ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

INGENIERÍA DE ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES

"DESARROLLO DE UN SISTEMA DE ACCESO PARA PARQUEADEROS Y COBRO TARIFARIO UTILIZANDO RECONOCIMIENTO DE PATRONES"

TRABAJO DE TITULACIÓN

Previo a la obtención del título de:

INGENIERO EN ELECTRÓNICA, CONTROL Y REDES INDUSTRIALES

AUTOR: MARTÍN DAMIAN MEDINA SÁNCHEZ
PEDRO EUGENIO CONTRERAS HURTADO

TUTOR: ING. MSC. FRANKLIN SAMANIEGO R.

Riobamba-Ecuador

2015

DEDICATORIA

A mi padres, quienes siempre supieron apoyarme y darme palabras de aliento. Con su ejemplo, sabiduría e infinito amor, Nelson Medina y Silvia Sánchez hicieron de mí un hombre de bien.

Martín

Con toda la humildad que de mi corazón se pueda emanar, dedico primeramente este trabajo a Dios, el que me ha dado fortaleza para continuar cuando a punto de caer he estado y permitirme haber llegado hasta este momento tan importante en mi formación profesional.

De igual forma dedico este trabajo a mis padres que han sabido formarme con buenos sentimientos, hábitos y valores y por ser el pilar más importante demostrando siempre su cariño y apoyo incondicional, y a todos los ingenieros que por su apoyo así como por la sabiduría que me transmitieron en todo este tiempo de estudiante.

Pedro

FIRMAS DE RESPONSABILIDAD Y NOTA.

NOMBRE	FIRMA	FECHA
Ing. Gonzalo Samaniego. Ph.D. **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	**….....**	**.....**
Ing. Alberto Arellano. **DIRECTOR DE LA ESCUELA DE INGENIERÍA ELECTRÓNICA EN CONTROL Y REDES INDUSTRIALES**	**….....**	**….....**
Ing. Franklin Samaniego. **DIRECTOR DE TESIS.**	**….....**	**….....**
Ing. Alberto Arellano **MIENBRO DEL TRIBUNAL.**	**….....**	**….....**
COORDINADOR SISBIB -ESPOCH	**….....**	**….....**

NOTA DE LA TESIS: **….....**

Nosotros **MARTÍN DAMIAN MEDINA SÁNCHEZ** y **PEDRO EUGENIO CONTRERAS HURTADO**, somos responsables de las ideas, doctrinas y resultados expuestos en esta tesis, y el patrimonio intelectual de la misma pertenece a la **ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**

.....….....	**.....**
Martín Damian Medina Sánchez	Pedro Eugenio Contreras Hurtado

ÍNDICE DE ABREVIATURAS

ANPR	Automatic Number Plate Recognition
ANT	Agencia Nacional de Tránsito
EEPROM	Electrically Erasable Programmable Read-Only Memory
FTDI	Future Technology Devices International
IA	Inteligencia Artificial
IDE	Integrated Development Environment
IP	Internet Protocol
KB	Kilo bytes
m	Metros
mA	Mili Amperios
MB	Mega Bytes
mm	Milímetros
OCR	Optic Caracter Recognition
OpenCV	Open Source Computer Vision
RAM	Random Access Memory
RNA	Red Neuronal Artificial
SRAM	Static Random Access Memory
TTL	Transistor-Transistor Logic
txt	Formato de Archivo de Texto
USB	Universal Serial Bus
V	Voltios
V.A.	Visión Artificial
WEB	World Wide Web

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El parque automotor de nuestro país se encuentra en constante crecimiento, y ello conlleva a que se susciten problemas debido la falta de estacionamientos y parqueo, además de deficiencias en el servicio como largas filas para el ingreso, registro y tarifación.

Es por ello que el presente trabajo tiene por objetivo dar una solución tecnológica acorde a este problema mediante la utilización de Redes Neuronales Artificiales. La RNA tiene por objetivo distinguir los diferentes patrones existente en las placas vehiculares, para que de manera automática se registre el ingreso y salida de los vehículos del área de parqueo. Los patrones a distinguir serán las letras mayúsculas del alfabeto, así como también los dígitos.

La Red Neuronal diseñada fue la red de retropropagación, debido a que es la red que presenta características más favorables para nuestro proyecto. La red fue implementada en el lenguaje C++ y se usaron software y librerías totalmente libres.

Previo al proceso de entrenamiento, se recopilaron datos para el entrenamiento, además del desarrollo de software para la segmentación y tratamiento de imágenes usando las librerías de OpenCV.

CAPÍTULO I

1. MARCO REFERENCIAL

En el presente capítulo se describirá de una manera más detallada los motivos y justificación del trabajo de investigación, así como los objetivos a cumplir.

1.1. PLANTEMIENTO DEL PROBLEMA

A continuación se presenta un estudio del problema, y se plantean soluciones acorde al mismo.

