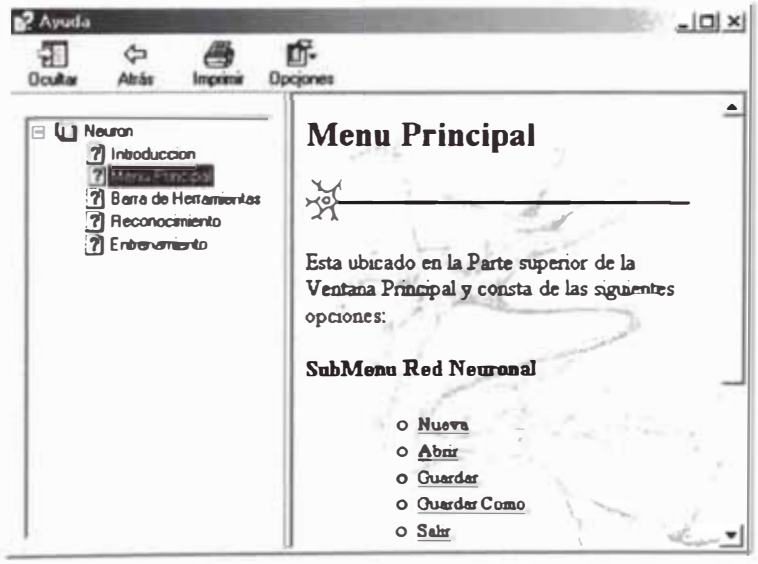


Figura. 3.47 Pantalla de Ayuda del subsistema Neuron

3.6 Reportes Generados

3.6.1 Formato de Archivos "PLA"

La información correspondiente a la Plantilla Formulario creada por el usuario, se almacena en archivos de texto, cuya extensión es "pla", esta plantilla se crea luego de que el usuario ha seleccionado los bloques que encierran a los campos que desea que el subsistema Recam reconozca.

La información almacenada en estos archivos esta organizada de la siguiente manera:

```
Numero_de_Bloques:        XX
Plantilla    YY

0
Nombre_del_bloque
Tipo_del_bloque
Tipo_ de_rejilla
  CoordenadaX1
  CoordenadaY1
  CoordenadaX2
  CoordenadaY2
1
Nombre_del_bloque
Tipo_del_bloque
Tipo_ de_rejilla
  CoordenadaX1
  CoordenadaY1
  CoordenadaX2
  CoordenadaY2
.....
  .....
  .....
  .....
N
Nombre_del_bloque
Tipo_del_bloque
Tipo_ de_rejilla
  CoordenadaX1
  CoordenadaY1
  CoordenadaX2
  CoordenadaY2
```

Figura. 3.48 Estructura del archivo "PLA"

Donde :

XX : Numero de Bloques
YY : Ubicación de la plantilla, por ejemplo :
C:\RecogForm\concytecRo\docu0001.bmp

3.6.2 Formato de Archivos "DNR"

Una vez que se ha definido y entrenado a la Red Neuronal, la información correspondiente se almacena en un archivo de texto, cuya extensión es "dnr" y su estructura se muestra a continuación:

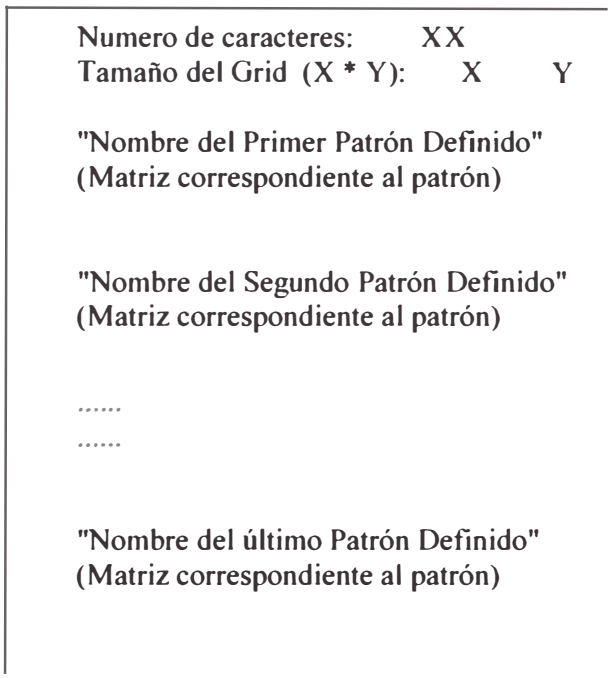


Figura. 3.49 Estructura del archivo "DNR"

Donde :

Número de caracteres	Es el número total de patrones definidos que se tiene en la Red Neuronal.
Tamaño del Grid (X*Y)	Representa a la dimensión de la matriz de correspondiente a los patrones definidos, cuyos valores representan a los enlaces entre las neuronas de capa de entrada y las neuronas de la capa de salida.

En los anexos 7 y 8 se muestra los archivos correspondientes a la Red Neuronal Entrenada de Dígitos y la Red Neuronal Entrenada de Letras utilizadas durante las pruebas realizadas a los subsistemas Recam y Neuron.

3.7 Dimensiones de la capa de entrada de la Red Neuronal

Una de las tareas imprescindibles durante la construcción del prototipo del sistema ha sido hallar el número de neuronas que conforman la Capa de Entrada de la Red Neuronal, y su correspondiente distribución bidimensional.
Como se explico en el subcapítulo [3.4.1.8.1] - diseño de la Red Neuronal, debido a la complejidad de las tareas de aprendizaje y entrenamiento, se han implementado dos redes neuronales, una Red Neuronal de Dígitos y una Red Neuronal de Letras, las mismas que son independientes una de la otra.

3.7.1 Red Neuronal de Dígitos:

Los patrones reconocidos y entrenados por Red Neuronal de Números son los dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

Para todos estos dígitos, antes de definir la cantidad de neuronas que conforman la Capa de Entrada de la Red Neuronal de Números, así como su adecuada distribución bidimensional, por cada digito se ha tomado como muestra cinco (05) imágenes, las mismas que fueron extraídas de los Formularios de Trabajo, y por cada muestra se analizo la cantidad de píxeles que la conformaban (a lo ancho y a lo alto).

Los resultados se muestran en la siguiente tabla:

Tabla 3.19. Distribución de píxeles de las imágenes de los diez dígitos vienen

Digito	Num. de Píxeles Ancho	Num. de Píxeles Alto
0	17	14
0	17	20
0	18	19
0	20	17
0	22	18
1	10	16
1	10	19
1	11	18
1	13	19
1	16	25
2	16	19
2	17	19
2	17	21
2	18	22
2	21	22
3	16	21
3	16	26
3	17	22
3	18	17
3	20	23
4	16	24
4	17	22
4	17	24
4	17	24
4	22	25
5	16	21
5	16	22
5	16	23
5	17	23
5	18	23
6	11	20
6	13	19
6	16	19
6	16	24
6	20	21
7	16	24
7	17	21
7	19	22
7	22	21
7	17	21
8	14	19
8	14	21
8	16	23
8	17	21
8	17	21
9	13	25
9	14	13
9	14	23
9	15	24
9	16	21

Van

De la Tabla 3.19, se derivan las siguientes estadísticas: Tablas: 3.20 y 3.21 :

Tabla 3.20 Resultados provenientes de la Tabla 3.19

	Num. de Píxeles Ancho	Num. de Píxeles Alto
Máximo	22	26
Minimo	10	13
Promedio	16.38	21.02

Tabla 3.21 Distribución de Frecuencias de los píxeles del alto y ancho de las imágenes de los dígitos

Ancho	Frec
10	2
11	2
12	0
13	3
14	4
15	1
16	13
17	13
18	4
19	1
20	3
21	1
22	3

Alto	Frec.
13	1
14	1
15	0
16	1
17	2
18	2
19	8
20	2
21	11
22	6
23	6
24	6
25	3
26	1

De las estadísticas mostradas en las Tablas 3.20 y 3.21, el valor promedio en cuanto al ancho es 16.38 píxeles y el de mayor frecuencia es de 16 o 17 píxeles, mientras que el valor mínimo es 10 píxeles, el valor máximo es de 22 píxeles por lo tanto, si se optaría por considerar el valor promedio o el de mayor frecuencia, se perderían algunas características de las imágenes de un número considerable de muestras, para este caso de 12 imágenes.

De manera similar, en cuanto al alto, el valor promedio es 21.02 píxeles y el de mayor frecuencia es de 21 píxeles, mientras que el valor mínimo es 13 píxeles, el valor máximo es de 26 píxeles por lo tanto, si se optaría por considerar el valor promedio o el de mayor frecuencia, se perderían algunas características de las imágenes de un número considerable de muestras, para este caso, son 22 imágenes.

De los resultados encontrados se puede deducir que el valor ideal, que permite conservar las características de las imágenes de los dígitos, esta representado por las dimensiones máximas de alto y ancho (22 píxeles en el caso del ancho y 26 píxeles en el caso del alto), por lo tanto, la capa de entrada de la Red Neuronal estará conformada por 572 neuronas, con una distribución bidimensionalmente de 22 x 26.

3.7.2 Red Neuronal de Letras:

Los patrones reconocidos y entrenados por Red Neuronal de Letras son las letras mayúsculas del Alfabeto: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y y Z.

Para todas estas letras del Alfabeto, antes de definir la cantidad de neuronas que conforman la Capa de Entrada de la Red Neuronal de Letras, así como su adecuada distribución bidimensional, se ha tomado en forma aleatoria un conjunto de 130 muestras de imágenes correspondientes a diferentes letras del Alfabeto, las mismas que fueron extraídas de los Formularios de Trabajo, y por cada muestra se analizo la cantidad de píxeles que la conformaban (a lo ancho y a lo alto). Los resultados se muestran en la siguiente Tabla:

Tabla 3.22. Distribución de píxeles de las imágenes de las 26 letras del Alfabeto

vienen

Letra	Num. de Píxeles Ancho	Num. de Píxeles Alto
A	15	20
A	17	22
A	19	23
A	16	24
A	17	22
A	15	18
A	21	21
A	18	20
A	17	19
A	18	18
A	15	20
A	17	20
A	15	20
A	17	21
B	16	20
B	15	21
B	19	23
C	17	21
C	18	20
C	14	18
C	14	15
C	16	20
C	17	18

Letra	Num. de Píxeles Ancho	Num. de Píxeles Alto
E	20	18
E	17	19
E	20	20
E	18	20
F	18	19
G	21	20
G	19	21
G	17	23
G	17	20
H	19	24
H	18	23
I	18	19
I	16	24
I	14	21
I	13	21
I	18	21
I	16	22
I	19	22
I	14	19
I	15	19
I	14	19
I	18	18
I	15	20

van

Letra	Num. de Píxeles Ancho	Num. de Píxeles Alto
C	15	20
C	15	21
D	15	18
D	21	18
D	17	18
D	17	19
D	21	19
E	18	18
E	18	19
E	17	19
E	18	20
E	19	17
E	20	18
E	14	19
E	21	19
E	19	19
E	17	19
E	18	20
E	19	17
N	19	22
N	20	20
N	19	20
N	21	21
N	23	19
N	20	19
N	18	19
N	19	21
O	18	21
O	18	20
O	17	20
O	16	17
O	21	20
O	18	21
O	18	17
O	17	18
O	17	18
O	19	17
O	16	20
P	16	17
P	17	19
P	16	19
R	17	20

van

vienen

Letra	Num. de Píxeles Ancho	Num. de Píxeles Alto
I	14	21
I	15	19
I	19	20
I	14	23
J	13	18
J	14	20
K	22	20
L	17	21
L	14	22
L	18	23
L	15	21
L	14	17
L	13	18
L	16	22
L	16	24
M	20	22
M	20	22
M	24	19
N	18	19
R	16	21
R	16	21
R	19	20
R	20	19
R	16	24
R	20	22
R	19	21
R	16	19
R	17	19
S	16	18
S	17	19
S	16	21
S	14	19
S	15	20
S	14	18
S	16	20
T	16	18
T	14	19
U	15	20
U	17	21
V	17	19
Y	13	22
Z	19	24

De la Tabla 3.22, se derivan las siguientes estadísticas: Tablas 3.23 y 3.24 :

Tabla 3.23 Resultados provenientes de la Tabla 3.22

	Num. de Píxeles Ancho	Num. de Píxeles Alto
Máximo	24	24
Mínimo	13	15
Promedio	17.153846	19.946154

Tabla 3.24 Distribución de Frecuencias de los píxeles del alto y ancho de las imágenes de las letras del Alfabeto

Ancho	*Frec.*
13	4
14	13
15	14
16	19
17	24
18	20
19	17
20	9
21	7
22	1
23	1
24	1

Alto	*Frec.*
15	1
16	0
17	7
18	18
19	30
20	30
21	21
22	11
23	6
24	6

De las estadísticas mostradas en las Tablas 3.23 y 3.24, el valor promedio en cuanto al ancho es 17.15 píxeles y el de mayor frecuencia es de 17 píxeles, mientras que el valor mínimo es 13 píxeles, el valor máximo es de 24 píxeles por lo tanto, si se optaría por considerar el valor promedio o el de mayor frecuencia, se perderían algunas características de las imágenes de un número considerable de muestras, para este caso de 56 imágenes.

De manera similar, en cuanto al alto, el valor promedio es 19.94 píxeles y el de mayor frecuencia es de 19 o 20 píxeles, mientras que el valor mínimo es 15 píxeles, el valor máximo es de 24 píxeles por lo tanto, si se optaría por considerar el valor promedio o el de mayor frecuencia, se perderían algunas características de las imágenes de un número considerable de muestras, para este caso, son 44 imágenes.

De los resultados encontrados se puede deducir que el valor adecuado, que permite conservar las características de las imágenes de las 26 letras del alfabeto, esta representado por las dimensiones máximas de alto y ancho (24 píxeles en el caso del ancho y 24 píxeles en el caso del alto), por lo tanto, la capa de entrada de la Red Neuronal estará conformada por 576 neuronas, con una distribución bidimensionalmente de 24 x 24.

CAPITULO IV

PRUEBAS DE SOFTWARE

Las pruebas, son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Existen varias estrategias de prueba las cuales deben ser utilizadas según sea el caso, para probar el funcionamiento del prototipo desarrollado en la presente investigación se ha elegido la prueba basada en componentes, entendiéndose por esta, como una parte del sistema que puede aislarse para la prueba; así un componente puede ser un objeto, un grupo de objetos, o uno o más subsistemas.

Dentro de las estrategias de pruebas basadas en componentes mas comunes se encuentran en la prueba de Unidad y la prueba de Integración:

La prueba de Unidad centra el proceso de verificación en la menor unidad de diseño del software: el componente software o módulo.

La prueba de Integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la integración. El objetivo es coger los módulos probados mediante la prueba de unidad y construir una estructura del programa que esté de acuerdo con lo que dista el sistema.

4.1. Pruebas de Unidad

En primera instancia se ha tratado las Pruebas de Unidad las cuales han permitido la verificación de los principales módulos de los Subsistemas

RECAM y NEURON, para lo cual se ha utilizado la siguiente plantilla para definir cada caso de Prueba:

Tabla 4.1. Plantilla para definir un Caso de Prueba

Atributo	Descripción
Nombre	Nombre del Caso de Prueba.
Entrada	Datos de entrada.
Oráculo	Resultados esperados de la prueba contra los que se compara la salida de la prueba.
Bitácora	Salida producida por la prueba.
Resultado de la Prueba.	Correcto o Incorrecto.

A) Caso de Prueba Crear Bloque :

a.1) Plantilla Caso de Prueba: Crear Bloque

Tabla 4.2. Plantilla para definir el Caso de Prueba Crear Bloque

Atributo	Descripción
Nombre	Crear Bloque
Entrada	Coordenadas del Bloques[n+1] a crear, Lista de Bloques, nombre, tipo, Tipo de Rejilla del Bloques[n+1].
Oráculo	Lista de Bloques + Bloques[n+1], Bloques[n+1](nombre)=nombre, Bloques[n+1](tipo) = tipo, Bloques[n+1](tipoRejilla) = tipoRejilla.
Bitácora	Lista de Bloques + Bloques[n+1], Bloques[n+1](nombre)=nombre, Bloques[n+1](tipo) = tipo, Bloques[n+1](tipoRejilla) = tipoRejilla.
Resultado	CORRECTO

a.2) Código de Prueba "Crear Bloque":

Tabla 4.3. Código Fuente correspondiente a la Prueba Crear Bloque

```
Privae Sub Picture2_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
 If Pintar Then
  If Seleccionando And Button = 1 Then
    Seleccionando = False
    Pintar = False
    Bloques(0, num) = xIni
    Bloques(1, num) = yIni
    Bloques(2, num) = xAnt
   Bloques(3, num) = yAnt
Picture2.Line (Bloques(0, num), Bloques(1, num))-(Bloques(2, num), Bloques(3, num)), , B
Picture2.DrawStyle = vbSolid
 Picture2.Line (Bloques(0, num), Bloques(1, num))-(Bloques(2, num), Bloques(3, num)),
 COLOR_ROJO, B
    Picture2.DrawStyle = vbDot
```

```
    num = num + 1
    Text1.Text = Str(num)
    If Text1.Text <> "" Then
       DesBloques(0, num) = Text1.Text
    Else
       DesBloques(0, num) = "noname"
    End If
    If Option1.Value = True Then
       DesBloques(2, num) = "1"      'rejilla
    ElseIf Option2 = True Then
          DesBloques(2, num) = "2"      'subdivisiones
       Else
          DesBloques(2, num) = "3"      'rectangular
    End If

    ReDim Preserve Bloques(3, num)   'Reservando Memoria para el siguiente Bloque
    xAnt = 0
  End If
 End If
End Sub
```

a.3) Resultados :

Al ejecutar el código de prueba anterior, se ha podido comprobar que cada vez que se crea un nuevo Bloque, este se añade al final de la Lista de Bloques, por lo cual el número de Bloques se incrementa en una unidad, y se actualiza toda la información.

Bloques[n+1](nombre)=nombre,
Bloques[n+1](tipo)=tipo,
Bloques[n+1] (tipoRejilla) = tipoRejilla.

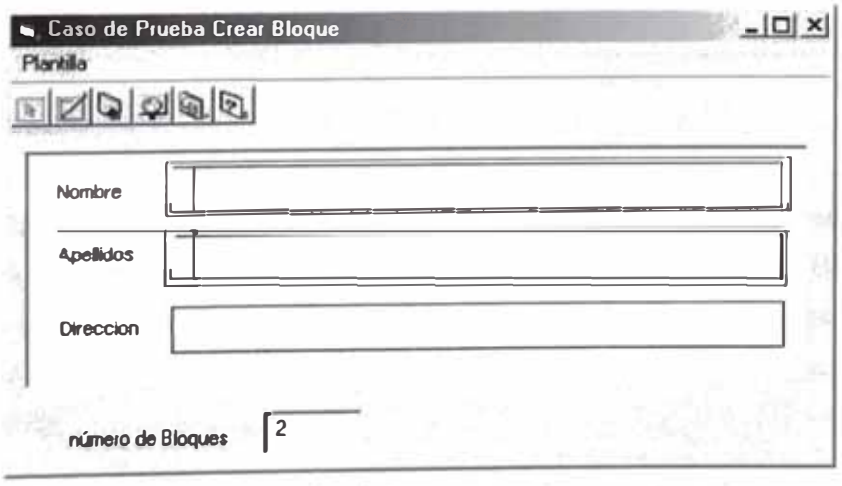


Figura 4.1 Formulario para el Caso de Prueba Crear Bloque

B) Caso de Prueba Eliminar Bloque :

b.1) Plantilla Caso de Prueba: Eliminar Bloque

Tabla 4.4 Plantilla para definir el Caso de Prueba Eliminar Bloque

Atributo	Descripción
Nombre	Eliminar Bloque
Entrada	Coordenadas del Bloques[i] a eliminar, Lista de Bloques.
Oráculo	Lista de Bloques – Bloques[i]
Bitácora	Lista de Bloques – Bloques[i]
Resultado	CORRECTO

b.2) Código de Prueba "Eliminar Bloque":

Tabla 4.5. Código Fuente correspondiente a la Prueba Eliminar Bloque

```
Sub EliminarBloque(NumAux As Integer)
   Dim temp As Integer
   If num >= 1 Then
      Picture2.DrawStyle = vbSolid
      Picture2.Line (Bloques(0, NumAux), Bloques(1, NumAux))-(Bloques(2, NumAux), Bloques(3,
                  NumAux)), COLOR_ROJO, B
      Picture2.DrawStyle = vbDot
      Bloques(0, NumAux) = 0
      Bloques(1, NumAux) = 0
      Bloques(2, NumAux) = 0
      Bloques(3, NumAux) = 0
      DesBloques(0, NumAux) = " "
      DesBloques(1, NumAux) = " "
      num = num - 1
      numSel = -1
      ReDim Preserve Bloques(3, num)
      ReDim Preserve DesBloques(1, num)
   End If
End Sub
```

b.3) Resultados :

Al ejecutar el código de prueba anterior, se pudo comprobar que antes de proceder a eliminar un Bloque se verifica el valor del número de Bloques de la Lista de Bloques, si éste es mayor o igual a la Unidad se procede a eliminar Bloques[i] seleccionado, en caso contrario se cancela la operación.

Así mismo en este caso de prueba también se puede aplicar "***la prueba de frontera***", es decir seleccionar los elementos extremos de la Lista de Bloques, el primer y último elemento de la Lista de Bloques.

b.4) Plantilla Caso de Prueba: Eliminar Primer Bloque de la Lista

Tabla 4.6 Plantilla para definir el Caso de Prueba Eliminar Primer Bloque de la Lista

Atributo	Descripción
Nombre	Eliminar primer Bloque de la Lista
Entrada	Coordenadas del Bloques[0] a eliminar, Lista de Bloques.
Oráculo	Lista de Bloques – Bloques[i]
Bitácora	Lista de Bloques – Bloques[i]
Resultado	CORRECTO

b.5) Código de Prueba "Eliminar Primer Bloque":

Tabla 4.7 Código Fuente correspondiente a la Prueba Eliminar Primer Bloque

```
Sub EliminarBloque(NumAux As Integer)
   Dim temp As Integer
   If num >= 1 Then
     Picture2.DrawStyle = vbSolid
     Picture2.Line (Bloques(0, NumAux), Bloques(1, NumAux))-(Bloques(2, NumAux), Bloques(3,
                    NumAux)), COLOR_ROJO, B
     Picture2.DrawStyle = vbDot
     Bloques(0, NumAux) = 0
     Bloques(1, NumAux) = 0
     Bloques(2, NumAux) = 0
     Bloques(3, NumAux) = 0
     DesBloques(0, NumAux) = " "
     DesBloques(1, NumAux) = " "

     If NumAux < num - 1 Then

       For temp = NumAux To num - 1
         Bloques(0, temp) = Bloques(0, temp + 1)
         Bloques(1, temp) = Bloques(1, temp + 1)
         Bloques(2, temp) = Bloques(2, temp + 1)
         Bloques(3, temp) = Bloques(3, temp + 1)
         DesBloques(0, temp) = Bloques(0, temp + 1)
         DesBloques(1, temp) = Bloques(1, temp + 1)
       Next
     End If
     num = num – 1
     numSel = -1
  End If
End Sub
```

b.6) Resultados :

Al ejecutar el código de prueba, se pudo comprobar que cuando se elimina el Primer Bloque de la Lista de Bloques, el segundo pasa a ser el primero, el tercero pasa a ser el segundo y así sucesivamente, liberándose el espacio de memoria que se había reservado para este último.

b.7) Plantilla Caso de Prueba: Eliminar Último Bloque de la Lista

Tabla 4.8 Plantilla para definir el Caso de Prueba Eliminar último Bloque de la Lista

Atributo	Descripción
Nombre	Eliminar último Bloque de la Lista
Entrada	Coordenadas del Bloques[n] a eliminar, Lista de Bloques.
Oráculo	Lista de Bloques – Bloques[n]
Bitácora	Lista de Bloques – Bloques[n]
Resultado	CORRECTO

b.8) Resultados :

Al ejecutar el código de prueba "Eliminar Bloque", mostrado en la Cuadro 4.7, se ha podido comprobar que cuando se elimina el Ultimo Bloque de la Lista de Bloques, solo se procede a liberar el espacio de memoria que se le había reservado.