1.1.1. ANTECEDENTES

La inteligencia artificial es una de las ramas relativamente nuevas en el ámbito tecnológico, área de investigación que involucra varios campos entre ellos la robótica, informática y lógica. Todo esto para poder crear máquinas capaces de aprender y razonar por sí mismas mediante el uso de programas de cómputo inteligentes.

La capacidad de aprendizaje y razonamiento de este tipo de máquinas ha sido implementado en ejes estratégicos de la evolución humana como la medicina, ingeniería, automatización, la milicia, economía, además de áreas triviales como los juegos de vídeo. La gama de aplicaciones es tan basta y prometedora, que la inteligencia artificial se ha convertido en una rama esencial en la investigación y desarrollo de

prototipos eficientes, capaces de cumplir funciones que de otra manera serían imposibles.

La visión artificial (V.A.) o visión por computador es un subcampo de la inteligencia artificial cuya principal funcionalidad es obtener información relevante, simulando el proceso de obtención y síntesis de datos propios de la visión humana. La visión artificial comprende el proceso de adquisición de imágenes, preprocesamiento, segmentación, reconocimiento e interpretación.

La gama de aplicaciones que abarca la visión por computador se ubica en procesos críticos como el control de calidad, en robótica, identificación, seguridad, control de tráfico y militares, entre otras.

Debido al aumento del parque automotor en nuestro país, se generan problemas de congestionamiento por causa de la falta de lugares apropiados para el estacionamiento, lo que obliga a la creación de parqueaderos y sistemas de acceso tarifados. Los actuales sistemas de acceso a parqueaderos tarifados en muchas ocasiones no satisfacen los requerimientos y necesidades de los usuarios.

Generando largas filas para el ingreso debido al registro manual que se realiza a la entrada y la salida de los parqueaderos tarifados. Esta manera de trabajo requiere de un mayor número de personal capacitado hecho que comúnmente no sucede debido al costo que genera contratar mayor cantidad de personal.

1.1.2. FORMULACION DEL PROBLEMA

¿Cómo mejorar el sistema de ingreso y salida en un parqueadero tarifado?

1.1.3. SISTEMATIZACIÓN DEL PROBLEMA

- ¿Qué factores se deben tomar en cuenta para mejorar en un sistema de parqueaderos tarifados?

- ¿Qué posibles soluciones se pueden tomar para mejorar las condiciones del servicio?

- ¿Qué tecnología resulta la más conveniente para dar una respuesta adecuada a la problemática?

1.2. JUSTIFICACION DEL TRABAJO DE TITULACIÓN

A continuación se detalla la justificación del proyecto.

1.2.1. JUSTIFICACION TEÓRICA

El presente proyecto plantea un estudio teórico y práctico acerca de una rama relativamente reciente, que se espera que en años no muy lejanos sea el eje de una nueva revolución tecnológica, llamada inteligencia artificial (IA). Es por ello que es necesario y obligatorio un estudio más detallado de una red neuronal, aplicaciones y por último la puesta en marcha de una red.

El conocimiento teórico de una red neuronal no es suficiente en sí para poder desarrollar una red neuronal, es por ello que muchos profesionales entendidos en el tema han optado por usar programas que poseen librerías listas para simular una red. El problema de usar librerías radica en que el investigador o desarrollador debe adaptar su proyecto para hacer uso del código mencionado, limitando o restringiendo el avance del proyecto.

La contra a esta limitación es la capacidad de desarrollar un código propio, que sea adaptable y maleable a las necesidades de cada proyecto y que además de afianzar el conocimiento acerca de redes neuronales sea un referente para poder desarrollar investigaciones futuras en diferentes plataformas.

El campo de la visión artificial es muy amplio y abarca una gran cantidad de técnicas y métodos, por esta razón el proyecto utilizará el método de procesamiento de imágenes.

Como aplicación práctica de la inteligencia artificial se ha escogido el reconocimiento de patrones alfanuméricos usando redes neuronales artificiales para el aprendizaje y reconocimiento de caracteres, haciendo una comparación con estudios previamente realizados en áreas o temas afines.

Se investigará acerca de los diferentes tipos de redes neuronales, y de entre ellas se escogerá la red que presente características más favorables para nuestro proyecto. Esta red será implementada en C++. Se recopilará imágenes de caracteres alfabéticos y se procederá al entrenamiento de nuestra red. Se espera que el diseño de la red neuronal en C++ sea capaz de migrar a otro tipo de plataformas y de esta forma sentar las bases para el desarrollo posterior de aplicaciones.

1.2.2. JUSTIFICACION PRÁCTICA

En el entorno actual resulta de mucha utilidad un software que sea capaz de reconocer caracteres alfabéticos, que sea eficiente y de acceso libre, orientado al desarrollo de aplicaciones varias en campos multidisciplinares. Susceptible a modificaciones y adaptaciones dependiendo de la aplicación seleccionada, que se encuentre muy bien documentada y además implementada en un lenguaje de programación estándar.

El acceso a herramientas comerciales disponibles en el mercado es de costo muy elevado; ésta será una herramienta para el investigador y desarrollador de aplicaciones basadas en el reconocimiento de caracteres, distribuida de una manera totalmente gratuita y de código abierto orientado al desarrollo de aplicaciones de investigación, enseñanza y comerciales.

Por tal razón el proyecto estará enfocado en la creación de un sistema automatizado para el ingreso y cálculo tarifario de parqueaderos disminuyendo el tiempo de ingreso y reduciendo el congestionamiento que generan a la entrada y salida. Además de reducir los costos de personal y brindar un mejor servicio.

La solución al problema se plantea mediante el desarrollo de un algoritmo basado en una red neuronal, entrenado con muestras debidamente seleccionadas. El algoritmo será

capaz de identificar caracteres alfanuméricos presentes en las placas vehiculares. Se adecuarán dos sensores ubicados de manera estratégica para la detección del automóvil en las posiciones de ingreso o salida. El sistema estará controlado mediante el computador, existiendo una etapa de acoplamiento entre los sensores y el computador. Los datos entregados por los sensores serán analizados mediante un microcontrolador.

Las cámaras serán colocadas de manera que se obtenga el mejor ángulo de la placa, usando para esto cámaras con la definición más alta disponible.

La creación del algoritmo, enriquecerá los conocimientos afines a la carrera, como son las redes neuronales artificiales y que acercará a sus usuarios a las nuevas áreas del conocimiento como son las redes neuronales artificiales y visión artificial. Se espera que el programa sea una herramienta eficiente pues hará uso de bibliotecas igualmente de acceso libre dedicadas exclusivamente al procesamiento de imágenes y vídeo, capaz de competir con los programas comerciales.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

- Desarrollar un sistema de acceso para parqueaderos y cobro tarifario utilizando reconocimiento de patrones.

1.3.2. OBJETIVOS ESPECIFICOS

- Analizar los diferentes tipos de redes neuronales y de entre ellas escoger la más adecuada para nuestro proyecto.
- Diseñar y codificar el algoritmo de la red neuronal especificada, dimensionándola de acuerdo a nuestra necesidad.
- Recolectar información para el entrenamiento supervisado de la red.
- Investigar el funcionamiento de las librerías de OpenCV dedicadas al procesamiento de imágenes.

- Realizar pruebas y obtener resultados.

CAPÍTULO II

2. MARCO TEÓRICO

Para comprender en su totalidad el estudio planteado, es necesario conocer la teoría en la que se fundamenta el proyecto. En las secciones siguientes se tratan los temas más relevantes del trabajo de investigación.

2.1. INFORMACIÓN DE LAS PLACAS VEHICULARES EN EL ECUADOR

El ente regulador de las características que debe tener una placa vehicular en el Ecuador es la Agencia Nacional de Tránsito. Según la nueva ley de tránsito vigente desde el 5 de marzo de 2013, se establece nuevos lineamientos en cuanto a características de la placa.

Entre las características más relevantes para nuestro trabajo se puede destacar el cambio de dimensiones de la placa que ahora son de 154x404 mm, así como las dimensiones del carater de 70x38 mm. Se ha estandarizado la información que contiene la placa en series alfanuméricas de tres letras y cuatro números. La primera letra identifica la provincia y la numeración va desde el número 0001 hasta el número 9999.

Posee en la parte superior el logotipo de la ANT y la palabra Ecuador como identificador del país de procedencia, también posee según la naturaleza del servicio del automotor una franja de color específico (Tabla II-N°I). El fondo de la placa es blanco y el color de los caracteres es negro, el material es autoreflectivo por lo que mejora el

contraste entre el fondo y la serie.

TABLA II-N°1: FRANJA DE COLOR DE LAS PLACAS

N°	SERVICIO	COLOR
1	PUBLICO O COMERCIAL	NARANJA
2	ORGANISMO DEL ESTADO	ORO
3	GAD's REGIONALES, PROVINCIALES, MUNICIPALES Y PARROQUIALES	VERDE LIMON
4	DIPLOMÁTICOS Y ORGANISMOS INTERNACIONALES	AZUL
5	INTERNACIÓN TEMPORAL	ROJO
6	PARTICULAR	BLANCO

Fuente: Agencia Nacional de Tránsito

2.2. VISIÓN ARTIFICIAL

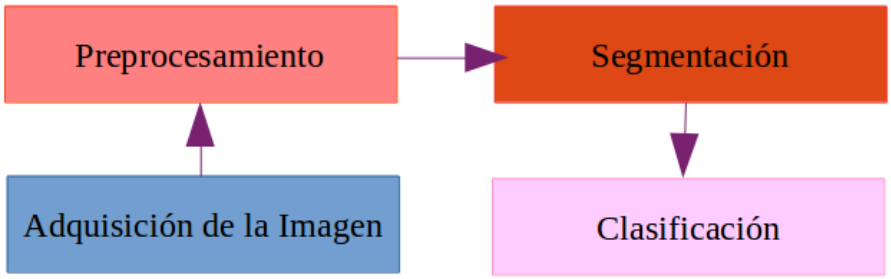
La visión artificial es una rama de la inteligencia artificial, que consiste en extraer la información de interés de una imagen para posteriormente procesarla. Es una potente herramienta para el desarrollo de proyectos y tiene un sinnúmero de aplicaciones, como en la robótica industrial, la milicia, la agricultura y muchas otras ramas de la ingeniería.