C) Caso de Prueba Filtrar Ruido :

c.1) Plantilla Caso de Prueba: Filtrar Ruido :

Tabla 4.9 Plantilla para definir el Caso de Prueba Filtrar Ruido

Atributo	Descripción
Nombre	Filtrar Ruido
Entrada	Imagen del Caracter con puntos y/o manchas alrededor.
Oráculo	Imagen del Caracter sin ruido.
Bitácora	Imagen del Caracter sin ruido
Resultado	CORRECTO

c.2) Código de Prueba "Filtrar Ruido":

Tabla 4.10 Código Fuente correspondiente a la Prueba Filtrar Ruido

```
Public Sub eliminaruido(lx2 as integer, lx1 as integer, ly1 as integer, ly1 as integer)
        ancho = lx2 - lx1
        alto = ly2 - ly1
        Erase Ruido
        ReDim Ruido(alto, ancho)
        wnum = 1
        For fi = 0 To alto
           For co = 0 To ancho
              If Picture1.Point(co + lx1, fi + ly1) = 0 Then ' si esta pintado
                'Picture1.PSet (co + lx1, fi + ly1), QBColor(14)
                If (fi = 0 And co = 0) Then
                  a = 0: B = 0: c = 0: d = 0
                ElseIf fi = 0 Then
                  a = Ruido(fi, co - 1): B = 0: c = 0: d = 0
```

```
            ElseIf co = 0 Then
              a = 0: B = 0: c = Ruido(fi - 1, co): d = Ruido(fi - 1, co + 1)
            ElseIf co = ancho Then
              a = Ruido(fi, co - 1): B = Ruido(fi - 1, co - 1): c = Ruido(fi - 1, co): d = 0
            Else: a = Ruido(fi, co - 1): B = Ruido(fi - 1, co - 1): c = Ruido(fi - 1, co):
                  d = Ruido(fi - 1, co + 1)
            End If

            If (a = 0 And B = 0 And c = 0 And d = 0) Then
               Ruido(fi, co) = wnum
               wnum = wnum + 1
            ElseIf (a <> 0 And d <> 0 And a <> d) Then
             If a < d Then
               Ruido(fi, co) = a 'remplazar todos los d por a
               For xfi = 0 To alto 'fi
                 For xco = 0 To ancho 'co
                   If Ruido(xfi, xco) = d Then Ruido(xfi, xco) = a
                 Next xco
               Next xfi
             Else 'remplazar todos los a por d
               Ruido(fi, co) = d
               For xfi = 0 To alto 'fi
                 For xco = 0 To ancho 'co
                   If Ruido(xfi, xco) = a Then Ruido(xfi, xco) = d
                 Next xco
               Next xfi
             End If
            ElseIf (B <> 0 And d <> 0 And B <> d) Then
             If B < d Then
               Ruido(fi, co) = B 'remplazar todos los d por b
               For xfi = 0 To fi
                 For xco = 0 To co
                   If Ruido(xfi, xco) = d Then Ruido(xfi, xco) = B
                 Next xco
               Next xfi
             Else 'remplazar todos los b por d
               Ruido(fi, co) = d
               For xfi = 0 To fi
                 For xco = 0 To co
                   If Ruido(xfi, xco) = B Then Ruido(xfi, xco) = d
                 Next xco
               Next xfi
             End If
            ElseIf a <> 0 Then
              Ruido(fi, co) = a
            ElseIf B <> 0 Then
              Ruido(fi, co) = B
            ElseIf c <> 0 Then
              Ruido(fi, co) = c
            ElseIf d <> 0 Then
               Ruido(fi, co) = d
            End If
          End If
        Next co
      Next fi
      ReDim Cont(wnum)
      'obtine las frecuencias acumuladas
      For fi = 0 To alto
        For co = 0 To ancho
          If Ruido(fi, co) >= 1 Then
            Cont(Ruido(fi, co)) = Cont(Ruido(fi, co)) + 1
          End If
        Next co
      Next fi
      'obtiene el mayor
      mayor = Cont(0)
      mayor_i = 0
```

```
        For xi = 0 To wnum
          If Cont(xi) > mayor Then
            mayor = Cont(xi)
            mayor_i = xi
          End If
        Next xi
        'elimina los menores
        For fi = 0 To alto
          For co = 0 To ancho
            If (Ruido(fi, co) <> mayor_i) Then 'aqui es <>
              Picture1.PSet (co + lx1, fi + ly1), QBColor(15)
            End If
          Next co
        Next fi
      End If 'de filtrar ruido
End Sub
```

c.3) Resultados :

Al ejecutar el código de prueba "Filtrar Ruido", se ha comprobado que se eliminan todos los puntos aislados y/o manchas que pudieran haber alrededor de la imagen del caracter a reconocer.

Dichos resultados se muestran a continuación, con el siguiente ejemplo demostrativo, en el cual se va someter a la imagen mostrada en la figura.4.2 al código de prueba Filtrar Ruido:

Figura 4.2 Imagen de un Campo de un Formulario con Ruido

Figura 4.3 Imagen de un Campo de un Formulario sin Ruido

Comparando las imágenes mostradas en las figuras 4.2 y 4.3, se puede apreciar que esta ha mejorado, puesto que han desaparecido los puntos y/o manchas que se encontraban en la figura 4.2.

D) Caso de Prueba Segmentación de Imágenes :

d.1) Plantilla Caso de Prueba: Segmentación de Imágenes :

Tabla 4.11 Plantilla para definir el Caso de Prueba Segmentación de Imágenes

Atributo	Descripción
Nombre	Segmentación de Imágenes.
Entrada	Imagen del Campo de un Formulario.
Oráculo	Conjunto de Imágenes de caracteres aislados.
Bitácora	Conjunto de Imágenes de caracteres aislados.
Resultado	CORRECTO

d.2) Resultados :

Antes de mostrar los resultados obtenidos al aplicar el código de prueba "Segmentación de Imágenes", se procederá mostrar diferentes ejemplos demostrativos, mediante los cuales se van someter a determinadas imágenes que representan a campos de los formularios a reconocer, al código de prueba Segmentación de Imágenes:

1. **Imagen de un campo a reconocer por medio de un conjunto de letras continúas.**

   En la figura 4.4, se muestra la imagen de un campo de un Formulario, nótese que las letras no están previamente separadas, sino que han sido escritas en forma continua por el usuario.

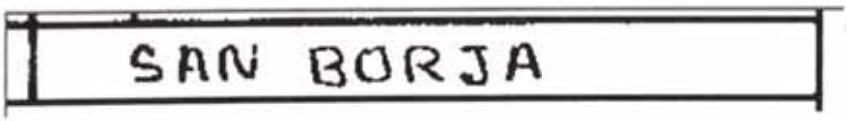


Figura 4.4 Imagen de un Campo de un Formulario

Luego de haber sido sometida la imagen representada en la figura 4.4, al código de prueba "Segmentación de imágenes", se ha obtenido los siguientes resultados (figura 4.5).

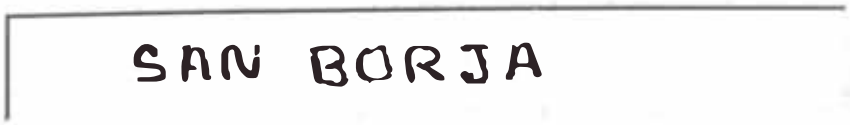


Figura 4.5. Conjunto de imágenes de los caracteres aislados, luego de haber sometido a la imagen de la figura 4.4 al código de prueba "Segmentación de Imágenes"

2. **Imagen de un campo a reconocer, donde cada letra encaja en una rejilla.**

En la figura 4.6, se muestra la imagen de un campo de un Formulario, nótese que los dígitos que la componen están separados unos de otros por medio de rejillas.

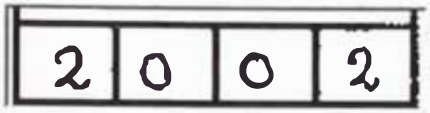


Figura. 4.6 Imagen de un Campo de un Formulario con Rejilla

Luego de haber sido sometida la imagen representada en la figura 4.6, al código de prueba "Segmentación de imágenes", se ha obtenido el siguiente resultado.

Figura 4.7 Conjunto de imágenes de los caracteres aislados, luego de haber sometido a la imagen de la figura 4.6 al código de prueba "Segmentación de Imágenes"

Nótese que para este caso incluso el algoritmo de segmentación ha eliminado las líneas de las rejillas, que enmarcaban a cada carácter de la imagen de la figura 4.6.

3. **Imagen de un campo a reconocer, donde cada letra esta separada por medio de líneas divisorias.**

En la figura 4.8, se muestra la imagen de un campo de un Formulario, nótese que los dígitos que la componen están separados unos de otros por medio de líneas divisorias.

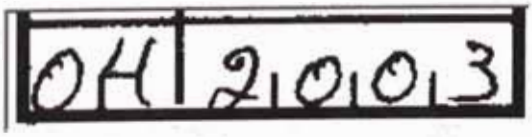


Figura 4.8 Imagen de un campo de un formulario con líneas divisorias

Luego de haber sido sometida la imagen representada en la figura 4.8, al código de prueba "Segmentación de imágenes", se ha obtenido el siguiente resultado.

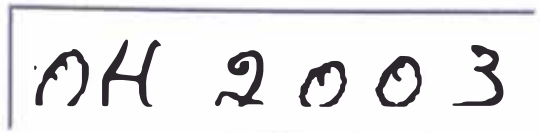


Figura 4.9. Conjunto de imágenes de los Caracteres Aislados

Observación: El problema es cuando la imagen del carácter se une con el borde inferior, lo cual provoca una confusión, y la imagen es mutilada en la parte inferior perdiendo algunas características, lo cual podría generar errores en la etapa del reconocimiento.

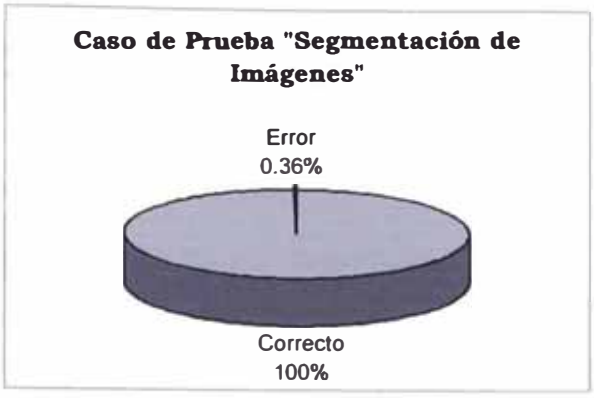
Nótese también que, cuando la imagen del carácter esta muy apegada a una línea divisoria (el carácter 4), ésta se convierte en parte del carácter, lo cual definitivamente altera las características de la imagen y por consiguiente provocará errores en la etapa de reconocimiento.

En general, durante esta etapa de prueba del código "Segmentación de Imágenes", se ha trabajado con los diez (10) Formularios de Muestra, los mismos que se presentan en el Anexo 3, y para cada uno de ellos se ha obtenido los siguientes resultados.

Tabla 4.12 Resultados Obtenidos en el Proceso de Segmentación a una muestra de diez formularios

N° Formulario	N° de campos	N° de Imágenes Segmentadas.	N° de Errores
01	10	78	0
02	10	81	0
03	10	78	1
04	10	77	0
05	10	81	0
06	10	85	1
07	10	79	0
08	10	88	1
09	10	87	0
10	10	89	0
	Total	823	3

Tabla 4.13 Resultados Obtenidos en el Proceso de Segmentación



Al analizar cada uno de los errores, se ha podido determinar que el 100% de los errores se debe a "**errores de escritura**", puesto que la imagen correspondiente a algún carácter de un campo se encontraba muy apegada al borde inferior, o a alguna línea divisoria, tal como se presento en el ejemplo de la figura 4.9.