La visión por computadora nos permite obtener una gran cantidad de información del medio circundante, que de otra manera sería imposible. Una imagen contiene mucha información, de la que hay que aislar la región de interés. Para esto la visión artificial cuenta con un conjunto de pasos que nos permite cumplir con los objetivos planteados.

Las etapas típicas de una aplicación consta de la adquisición de la imagen, un preprocesamiento, segmentación y como último paso el reconocimiento o clasificación (Figura II-N°1).

Existen varios métodos para el reconocimiento de patrones, entre los principales se tienen métodos estadísticos, redes neuronales y lógica difusa.

FIGURA II-N°1: PASOS DE LA VISIÓN ARTIFICIAL



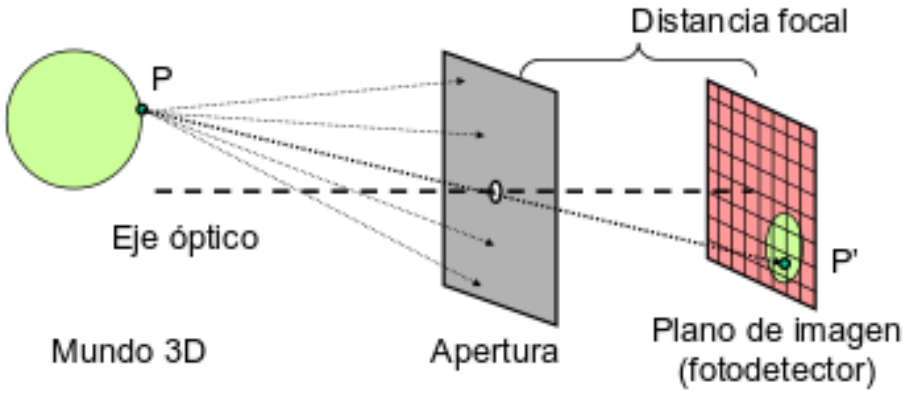
Fuente: Los Autores

2.3. IMAGEN DIGITAL

Una imagen digital es la representación matricial de la información captada por un sensor óptico. Los sensores se encargan de convertir la señal analógica proveniente de la fuente de iluminación en información digital. Típicamente los sensores usados para al adquisición de imágenes son las cámaras digitales.

FIGURA II-N°2: DIGITALIZACIÓN DE UNA IMAGEN

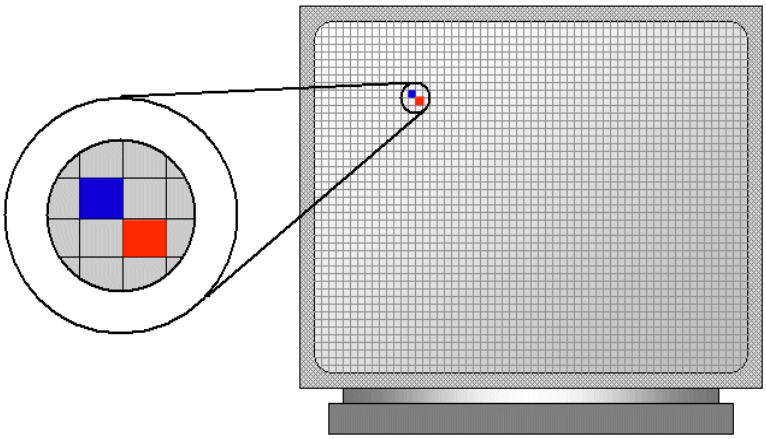
Fuente:



http://alereimondo.no-ip.org/OpenCV/uploads/41/tema1.pdf

La matriz que representa la imagen es bidimensional y cada uno de sus pares ordenados fila-columna contiene la información del color. Si la imagen está en escala de grises, la matriz tendrá un solo canal, pero si la imagen posee color, tendrá tres canales correspondientes al color rojo, verde y azul. A cada uno de los puntos fila-columna se los denomina píxeles. A mayor cantidad de píxeles, mayor es la fidelidad con la que se representa una imagen.

FIGURA II-N°3: PÍXELES DE UNA IMAGEN



Fuente: http://www.functionx.com/win32/images/pixel1.gif

2.4. OPENCV

OpenCV son librerías desarrolladas bajo el lenguaje de C/C++, distribuidas de forma gratuita para aplicaciones académicas y comerciales, que tienen la finalidad de ser un software para visión y aprendizaje por computadora. Puede ser instalada exitosamente en sistemas operativos como Windows, Mac, Android y Linux, además posee interfaces para C, C++, Java y Phyton.

Posee mas de 2500 funciones optimizadas para el procesamiento de imágenes, y ha sido ampliamente utilizado en aplicaciones como reconocimiento de patrones, seguimiento de objetos, extracción de modelos en 3D, etc.. También ha sido utilizada por compañías como Google, Intel, Sony, entre otras.

Es por esto que OpenCV se ha convertido en una de las principales herramientas para el desarrollo de proyectos que involucren visión por computador y procesamiento de imágenes.

2.5. OPERACIONES MORFOLÓGICAS

Son operaciones que extraen la información útil presente en las imágenes. La morfología es la forma y la estructura de una imagen. Las principales operaciones son: Dilatación, erosión, cierre y apertura.

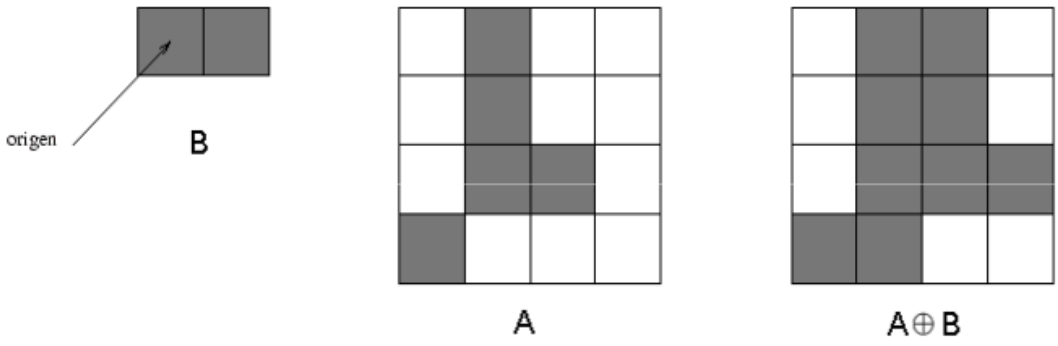
A partir de las operaciones de dilatación y erosión, su combinación y orden se puede obtener las operaciones de cierre y apertura. La operación de apertura está formada por una dilatación seguida de una erosión, y la de cierre de una erosión seguida de una de dilatación.

2.5.1. DILATACIÓN

La dilatación consiste en añadir píxeles a una imagen mediante un elemento reestructurante. El elemento reestructurante debe tener la dimensión que se desea dilatar en la imagen. Sea una imagen A, y un elemento reestructurante B, la dilatación se define como:

$$A \oplus B = \left\{ x \,|\, (\hat{B})_x \cap A \neq \varnothing \right\} \qquad \textbf{(1.1)}$$

FIGURA II-N°4: EJEMPLO DE DILATACIÓN



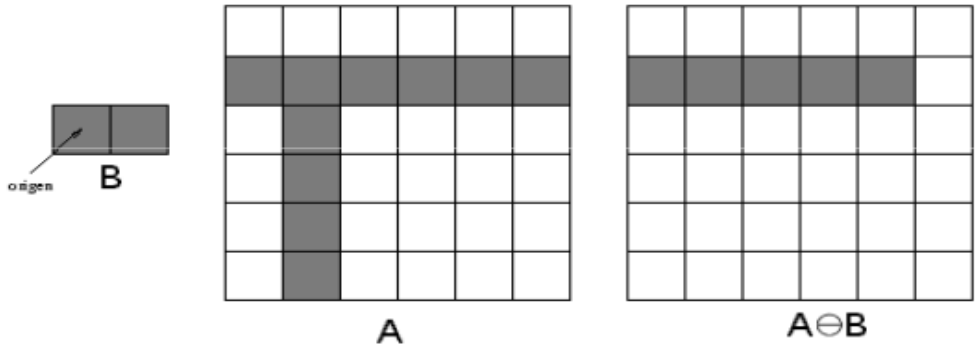
Fuente: http://alojamientos.us.es/gtocoma/pid/tema5-1.pdf p.7

2.5.2. EROSIÓN

La operación de erosión tiene como objetivo eliminar los píxeles adyacentes a la imagen A, igualmente depende de su elemento reestructurante B. Por definición la erosión es el conjunto de todos los elementos x para los cuales B trasladado por x está contenido en A.