E) Caso de Prueba Preprocesamiento de Imágenes :

e.1) Plantilla Caso de Prueba: Preprocesamiento de Imágenes:

Tabla 4.14 Plantilla para definir el Caso de Prueba Preprocesamiento de Imágenes

Atributo	Descripción
Nombre	Preprocesamiento de Imágenes.
Entrada	Imagen de un carácter
Oráculo	Imagen de un carácter preparada para ser reconocida, es decir sin zonas en blanco alrededor de ella.
Bitácora	Imagen de un carácter preparada para ser reconocida, es decir sin zonas en blanco alrededor de ella.
Resultado	CORRECTO

e.2) Código de Prueba "Preprocesamiento de Imágenes":

Tabla 4.15 Código Fuente de la Prueba Preprocesamiento de Imágenes

```
Sub Preprocesar( )
' Encontrando los cuatro bordes de la imagen scaneada
 Lado_sup = 0: Lado_Inf = 0: Lado_Izq = 0: Lado_Der = 0
 temp1 = False
 For X = 0 To Picture5.ScaleWidth Step Acc  ' Margen Izquierdo
   For Y = 0 To Picture5.ScaleHeight Step Acc
     If 0 = Picture5.Point(X, Y) Then temp1 = True: Exit For
   Next Y
   If temp1 Then Lado_Izq = X - Acc: Exit For
 Next X
```

```
If Not (temp1) Then
      MsgBox "La imagen esta en blanco", 48, "Preprocesamiento"
      Exit Sub
  End If
  temp1 = False
  For X = Picture5.ScaleWidth To Lado_Izq Step -Acc 'Margen derecho
    For Y = 0 To Picture5.ScaleHeight Step Acc
      If 0 = Picture5.Point(X, Y) Then temp1 = True: Exit For
    Next Y
    If temp1 Then Lado_Der = X + Acc: Exit For
  Next X
  temp1 = False
  For Y = 0 To Picture5.ScaleHeight Step Acc ' Margen Superior
    For X = Lado_Izq To Lado_Der Step Acc
      If 0 = Picture5.Point(X, Y) Then temp1 = True: Exit For
    Next X
    If temp1 Then Lado_sup = Y - Acc: Exit For
  Next Y
  temp1 = False
  For Y = Picture5.ScaleHeight To Lado_sup Step -Acc ' Margen Inferior
    For X = Lado_Izq To Lado_Der Step Acc
      If 0 = Picture5.Point(X, Y) Then temp1 = True: Exit For
    Next X
    If temp1 Then Lado_Inf = Y + Acc: Exit For
  Next Y
End Sub
```

e.3) Resultados :

Al ejecutar el código de prueba, se ha podido comprobar que al preprocesar una imagen de un caracter a reconocer, esta termina recortada de las posibles zonas en blanco que rodean a la imagen misma, de tal manera que en la etapa de Reconocimiento se tenga en cuenta solamente la zona de interés (zona enmarcada con líneas entrecortadas de la figura 4.10).

Para entenderlo mejor, veamos a continuación un ejemplo demostrativo :

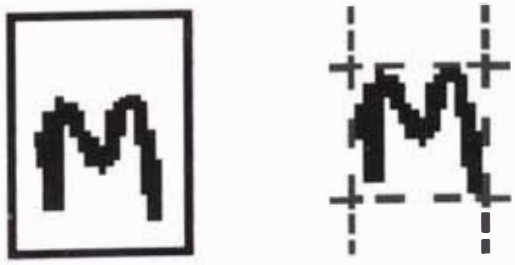


Figura. 4.10 Izquierda. Imagen original del caracter "M", Derecha. Imagen Preprocesada preparada para ser reconocida.

F) Caso de Prueba Reconocimiento de Imágenes :

f.1) Plantilla Caso de Prueba: Reconocimiento de Imágenes:

Tabla 4.16 Plantilla para definir el Caso de Prueba Reconocimiento de Imágenes

Atributo	Descripción
Nombre	Reconocimiento de Imágenes.
Entrada	Imagen de un Caracter.
Oráculo	Caracter reconocido correctamente.
Bitácora	La imagen de carácter corresponde a la imagen del patrón ganador de la Red Neuronal.
Resultado	CORRECTO

f.2) Código de Prueba "Reconocimiento de Imágenes":

Tabla 4.17 Código Fuente de la Prueba Reconocimiento de Imágenes

```
For test = 1 To Digitos(0)
   result = 0: max = 0
   For Index = 1 To GridX * GridY
     temp = Val(Grids((test - 1) * GridX * GridY + Index))
     If Abs(temp) > max Then max = Abs(temp)
     If Grid(Index - 1) = 0 Then temp = temp * (-1)
     result = result + temp
   Next Index
If max > 0 Then result = Int((100 * result / (max * GridX * GridY)) + 0.5)
   temp2 = 0
For Index = 0 To List2.ListCount - 1
     Text = List2.List(Index)
     Porcentaje = Val(Right(Text, Len(Text) - InStr(Text, Chr(9))))
     If Porcentaje > result Then        temp2 = temp2 + 1
     Else
       Exit For
     End If
   Next Index
   List2.AddItem (Digitos(test) + Chr(9) + Str(result) + "%"), temp2
 Next test
```

f.3) Resultados :

Se ha ejecutado el código de prueba, con una muestra de quince imágenes de caracteres aislados tomados al azar, además se ha utilizado una Red Neuronal con las siguientes características:

- La Capa de Entrada de la Red Neuronal conformada por 576 neuronas, con una distribución bidimensionalmente de 24 x 24.
- Número de Entrenamientos : 13
- Profundidad de Entrenamiento : 5

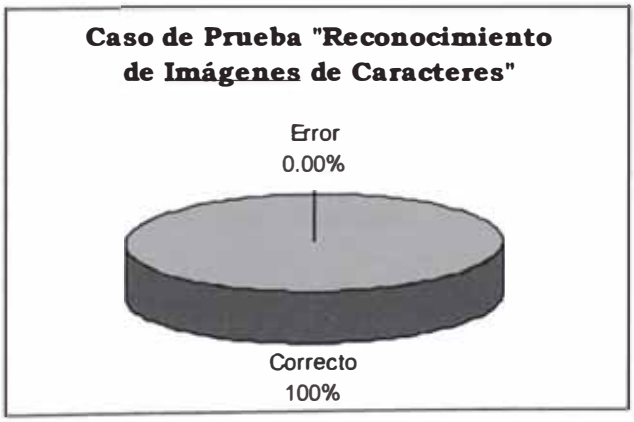
Obteniéndose los siguientes resultados :

Tabla 4.18 Resultados Obtenidos en el Proceso de Reconocimiento

Caracter a reconocer	Caracter Reconocido	% de Similitud
	A	58%
	U	52%
	N	49%
	E	48%
	Q	53%
	I	57%
	R	63%
	A	61%
	V	65%
	E	70%
	D	68%
	T	80%
	A	64%
	P	62%
	Z	67%

Con lo cual se ha concluido que se ha obtenido un 0% de Error en la Fase de Reconocimiento de imágenes de caracteres aislados.

Tabla 4.19 Resultados Obtenidos en el Proceso de Reconocimiento



G) Caso de Prueba Entrenamiento de la Red Neuronal :

g.1) Plantilla Caso de Prueba: Entrenamiento de la Red Neuronal

Tabla 4.20 Plantilla para definir el Caso de Prueba entrenamiento de la RN

Atributo	Descripción
Nombre	Entrenamiento de la Red Neuronal.
Entrada	Patrón a entrenar de la Red neuronal.
Oráculo	Patrón entrenado de la Red neuronal.
Bitácora	Patrón entrenado de la Red neuronal.
Resultado	CORRECTO

g.2) Código de Prueba "Entrenamiento de la Red Neuronal"

Tabla 4.21 Código Fuente de la Prueba Entrenamiento de la Red Neuronal

```
Private Sub AprenderChar(Letra)
  profundidad = 5
  For i = 1 To GridX * GridY
    If (Grid(i - 1) = 0) Then   ' Verificando el peso de la Neurona de la Capa de Entrada
       delta = -1
    Else
       delta = 1
    End If
    val1 = Grids((temp1 - 1) * GridX * GridY + i)
    If Abs(val1 + delta) > profundidad Then delta = 0
    Grids((temp1 - 1) * GridX * GridY + i) = val1 + delta
  Next i
End Sub
```

g.3) Resultados :

Al ejecutar el código de prueba, se ha comprobado que al entrenar la Red Neuronal, el patrón al que se le aplica el algoritmo de aprendizaje, sufre alteraciones en su estructura, pronunciándose más intensamente aquellos puntos que son coincidentes entre patrones.

En la siguiente figura se muestra paso a paso los cambios efectuados sobre el patrón "A" de la red Neuronal.

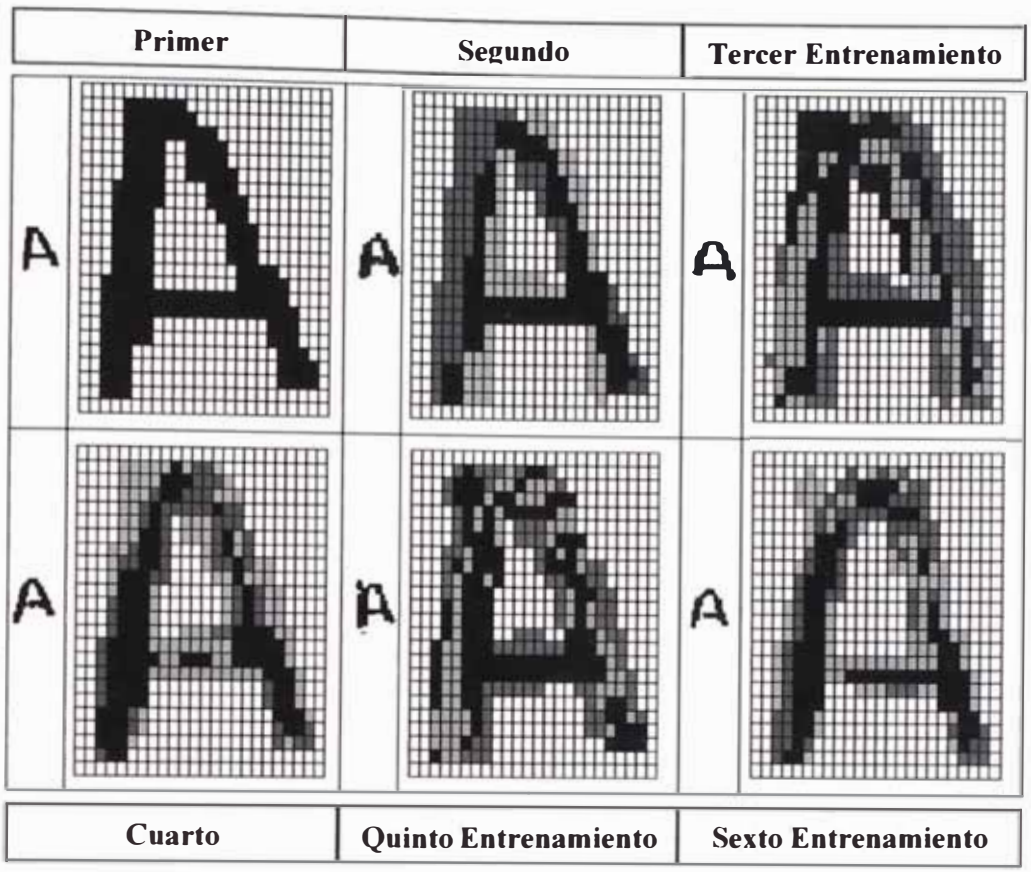


Figura 4.11 A la Izquierda, patrones entrenadores, a la derecha patrón "A", luego de ser entrenado.

4.2. Pruebas de Integración

Este subcapítulo contiene las pruebas realizadas al sistema en conjunto, para lo cual se ha procesado una muestra de diez Formularios de Trabajo, (Anexo 3), que corresponden a una "Encuesta de Ciencia, Tecnología e Innovación Tecnológica – 2000", aplicada por el Instituto Nacional de Investigación e Informática (INEI), los mismos que para este caso de Pruebas de Integración, han sido rellenados por un solo usuario y contienen información diversa como :

- Razón social de la empresa (razon_emp),
- Domicilio Fiscal :
  - Departamento (Depart),
  - Provincia (Provincia),
  - Distrito (Distrito),
  - Dirección – Av. O Urbanización (Dirección),
  - Manzana (M),

- Actividad económica principal de la empresa (Actividad).
- Gerente general (Ggeneral)
- Gerente de Producción (Gproduc)

Por lo tanto, toda esta información será seleccionada para ser reconocida por el Subsistema Recam, que almacenará los resultados en un archivo de extensión "Pla" (Anexo 4).

En cuanto a las características de cada uno de los formularios que conforman la muestra se puede mencionar:

- Del Formulario
  - Las diez muestras han sido escaneadas a una Resolución de 200 ppp
- De la Red Neuronal
  - La Capa de Entrada de la Red Neuronal de letras esta conformada por 576 neuronas con una distribución bidimensional de 24 x 24, tal como se sugirió en el apartado [3.7.2] Red Neuronal de Letras.
  - La Capa de Entrada de la Red Neuronal de Números, esta conformada por 572 neuronas con una distribución bidimensional de 22 x 26, tal como se sugirió en el apartado [3.7.1] Red Neuronal de Dígitos.
  - Con respecto al número de profundidad de entrenamiento, este varía entre 3 y 5 y el número de entrenamientos de la Red Neuronal también son diferentes para cada Red Neuronal.

Luego de haber aplicado sobre cada muestra siete Pruebas de Integración, los resultados obtenidos se presentan en las siguientes cuadros, donde las celdas sombreadas indican que se produjo al menos un error en el reconocimiento de caracteres aislados de un campo del Formulario de Trabajo.

Al final de este apartado se concluye analizando los posibles factores que provocaron estos errores, en este caso específicamente si los errores se generaron por el reconocimiento (E_Recon), o fue un error de escritura (E_Escrit).

DATOS PROCEDENTES DE LOS FORMULARIOS DE TRABAJO

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	# Caracteres
ESCUELA NUEVA	LIMA	LIMA	SJM	AREQUIPA		VENTA DE UTILES ESCOLARES	MATILDE GARZON	YENNY LINARES	78
EL ESTUDIANTE	LIMA	LIMA	COMAS	LOS JAZMINES		VENTA DE PIZARRAS ACRILICAS	RUBEN CARREON	DAVID DIAZ	81
COMPUTER REY	LIMA	LIMA	CERCADO	WILSON		VENTA DE COMPUTADORAS	JAVIER CONTRERAS	DANIELA RIVAS	78
SELVA NEGRA	LIMA	LIMA	SMP	SANTA ROSA		MATERIALES DE CONSTRUCCION	DANIEL CARO	WILLIAN DAMIAN	77
ELECTRONICA HENRY	LIMA	LIMA	SAN MIGUEL	LOS ROBLES	J	ELECTRONICA EN GENERAL	ENRIQUE PEREZ	DAMIAN	81
SERVICIOS CAMILA	LIMA	LIMA	CALLAO	LOS CLAVELES	J	SERVICIO DE FOTOCOPIAS	RICARDO JIMENEZ	MANUEL TITO	85
EL MARTILLO	LIMA	LIMA	SAN BORJA	LOS SAUCES		FABRICACION DE MUEBLES	SAMUEL GARCES	DANIEL ESPEJO	79
IMPRENTA VELOZ	LIMA	LIMA	SAN MARTIN	LOS DOMINICOS		IMPRESIONES EN GENERAL	HECTOR CARRASCO	CARMEN LOAYSA	88
SUPERMERCADO EL TIGRE	LIMA	LIMA	SAN MIGUEL	LAS CAMELIAS		INSUMOS ALIMENTICIOS	SARA RIQUEL	MANUEL VELEZ	87
EL RINCON PIURANO	LIMA	LIMA	CHORRILLOS	LOS ALAMOS	B	VENTA DE PLATOS TIPICOS	CARLOS GAMARRA	BEATRIZ SANTOS	89
								TOTAL	823

CARACTERES RECONOCIDOS POR RECAM

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	E. Recon	E. Escrit.
ESCUELA NUEVA	LIMA	LIMA	EIM	AREQLIPA		VENIA DE UTTLES ESCOLAGES	NAYILOE GARIGN	YENNY LIKAPZS	14	
ELESIOCIANTE	LINF	LIMA	COMAS	LOS JFZMINFS		VFNTA OE PIIARTAS ALRILILFS	RUBFN CAOKEON	DAVIO DIAY	19	1
COMPUJFR REY	LIMA	LIMA	CERCGDO	WTLFON		VENTA OF COMPUTAOORAS	JAVIER CONTRERAS	DAKICLA FIVAS	11	
SELVA NEGRA	LIMA	LINL	SM K	SANIA RDSA		MATERIALEG DE COUSIFUCCION	DANIEL LANU	WLLLIAN DANIAN	14	1
ELFLTRSNICA KENRY	LIMA	LIMA	SAN MIQUEC	LOS ROBLES	J	FLECTRLNICA EN GENCRHL	ENCIXVF FERFY	OAMIAU	20	
SEPVTCIOS EAMILA	LIMA	LIMA	CALLAO	LDS LLAVCCES	J	SERVILID CE FGIOLDFIAS	RILARDU JIMENEI	MANUFL JITO	20	
ELMARIILLO	LIMA	LIMA	SANGORJA	LDS JALLES		LABRICALIJN DE MUEBLES	KAMUFL GARLFS	DANJLL FGPEIO	18	2
IMPAEGYA VELOI	LIMF	LIMH	SDN MAPILN	LOS ODMINICOS		IMPFESIGNES FN OENYRAC	HKCIUR CARRASLO	CARMEN LOAYSA	22	
SUPERMLPCAOO EL TIORE	LIMA	LIMA	SAN MIGUEL	LAS CKMELIAS		TNSUMGS ALIMENJICIOS	SAGA RIOUFL	MAUUFL VELEZ	13	1
EL RINCON PIUFANO	LIMF	LIMK	CNGRMILLGS	LDS ALAMOS	B	VENJA DE PLATDS JIPILOS	CARLUS GAMARRA	EEATAIZ SAGTUS	17	
								TOTAL	168	5

Tabla 4.22 Resultados de la Prueba de Integración : Reconocimiento con Profundidad de Entrenamiento = 3 y Número de Entrenamientos = 8

Resumen de la Tabla 4.22

Total errores	173
Porcentaje de Error Reconocimiento	20.41
Porcentaje de Error Escritura	0.61
Total Porcentaje de Error	21.02
Total Porcentaje de Acierto	78.98

Tabla 4.23 Resumen de los resultados obtenidos por RECAM, con profundidad de entrenamiento = 3 y número de entrenamientos = 8

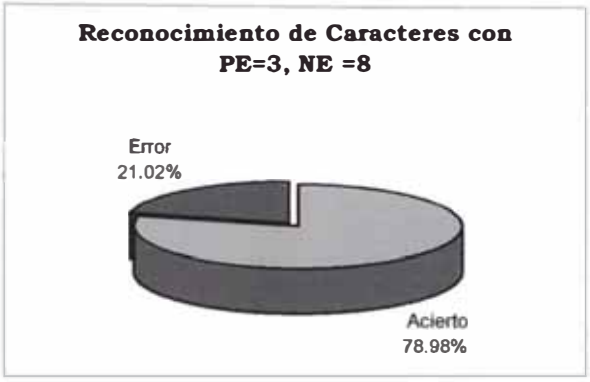


Tabla 4.24 Representación gráfica del total de porcentajes de Error y Acierto mostrados en la Tabla 4.23

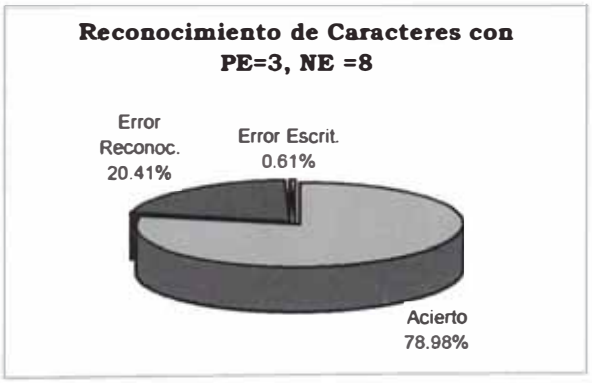


Tabla 4.25 Representación gráfica de la distribución de Errores de Escritura y errores de Reconocimiento

Donde :

PE : Profundidad de Entrenamiento

NE : Número de Entrenamientos aplicados a la Red Neuronal.

DATOS PROCEDENTES DE LOS FORMULARIOS DE TRABAJO

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	# Caracteres
ESCUELA NUEVA	LIMA	LIMA	SJM	AREQUIPA		VENTA DE UTILES ESCOLARES	MATILDE GARZON	YENNY LINARES	78
EL ESTUDIANTE	LIMA	LIMA	COMAS	LOS JAZMINES		VENTA DE PIZARRAS ACRILICAS	RUBEN CARREON	DAVID DIAZ	81
COMPUTER REY	LIMA	LIMA	CERCADO	WILSON		VENTA DE COMPUTADORAS	JAVIER CONTRERAS	DANIELA RIVAS	78
SELVA NEGRA	LIMA	LIMA	SMP	SANTA ROSA		MATERIALES DE CONSTRUCCION	DANIEL CARO	WILLIAN DAMIAN	77
ELECTRONICA HENRY	LIMA	LIMA	SAN MIGUEL	LOS ROBLES	J	ELECTRONICA EN GENERAL	ENRIQUE PEREZ	DAMIAN	81
SERVICIOS CAMILA	LIMA	LIMA	CALLAO	LOS CLAVELES	J	SERVICIO DE FOTOCOPIAS	RICARDO JIMENEZ	MANUEL TITO	85
EL MARTILLO	LIMA	LIMA	SAN BORJA	LOS SAUCES		FABRICACION DE MUEBLES	SAMUEL GARCES	DANIEL ESPEJO	79
IMPRENTA VELOZ	LIMA	LIMA	SAN MARTIN	LOS DOMINICOS		IMPRESIONES EN GENERAL	HECTOR CARRASCO	CARMEN LOAYSA	88
SUPERMERCADO EL TIGRE	LIMA	LIMA	SAN MIGUEL	LAS CAMELIAS		INSUMOS ALIMENTICIOS	SARA RIQUEL	MANUEL VELEZ	87
EL RINCON PIURANO	LIMA	LIMA	CHORRILLOS	LOS ALAMOS	B	VENTA DE PLATOS TIPICOS	CARLOS GAMARRA	BEATRIZ SANTOS	89
								TOTAL	823