$$A \ominus B = \{x \mid B_x \subseteq A\} \qquad \textbf{(1.2.)}$$

FIGURA II-N°5: EJEMPLO DE EROSIÓN



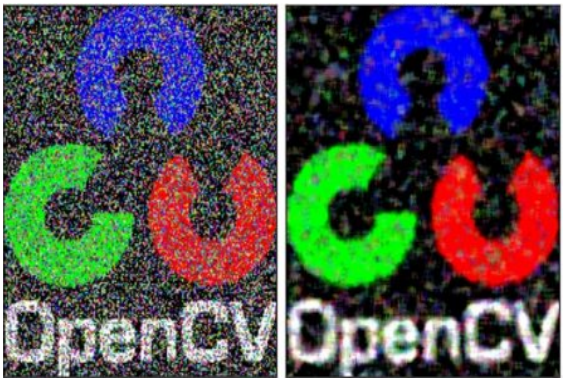
Fuente: http://alojamientos.us.es/gtocoma/pid/tema5-1.pdf p.12

2.6. FILTROS EN EL DOMINIO ESPACIAL

Los filtros en el dominio espacial actúan directamente sobre los valores de los píxeles. En estos filtros los píxeles interactúan con sus vecinos mediante una máscara cuadrada o rectangular, realizando operaciones generalmente de sumas ponderadas. El tamaño de la máscara determinará cuantos píxeles serán involucrados en la operación.

Los filtros lineales más utilizados son los filtros de suavizado y filtros de acentuamiento. El filtro de suavizado consiste en eliminar el ruido o detalles pequeños presentes en la imagen. Usualmente se le denomina filtro pasa bajos, debido a que elimina el ruido causado por altas frecuencias. El efecto obtenido es el equivalente a difuminar los detalles de una imagen.

FIGURA II-N°6: FILTRO DE SUAVIZADO

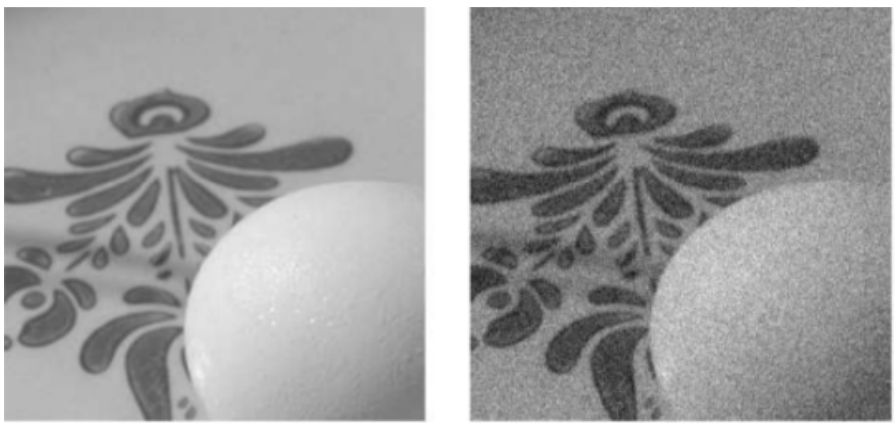


Fuente: http://opencv-python-tutroals.readthedocs.org/en/latest/_images/ median.jpg

Existen varios métodos para realizar éste filtro, uno de ellos es el filtro gaussiano, que es una aproximación normal o gaussiana en dos dimensiones.

Los filtros de acentuado resaltan los puntos de alta frecuencia, también se denomina filtros pasa altos.

FIGURA II-N°7: FILTRO DE ACENTUAMIENTO



Fuente: Sucar, L., Gómez, G., s.f., p.25

2.7. RECONOCIMIENTO DE PATRONES

El reconocimiento de patrones tiene por objetivo extraer de una imagen la información necesaria para compararla con patrones preestablecidos a fin de poder clasificarla.

Previo al reconocimiento de patrones, la imagen debe ser procesada para poder extraer las características más relevantes. Se puede representar la imagen mediante descriptores, que disminuye la información disponible en la imagen a clasificar.

El reconocimiento de patrones mediante métodos estadísticos se basa en la probabilidad y estadística, posee medidas numéricas con distribuciones de probabilidad, mediante ellas se realiza el reconocimiento.

El reconocimiento de patrones mediante redes neuronales artificiales es un sistema robusto, que ha sido desarrollado y perfeccionado mediante la evolución del modelo de las redes neuronales. Las primeras redes neuronales eran capaces de clasificar patrones sencillos, y las actuales pueden distinguir patrones complejos.

2.8. OCR

Un OCR es un software que tiene la capacidad de clasificar los caracteres de texto de un determinado alfabeto y digitalizarlos para su posterior procesamiento.

Actualmente cumplen tareas como digitalizar hojas de texto, licencias vehiculares, soporte para invidentes, etc..

Existen librerías específicas para realizar esta tarea como MODI o GOCR, que se puede utilizar mediante el lenguaje C/C++. El proceso consiste en segmentar la imagen, aplicar operaciones morfológica y comparación para su clasificación. La eficacia del sistema radica en la calidad de la imagen.

2.9. QT CREATOR

Es un IDE creado por Nokia para crear entornos gráficos. Su editor nativo es de C++ pero también soporta lenguajes como Pascal, Phyton, C#. De fácil utilización y multiplataforma ha sido utilizado para crear aplicaciones como VirtualBox, Google Earth y Skype.