CARACTERES RECONOCIDOS POR RECAM

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	E. Recon	E. Escrit.
ESCUELA NUEVA	LIMA	LIMA	SJM	AREQLIPA		VENTA DE UIILES ESCOLARES	MAIILDE GARZGN	YENMY LJKARZS	8	
ELESTDDIANIG	LIMA	LTMA	CDMAS	LOS JAZMINES		VENIA DE PIIARIAS ALRILICFS	RUBEN CHRKEDN	OAVID DIAZ	14	1
COMPDTFR REY	LIMA	LIMA	CERCHDO	WILSPN		VENIA DE COMPUTADCRAS	JAVIER CONTREKAS	DAKIELA AIVAS	9	
SELVA NEORA	LIMA	LIMA	SN P	SANTA RDSA		MATERIALES DE CONSTRUCCION	DANIEL CANU	WILLIAN DAMIAR	6	1
ELECIRUNICA KENRY	LIMA	LIMA	SAN MIGUGL	LDS RDBLES	J	ELECIRUMICA EN GENZRAL	EMZIQUP PERQY	OAMIAU	17	
SEFVICIOS CAMILA	LIMA	LIMA	CALLAD	LOS CLAVELES	J	SERVICIO CE FOTOCOPIAS	RICARDU JIMENEZ	MANUEL TIID	6	
FLMAKTILID	LIMA	LIMA	SANBDRJA	LOS JAUCES		FABRICACICN DE MUEBLES	KAMUFL GARCES	DANJEL FSPEJO	11	2
IMPREAYA VELDZ	LIMA	LIMA	SHN MAPTIN	LOS DOMINICOS		IMPPESIONES FN OENYRAL	HECTUR CARRASCC	CARMEN LOAYSA	11	
SUPERMLRCADD EL TIGRE	LIMA	LIMA	SAN MIGUEL	LAS CAMELIAS		INSUMOS ALIMENTICIOS	SARA RIOUEL	MANUFL VELEZ	4	1
RL RINCDN PIURAND	LIMF	LIMA	CHORRILLOS	LOS ALAMDS	B	VENTA DE PLATOS TIPICDS	CARLOS GAMARRA	BEATRIZ SARTOS	7	
								TOTAL	93	5

Tabla 4.26 Resultados de la Prueba de Integración : Reconocimiento con Profundidad de Entrenamiento = 3 y Número de Entrenamientos = 9

Resumen de la Tabla 4.26

Total errores	98
Porcentaje de Error Reconocimiento	11.30
Porcentaje de Error Escritura	0.61
Total Porcentaje de Error	11.91
Total Porcentaje de Acierto	88.09

Tabla 4.27 Resumen de los resultados obtenidos por RECAM, con profundidad de entrenamiento = 3 y número de entrenamientos = 9

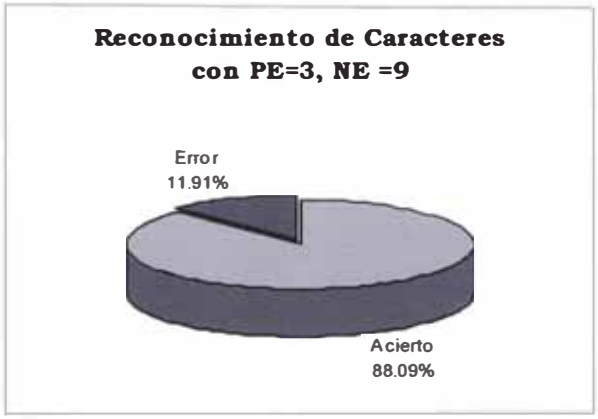


Tabla 4.28 Representación Grafica del total de porcentajes de Error y Acierto mostrados en la Tabla 4.27

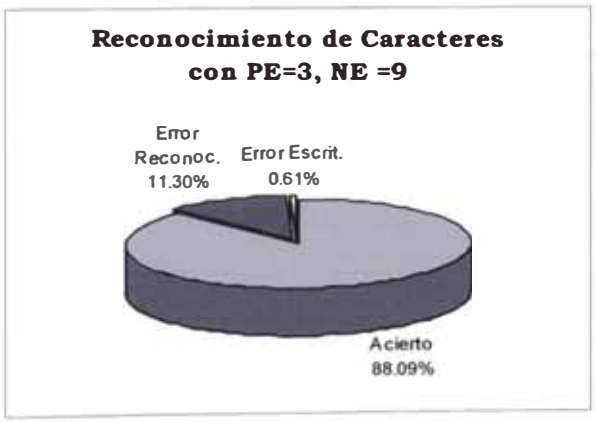


Tabla 4.29 Representación Gráfica de la distribución de errores de Escritura y errores de Reconocimiento

Donde :

PE : Profundidad de Entrenamiento

NE : Número de Entrenamientos aplicados a la Red Neuronal.

DATOS PROCEDENTES DE LOS FORMULARIOS DE TRABAJO

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	# Caracteres
ESCUELA NUEVA	LIMA	LIMA	SJM	AREQUIPA		VENTA DE UTILES ESCOLARES	MATILDE GARZON	YENNY LINARES	78
EL ESTUDIANTE	LIMA	LIMA	COMAS	LOS JAZMINES		VENTA DE PIZARRAS ACRILICAS	RUBEN CARREON	DAVID DIAZ	81
COMPUTER REY	LIMA	LIMA	CERCADO	WILSON		VENTA DE COMPUTADORAS	JAVIER CONTRERAS	DANIELA RIVAS	78
SELVA NEGRA	LIMA	LIMA	SMP	SANTA ROSA		MATERIALES DE CONSTRUCCION	DANIEL CARO	WILLIAN DAMIAN	77
ELECTRONICA HENRY	LIMA	LIMA	SAN MIGUEL	LOS ROBLES	J	ELECTRONICA EN GENERAL	ENRIQUE PEREZ	DAMIAN	81
SERVICIOS CAMILA	LIMA	LIMA	CALLAO	LOS CLAVELES	J	SERVICIO DE FOTOCOPIAS	RICARDO JIMENEZ	MANUEL TITO	85
EL MARTILLO	LIMA	LIMA	SAN BORJA	LOS SAUCES		FABRICACION DE MUEBLES	SAMUEL GARCES	DANIEL ESPEJO	79
IMPRENTA VELOZ	LIMA	LIMA	SAN MARTIN	LOS DOMINICOS		IMPRESIONES EN GENERAL	HECTOR CARRASCO	CARMEN LOAYSA	88
SUPERMERCADO EL TIGRE	LIMA	LIMA	SAN MIGUEL	LAS CAMELIAS		INSUMOS ALIMENTICIOS	SARA RIQUEL	MANUEL VELEZ	87
EL RINCON PIURANO	LIMA	LIMA	CHORRILLOS	LOS ALAMOS	B	VENTA DE PLATOS TIPICOS	CARLOS GAMARRA	BEATRIZ SANTOS	89
								TOTAL	823

CARACTERES RECONOCIDOS POR RECAM

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	E. Recon.	E. Escrit.
ESCUELA NUEVA	LIMA	LIMA	SJM	AREQVIPA		VENTA DE UTILES ESCDLAHES	MAIILDE GARZGN	YENNY LIKARSS	7	
ELESTUDIANTG	LIMA	LIMA	GDMAS	LOS JAZMINES		VENIA DE PIZARIAS ACRILICFS	RUBEN CARKEDN	DAVID DIAZ	9	1
COMPUTFR REY	LIMA	LIMA	CEREHDD	NILSDN		VENIA DE COMPUTADCRAS	JAVTER CONTREKAS	DAKIELA ATVAS	13	
SELVA NEGRA	LIMA	LIMA	SM P	SAMTA RDSA		MATERIALES UE COUSTRUDCION	DANIEL CANU	RILLIAN DAMIAN	8	1
ELECTRCNICA KENRY	LIMR	LIMA	SAM MIGUGL	LDS RDBLES	J	FLECTRUNICA EN GENIRAL	GNZIQUP PEUHZ	DAMIAU	17	
SEFVICIOS CAMILA	LIMA	LIMA	CALLAD	LOS CLAVELES	J	SERVICIO DE FCTOCOPIAS	RICARDU JIMENEZ	MANUFL TITO	5	
FLMAKTILLO	LIMA	LIMA	SANBDRCA	LOS SALCES		FABRICACICN OE MUEBLES	KAMUFL GARCES	DANJEL FSPFJO	12	2
IMPRENYA VELOZ	LIMA	LIMA	SAN MARTSM	LOS DOMINICOS		IMPPESIONES FN GENYRAL	HECTCR CARRASCC	CARMEW LOAYSA	9	
SUPERMLPCAOO EL TIGRE	LIMA	LIMA	SAN MIGUEL	LAS CAMELIAS		INSUMOS ALIMENJICIOS	SARA RIQUEL	MAUUQX VELEZ	7	1
EL RTNCON PIURAND	LIMA	LIMA	CHOPRILLOS	LOS ALAMOS	B	VENTA DE PLATOS TIPICDS	CARLOS AAMARRA	BEATRIZ SANTOS	5	
								TOTAL	92	5

Tabla 4.30 Resultados de la Prueba de Integración : Reconocimiento con Profundidad de Entrenamiento = 3 y Número de Entrenamientos = 10

Resumen de la Tabla 4.30

Total errores	97
Porcentaje de Error Reconocimiento	11.18
Porcentaje de Error Escritura	0.61
Total Porcentaje de Error	11.79
Total Porcentaje de Acierto	88.21

Tabla 4.31. Resumen de los resultados obtenidos por RECAM, con profundidad de entrenamiento = 3 y número de entrenamientos = 10

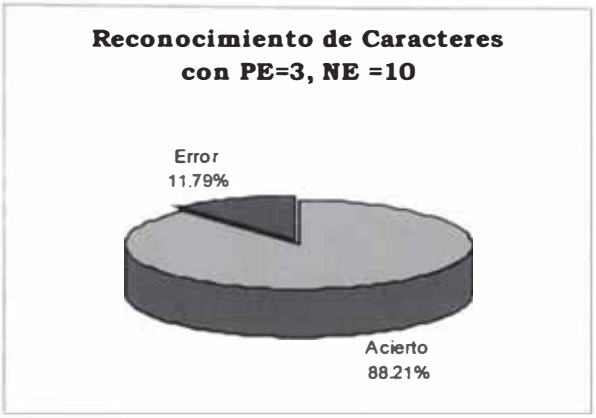


Tabla 4.32 Representación grafica del total de porcentajes de Error y Acierto mostrados en la Tabla 4.31

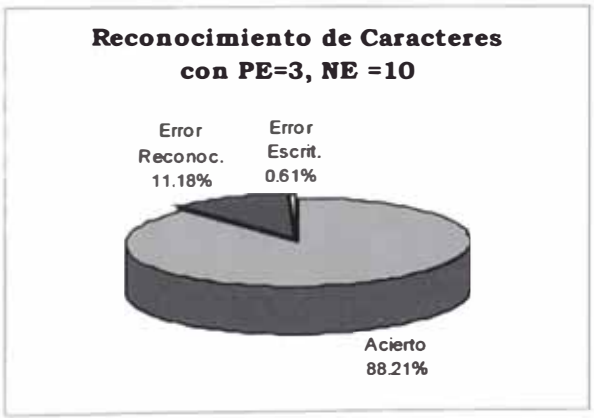


Tabla 4.33 Representación gráfica de la distribución de errores de Escritura y errores de Reconocimiento

Donde :

PE : Profundidad de Entrenamiento
NE : Número de Entrenamientos aplicados a la Red Neuronal.

DATOS PROCEDENTES DE LOS FORMULARIOS DE TRABAJO

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	# Caracteres
ESCUELA NUEVA	LIMA	LIMA	SJM	AREQUIPA		VENTA DE UTILES ESCOLARES	MATILDE GARZON	YENNY LINARES	78
EL ESTUDIANTE	LIMA	LIMA	COMAS	LOS JAZMINES		VENTA DE PIZARRAS ACRILICAS	RUBEN CARREON	DAVID DIAZ	81
COMPUTER REY	LIMA	LIMA	CERCADO	WILSON		VENTA DE COMPUTADORAS	JAVIER CONTRERAS	DANIELA RIVAS	78
SELVA NEGRA	LIMA	LIMA	SMP	SANTA ROSA		MATERIALES DE CONSTRUCCION	DANIEL CARO	WILLIAN DAMIAN	77
ELECTRONICA HENRY	LIMA	LIMA	SAN MIGUEL	LOS ROBLES	J	ELECTRONICA EN GENERAL	ENRIQUE PEREZ	DAMIAN	81
SERVICIOS CAMILA	LIMA	LIMA	CALLAO	LOS CLAVELES	J	SERVICIO DE FOTOCOPIAS	RICARDO JIMENEZ	MANUEL TITO	85
EL MARTILLO	LIMA	LIMA	SAN BORJA	LOS SAUCES		FABRICACION DE MUEBLES	SAMUEL GARCES	DANIEL ESPEJO	79
IMPRENTA VELOZ	LIMA	LIMA	SAN MARTIN	LOS DOMINICOS		IMPRESIONES EN GENERAL	HECTOR CARRASCO	CARMEN LOAYSA	88
SUPERMERCADO EL TIGRE	LIMA	LIMA	SAN MIGUEL	LAS CAMELIAS		INSUMOS ALIMENTICIOS	SARA RIQUEL	MANUEL VELEZ	87
EL RINCON PIURANO	LIMA	LIMA	CHORRILLOS	LOS ALAMOS	B	VENTA DE PLATOS TIPICOS	CARLOS GAMARRA	BEATRIZ SANTOS	89
								TOTAL	823

CARACTERES RECONOCIDOS POR RECAM

razon_emp	Depart	Provincia	Distrito	Direcc	M	Actividad	Ggeneral	gproduc	E. Recon.	E. Escrit.
FSCUELA NUEVA	LIMA	LIMA	SJM	AREQULPA		VENTA DE UIILES ESCLLAHES	MAIILDE GARZGN	YENNY LIKASSS	10	
ELESTDDIHNIE	LIMA	LIMA	CDMAS	LOS JAZMINES		VENIA DE PIIARIAS ACRILICAS	RUBEN CHRKEDN	OAVID OIAZ	12	1
COMPLTFR REY	LIMA	LIMA	CERCHDO	NILSDM		VENIA DE COMPUTADCRAS	JAVIER CONTREKAS	DANIELA AIVAS	10	
SELVA NEGRA	LIMA	LIMA	SM P	SAMTA ROSA		MATERIALES DE COUSTRUCCION	DANIEL CANU	NILLIAN DAMIAN	5	1
ELECIRCNICR KENRY	LIMR	LZMA	SAM MIGUGL	LDS RDBLES	J	FLECIRUNICA EN OEMERAL	GNZIQUE PERLZ	DAMIAU	19	
SEFVICIOS CAMILA	LIMA	LIMA	CALLAD	LOS CLAVELES	J	SERVICIO DE FOICCOPIAS	RICARDD JIMENEZ	MANUFL TIID	8	
FLMAKTILLD	LIMA	LIMA	SANBDRJA	LOS SAUCES		FABRICACICN DE MUEBLES	KAMUFL GARCES	OANTEL FSPFJD	12	2
IMPRENIA VELDZ	LIMA	LIMA	SHM MAPTSM	LOS DOMINICDS		IMPRESIONES EN GENYRAL	HECTOR CARRASCC	CARMEN LDAYSA	11	
SUPERMLPCADD EL TIGRE	LIMA	LIMA	SAN MIGUEL	LAS CAMELIAS		INSUMOS ALIMENTICIDS	SARA RIQUEL	MANUEL VELEZ	4	1
EL RINCDN PIURAND	LIMA	LIMA	CHOPRILLOS	LOS ALAMDS	B	VENTA DE PLATOS TIPICLS	CARLOS GAMARRA	BEATRIZ SANTOS	5	
								TOTAL	96	5

Tabla 4.34 Resultados de la Prueba de Integración : Reconocimiento con Profundidad de Entrenamiento = 3 y Número de Entrenamientos = 11

Resumen de la Tabla 4.34

Total errores	101
Porcentaje de Error Reconocimiento	11.66
Porcentaje de Error Escritura	0.61
Total Porcentaje de Error	12.27
Total Porcentaje de Acierto	87.73

Tabla 4.35. Resumen de los resultados obtenidos por RECAM, con profundidad de entrenamiento = 3 y número de entrenamientos = 11

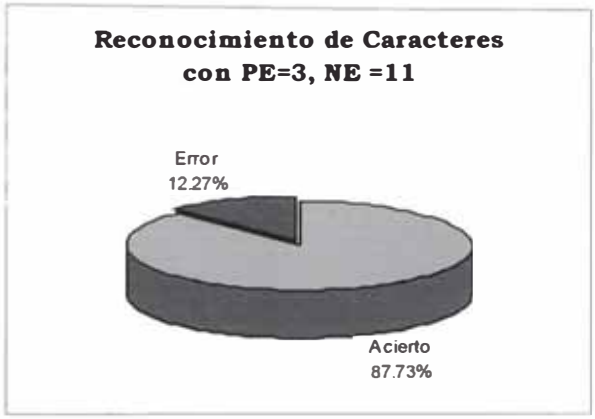


Tabla 4.36 Representación gráfica del total de porcentajes de Error y Acierto mostrados en la Tabla 4.35

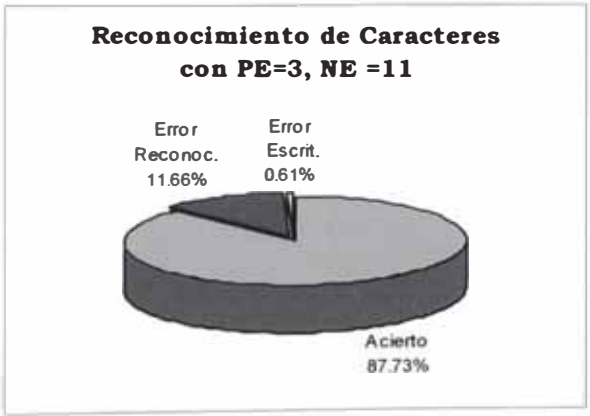


Tabla 4.37 Representación gráfica de la distribución de errores de Escritura y errores de Reconocimiento

Donde :

PE : Profundidad de Entrenamiento

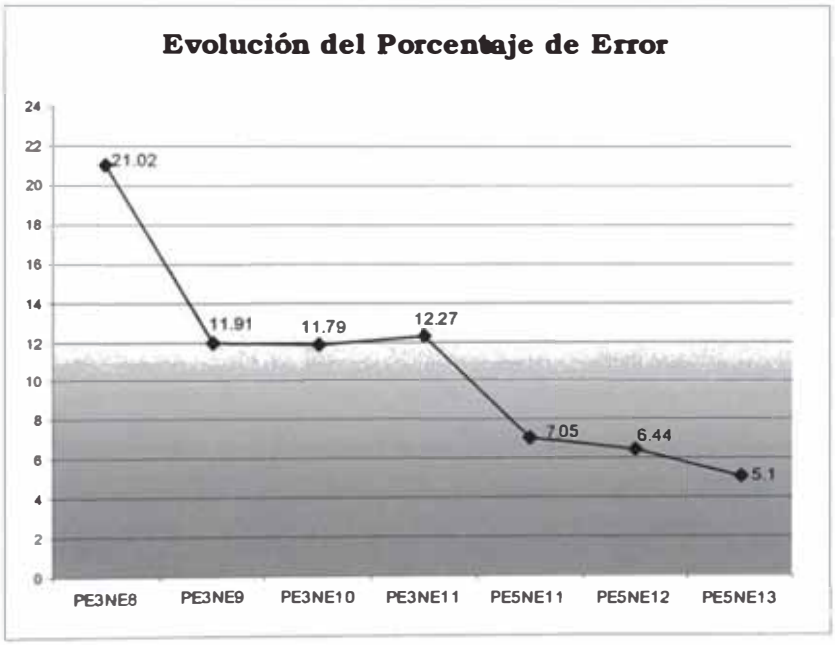
NE : Número de Entrenamientos aplicados a la Red Neuronal.

4.2.1. Interpretación de Resultados :

De las Cuadros Resumen de los Resultados Obtenidos por RECAM, se ha podido extraer los resultados finales de las pruebas de integración, es decir los resultados del funcionamiento del Sistema en conjunto.

Claramente se puede diferenciar que según la variación del valor de los parámetros ***Profundidad de entrenamiento (PE)*** y ***Número de Entrenamientos (NE)*** se obtiene resultados diferentes, los cuales se muestran en el siguiente cuadro comparativo.

Tabla. 4.50 : Cuadro comparativo de los porcentajes de error



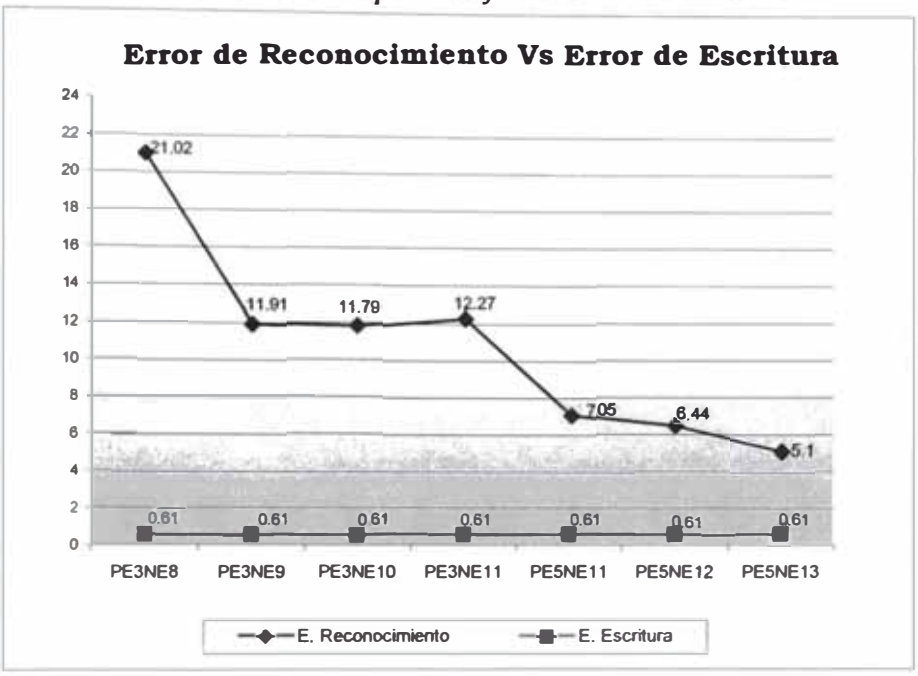
Según la tabla 4.50, podemos apreciar que el mínimo porcentaje de error sucede cuando la profundidad de entrenamiento es de 5 y número de entrenamientos es 13.

Pero aquí debemos tener en cuenta que los errores que se encontraron son Errores tanto de Reconocimiento así como Errores de Escritura.

Los errores de Reconocimiento son dependientes del funcionamiento y rendimiento del sistema en su totalidad, y son factibles de ser mejorados. Pero aquellos errores de Escritura, son independientes del Sistema, por ello según la

variación de los parámetros: Profundidad de Entrenamiento y Número de Entrenamientos, estos valores han permanecido inalterables.

Tabla. 4.51 Cuadro comparativo de los porcentajes de error de reconocimiento Vs porcentajes de error de escritura



En cuanto a los errores de Escritura, estos se generan con la Escritura confusa al momento de rellenar la información en los Formularios, es decir son errores causados por lo usuarios, a continuación se muestra algunos casos de ejemplo:

Tabla 4.52 Casos representativos de errores de escritura.

CASO	OBSERVACION
	Según el contexto : "D" Según la escritura : "O"
	Según el contexto : "R" Según la escritura : "A o R"
	Según el contexto : "I" Según la escritura : "L o I"
	Según el contexto : "I" Según la escritura : "I"

Otros errores debido a la escritura ocurren cuando la distancia entre uno y otro caracter no es el adecuado, es decir, si esta distancia es demasiado pequeña

ocurrirá un error de Solapamiento y si la distancia es grande, inmediatamente el Sistema deducirá que se trata de un espaciado entre dos palabras, tal es el siguiente caso.

CASO	OBSERVACION
	Según el contexto : "SMP" Según la escritura : "SM P"

Figura. 4.12 Caso representativo de error en la separación de letras

A pesar que la información ha sido rellenada por una sola persona, se ha podido observar la diversidad de formas que se puede tener al momento de escribir el mismo caracter. A continuación se presenta la diversidad de formas en escritura para el caracter "E".