Puede trabajar con las librerías de procesamiento de imágenes como OpenCV sin ningún problema.

2.10. ARDUINO MEGA

El Arduino Mega es una tarjeta de desarrollo de hardware libre, basado en el microcontrolador ATmega1280. Posee conexión USB, un cristal de 16Mhz y un puerto de alimentación adicional.

TABLA II-N°2: CARACTERÍSTICAS DEL ARDUINO MEGA

Microcontrolador	ATmega1280
Voltaje de Operación	5V
Voltaje de Entrada (Recomendado)	7-12V
Voltaje de Entrada (límite)	6-20V
Pines Digitales	54
Pines Analógicos	16
Corriente por Pin Digital	40mA
Flash Memory	128 KB
SRAM	8KB
EEPROM	4KB
Velocidad del Reloj	16MHz

Fuente: http://arduino.cc/en/Main/arduinoBoardMega25

Arduino Mega posee un integrado FTDI FT232RL para la conversión de USB a TTL, que provee un puerto virtual en el computador.

2.11. SELECCIÓN DE LA CÁMARA

Para el desarrollo del proyecto se ha considerado la posibilidad de usar dos tipos de cámaras: el primer tipo ha sido las cámaras Web, y el segundo cámaras IP.

Las cámaras Web presentan la ventaja de un fácil uso en OpenCV, además de una alta calidad de imagen. La desventaja radica en que si se quiere conectar dos cámaras, éstas deben compartir el mismo canal USB que generalmente se satura y se obtiene un error de hardware. Para solucionar este problema se debe considerar el cambio del hardware o disminuir la calidad de la imagen, si el problema persiste se debe probar con el cambio de cámara. El proceso de diagnosticar el problema presenta un gasto enorme de tiempo y recursos, añadido a la falta de información del error hace que se descarte la posibilidad de usar cámaras Web.

Por otro lado las cámaras IP son relativamente difíciles conectar a OpenCV, debido a que necesitan una configuración previa, además de una dirección web específica para cada modelo de cámara. La ventaja radica en que se puede conectar el número de cámaras necesario sin tener problemas de saturación o similares, a costa de ralentizar el sistema.

2.12. INTELIGENCIA ARTIFICIAL

Es la rama de la ciencia que se encarga de resolver problemas usando como modelo la inteligencia humana y que se apoya en otras ciencias como la ciencia de la computación y la filosofía.

En 1956 se acuñó el término inteligencia artificial como: “Es la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligente” (McCarthy, J., 1956).

Dentro de la inteligencia artificial existen otras ramas como son las redes neuronales artificiales, los algoritmos genéticos, y la lógica difusa.

2.12.1. HISTORIA DE LA INTELIGENCIA ARTIFICIAL

La historia comienza con Aristóteles, en el año (384 a 322 a. C.) que describió un conjunto de leyes que regían la parte racional de la inteligencia, que le permitió desarrollar un sistema que extraía conclusiones a partir de premisas iniciales.

Posteriormente el siguiente avance significativo fue el modelo de redes neuronales presentado en 1943 por McCulloch y Pitts. Pero fue con la llegada de Alan Turing que en verdad se comenzaron a ver cambios relevantes en la inteligencia computacional.

La primera red neuronal que se basa en el diseño de la neurona biológica fue la red perceptrón, creada por Rosenblatt en 1959. A partir de este año la inteligencia artificial sufrió un abandono parcial debido a las limitaciones que tenían las redes del momento.

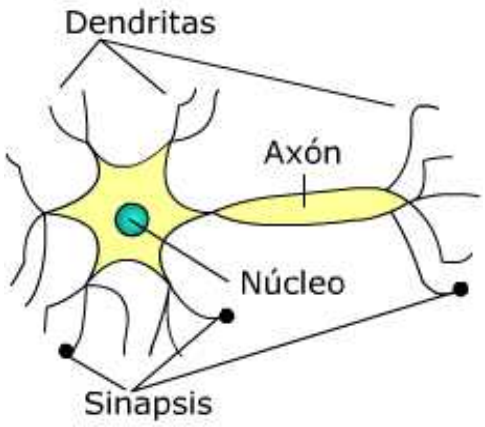
Con la publicación de Parallel Distribute Procesing vuelve la inteligencia artificial y toma impulso, creando un sinnúmero de aplicaciones en las diferentes ramas de la ingeniería. Actualmente ya se ha superado la prueba de Turing, que menciona que existirá inteligencia artificial cuando no seamos capaces de distinguir si estamos conversando con una máquina o una persona.

2.12.2. REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales tratan de emular el funcionamiento del cerebro humano, creando conexiones entre unidades de procesamiento denominadas neuronas. La unidad central de la red es una neurona artificial que ha sido modelada matemáticamente a partir de una neurona biológica.