Figura 4.13 Formas diferentes de escribir el caracter "E"

CAPITULO V

CONCLUSIONES

Primero : Para el reconocimiento de caracteres manuscritos aislados en campos de formularios, se ha diseñado dos Redes Neuronales Competitivas Simples de dos capas, una exclusivamente para el reconocimiento de dígitos y otra para el reconocimiento de letras, donde las neuronas de la capa de entrada en ambas redes neuronales, representan a los pixeles que conforman la imagen de los caracteres, por lo cual tiene una distribución bidimensional y las neuronas de la capa de salida están constituidas por el espacio de caracteres que la Red Neuronal es capaz de reconocer, para el caso de los dígitos comprende el espacio de números entre 0 y 9 y para el caso de las letras comprende el espacio de letras mayúsculas del alfabeto occidental que van de la A a la Z.

Segundo : En la determinación del posicionamiento de los caracteres en los recuadros de los campos de formularios, se ha utilizado las técnicas de Histogramas Verticales e Histogramas Horizontales, con las cuales se logro eliminar los recuadros contenedores quedando disponibles solo las imágenes aisladas de los caracteres (números y/o letras), las cuales algunas veces contenían manchas y puntos no deseados (ruido), lo que alteraba los resultados del proceso de reconocimiento, para eliminar este inconveniente se ha diseñado un algoritmo de Filtrado de Ruido basado en el concepto de 4–vecindad.

Tercero : Debido a la complejidad del Sistema, se vio por conveniente dividirlo en dos Subsistemas el primero "Neuron", que se encarga del Reconocimiento de imágenes propiamente dicho, en el cual se ha generado los archivos de extensión "DNR", donde se almacena la Red Neuronal, y el segundo "Recam", que engloba al anterior y se ocupa además del tratamiento de Formularios, este último genera archivos de extensión "PLA", en los cuales se almacena la información correspondiente a las plantillas que permiten identificar los campos a reconocer en los Formularios de Trabajo.

Cuarto : Los resultados en conjunto emitidos por el prototipo, demuestran resultados óptimos al problema de reconocimiento de caracteres manuscritos aislados en campos de formularios, puesto que el mejor resultado en las Pruebas de integración identifica un porcentaje mínimo, 4.49% de error en cuanto al reconocimiento y el porcentaje de error de 0.61% en cuanto a la escritura.

Quinto : El Reconocimiento propiamente dicho, depende de varios factores, tales como: la resolución de Escaneo, la cual se ha determinado en un optimo de 200 pixeles por pulgada (ppp), el formato de la imagen, que debe estar en formato BMP Monocromático, el ángulo de inclinación de la imagen, que debe ser cero, el número de neuronas de Capa de Entrada de la Red Neuronal que debe permitir representar a los caracteres con todos sus detalles, la profundidad del entrenamiento y el número mismo de entrenamientos que se le aplique a la Red Neuronal.

Sexto : Con respecto al número de neuronas de la capa de entrada, mientras mayor sea este número, se obtiene una representación más detallada de las letras y números que la Red Neuronal es capaz de reconocer, pero se corre el riesgo de incrementar notablemente el tiempo de procesamiento. Por ello se ha realizado un estudio de los diferentes tamaños característicos de letras y números, llegándose a los siguientes resultados, en el caso de números el alto característico es de 26 píxeles, y el ancho característico es de 22 píxeles, por lo tanto la Red Neuronal de Números en su capa de entrada consta de 572 neuronas distribuidas en forma bidimensional; y en el caso de las

letras, estas tienen un alto característico de 24 píxeles y un ancho característico también de 24 píxeles, por tanto la Red Neuronal de Letras en su capa de entrada consta de 576 neuronas, distribuidas en forma bidimensional.

Séptimo : El número de entrenamientos de la Red Neuronal, es proporcional al número de neuronas de la Capa de Salida de la Red Neuronal, es decir mientras mayor sea el número de patrones que a reconocer y/o entrenar por la Red Neuronal, mayor ha de ser el número de entrenamientos aplicados a la Red Neuronal.

Octavo : La Red Neuronal no esta diseñada exclusivamente para el reconocimiento de números y letras, en los formularios utilizados como muestra para la presente investigación, sino que su uso puede ser amplificado a diversos formularios cuya información este enmarcada en campos, así mismo la Red neuronal diseñada no solo se limita a reconocer letras y/o dígitos, sino que su uso puede ser ampliado al reconocimiento de diversos patrones.

CAPITULO VI

RECOMENDACIONES Y SUGERENCIAS

Primero : Utilizar la técnica de Redes Neuronales en problemas de Reconocimiento de Caracteres Manuscritos, puesto que ha demostrado ser eficiente.

Segundo : Se recomienda tener en cuenta las dimensiones en cuanto al alto y ancho del tipo de caracteres manuscritos que la Red Neuronal va ha reconocer, antes de definir el número de neuronas de la capa de entrada de la Red Neuronal.

Tercero : Se recomienda realizar un número considerable de entrenamientos a la Red Neuronal, de manera que se puedan obtener mejores resultados en la etapa de reconocimiento.

Cuarto : En cuanto al rellenado de información en los formularios, se recomienda, que la letra debe ser legible y todo en letras mayúsculas, además considerar un espacio mínimo de por lo menos 1m.m entre caracteres en el caso de que los Formularios no tengan líneas divisorias ni rejillas en la imagen de sus campos, y evitar pegarse al borde inferior o a los laterales, a fin de evitar problemas en cuanto la segmentación.

Quinto : Se sugiere implementar un Modulo de detección y corrección del ángulo de inclinación de las imágenes de los formularios escaneados, lo cual ayudaría en gran medida a disminuir el número de errores en cuanto al reconocimiento.

Sexto : Se sugiere implementar un analizador sintáctico que nos ayude en la detección y corrección de errores de reconocimiento emitidos por el Sistema.

BIBLIOGRAFÍA

[1] BERND BRUEGGE, ALLEN H. DOTOIT "Ingeniería de Software Orientado a Objetos", Prentice Hall, Primera Edición, 2002.

[2] BERRIN YANIKOGLU, PETER A. SANDON. "Segmentation of Off Line Cursive Handwriting using Linear Programming", Pattern Recognition Society. Elsevier Science Ltd. 1998.

[3] CEVALLOS SIERRA, JAVIER. "Microsoft Visual Basic 6.0, Curso de Programación". Editorial AlfaOmega 1999.

[4] FERNANDO MARTÍN RODRÍGUEZ, DAVID LÓPEZ CRESPO Y FRANCISCO PARADA LOIRA, "Localización y Segmentación de Caracteres en Formularios manuscritos" Departamento de Tecnologías de las Comunicaciones Universidad de Vigo URSI 2001

[5] HERTZ J., A. KROGH, R. PALMER. "Introduction to the Theory of Neural Computation". Santa Fe, Institute Editorial Board. 1990.

[6] HILERA, J.R. MARTINEZ, "Redes Neuronales Artificiales. Fundamentos, Modelos y Aplicaciones", V.J. (RA-MA) 2003.

[7] IÑARI ARRIAGA IBANEZ, "Reconocimiento de Caracteres Manuscritos en Plantillas mediante Procesado de distorsiones y Aplicaciones de Mascaras Flotantes". Universidad del País de Vasco,

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicaciones. Estudio de Viabilidad. Diciembre de 2000.

[8] KNERR S., L. PERSONNAZ Y G. DREYFUS. "Handwritten Digit Recognition by Neural Networks with single layer training", IEEE Transactions on Neural Networks, VolUmen 3, número 6, 962-968. - 1992.

[9] KONG, MAYNARD "Inteligencia Artificial" Fondo Editorial de la Pontificia Universidad Católica del Perú Lima, Perú 1993.

[10] KUO-CHIN FAN JENG-MING LU JING-YUH WANG, "A Feature Point Clustering Approach to the Segmentation of Form Documents", Institute of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan, R.O.C. 1995.

[11] LISA AVILA, CESAR, "Modelando con UML, Principios y aplicaciones", RJ. S.R. Ltda. Trujillo – Perú 2001.

[12] M. BLUMENSTEIN, "A New Segmentation Algorithm for Handwritten Word Recognition", Verma1,2. 2001

[13] PAJARES MARTINSANZ, GONZALO – SANTOS PEÑAS, MATILDE, "Inteligencia Artificial e Ingeniería del Conocimiento", Facultad de Informática – Universidad Complutense de Madrid, Editorial RA-MA, 2005.

[14] PANDYA, ABHIJIT S. MACY ROBERT B. "Pattern Recognition with Neural Networks in C++".CRC Press. 1996.

[15] PERDITA STEVEN, ROB POOLEY, "Utilización de UML en Ingeniería de Software con Objetos y Componentes", Addison Wesley, 2002.

[16] PRESSMAN, ROGER S. "Ingeniería de Software"Un enfoque Práctico. Editorial McGraw-Hill / Interamericana de España S.A.U. Quinta Edición. España 2001.

[17] RUSELL, STUART Y NORVING, PETER. "Inteligencia Artificial". Un Enfoque Moderno. Editorial Prentice Hall Hispanoamericana S.A. México. 1996.

[18] SIMPSON, PATRICK K. "Foundation of Neural Networks" – Neural Network Theory, Technology, an Applications – IEEE Technology Update Series.

[19] SIMPSON P.K. "Foundations of neural Networks". Artificial Neural Networks. IEEE, PRESS. New York. 1992

[20] SOMAYA AL-MA'ADEED, "An Off Line Arabic Handwriting Recognition System", School of Computer Science and information Technology, Jubilee Campus, University of Nottingham. 2001

[21] SOMMERVILLE IAN, "Ingeniería de Software" Pearson Educación, Addison Wesley, Sexta Edición México 2002.

[22] SRIRAM, RAM D. "Intelligent Systems for Engineering", A knowledge-based Approach. Springer-Verlag London, Great Bretain, 1997.

7.1 REFERENCIAS ELECTRÓNICAS

[1] ABBYY SOFTWARE HOUSE, ABBYY develops OCR/ICR/OMR, Forms Processing and Language Software.
http://www.abbyy.com

[2] ANALISIS Y DISEÑO DE SOFTWARE – EJEMPLOS UML
http://dis.um.es/jmolina/as.html

[3] ANTONIO BLASCO LÓPEZ , FRANCISCO FÉLEZ ESTEBAN, "Reconocimiento De Caracteres Manuscritos", Tutorial de Redes Neuronales aplicadas en Ingeniería Eléctrica, Facultad de Ingeniería Eléctrica de la Universidad Tecnológica de Pereira, Ing. Maria Isabel Acosta Buitrago, Ing. Camilo Alfonso Zuluaga Muñoz, Ing. Harold Salazar 2000
http://ohm.utp.edu.co/neuronales

[4] CARLOS SERRANO, Las redes neuronales artificiales, Sistemas. Informativos Contables, Profesor de la Universidad de Zaragoza (España).
http://ciberconta.unizar.es/LECCION/REDES/

[5] DR. DIEGO ANDINA DE LA FUENTE "REDES NEURONALES ARTIFICIALES" Tutorial desarrollado en la Universidad Politécnica de Madrid-UPM(España), departamento de Señales, Sistemas y Radiocomunicaciones (SSR), Grupo de Circuitos (GC) y desarrollado con la participación del M. en I. Antonio Vega Corona de la Facultad de Ingeniería Mecánica, Eléctrica y Electrónica de la Universidad de Guanajuato dentro del Programa de Mejoramiento del Profesorado (PROMEP).
http://www.gc.ssr.upm.es/inves/neural/ann2/anntutorial.html

[6] DIGITAL FILE Reconocimiento Automático de Datos, Software al servicio de la Gestión Documental y de Contenidos.
http://www.digitalfile.net,

[7] ESTUDIO COMPARATIVO DE PROGRAMAS OCR – 2002, Fundación de CECS Manuel Caragol
http://www.funcaragol.org/html/cmocr2sl.htm

[8] ODEC, CAPTURA DE DATOS
http://www.odec.es/odec/index.php?id=400

[9] READSOFT, TEACHING THE WORLD'S COMPUTER TO READ
http://www.readsoft.es /ocr/forms/index.htm

[10] SCANSOFT, PRODUCTIVITY WITHOUT BOUNDARIES
http://www.scansoft.com/omniform/