Las dendritas y la sinapsis proveen el medio para conectar la neurona con sus vecinas, se ha encontrado que una neurona puede estar conectada con una gran cantidad de neuronas, de capas diferentes y que además de intercambiar información intercambian otro tipo de proteínas. En el núcleo se procesa la información recibida y se prepara para propagar o inhibir la información analizada mediante el axón.

FIGURA II-N°8: NEURONA BIOLÓGICA

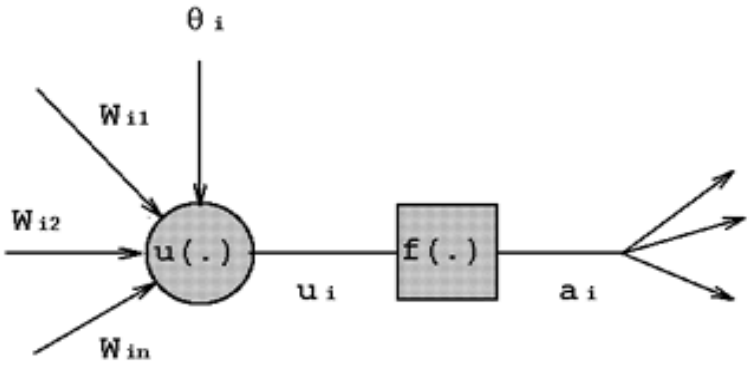


Fuente: http://www.virtual.unal.edu.co/cursos/ingenieria/2001832/lecciones/ cap_4/images/neurona.jpg

El aprendizaje biológico consiste en fortalecer las uniones de las neuronas involucradas en el proceso. Cuantas más neuronas sean involucradas mejor serán los resultados.

Por otra parte la neurona artificial cuenta con similares características aunque a un nivel limitado de información y número de conexiones. La neurona posee entradas con sus respectivos pesos, los mismo que simularán el proceso de fortalecimiento de los enlaces mediante un valor numérico. Éste número puede ser negativo si se trata de inhibir la entrada.

FIGURA II-N°9: NEURONA ARTIFICIAL



Fuente: http://www.virtual.unal.edu.co/cursos/ingenieria/2001832/lecciones/c ap_4/images/escalon.gif

Es conveniente acotar la salidas de la neurona mediante funciones de transferencia como la tangente hiperbólica y sigmoidal.

El esfuerzo por simular la inteligencia humana radica en varias ventajas como la solución de problemas no lineales, la distribución de la información en cada una de las neuronas, de manera que si una parte de la red es afectada, el rendimiento se ve disminuido pero el sistema no colapsa totalmente. Otra característica a destacar es que la misma estructura de red, se puede utilizar para solucionar diversos problemas, sin necesidad de cambiar el código fuente.

La desventaja de las redes es que necesitan de una gran cantidad de muestras para su entrenamiento, también la necesidad de varias horas para el proceso de aprendizaje.

Debido a las características que presenta la red neuronal la gama de aplicaciones es muy grande y abarca temas como el procesamiento de señales, filtros de ruido, procesamiento de lenguaje, reconocimiento de patrones, entre otros.

2.12.3. TIPOS DE APRENDIZAJE

Existen dos tipos de aprendizaje: el supervisado y no supervisado. El aprendizaje supervisado consiste en mostrar a la red un vector de entrada y su correspondiente vector de salida. Los pesos de la neurona se ajustarán dependiendo del error generado en la salida.

En el aprendizaje no supervisado los datos de entrada son ingresados a la red, y los mismo son tomados como valores aleatorios. Se busca similitudes entre los datos y se irán auto organizando. En este tipo de entrenamiento no se necesita especificar a la red un vector de salidas deseadas.

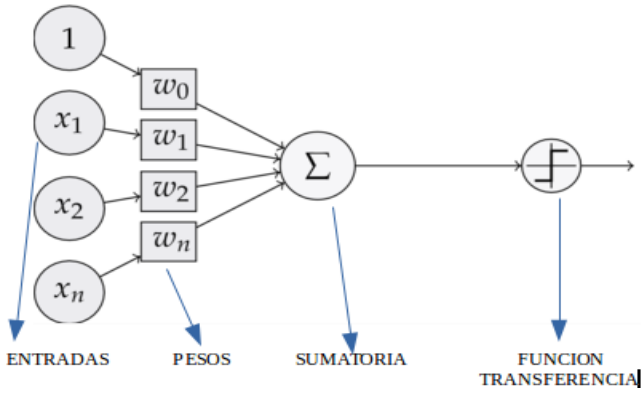
2.12.4. REDES DE APRENDIZAJE SUPERVISADO

Las redes de aprendizaje supervisado son las redes perceptrón, adaline, madaline y retropropagación.

2.12.4.1. PERCEPTRÓN

Es el primer modelo de red creada y esta basada en el modelo de una sola neurona. Puede tener hasta dos capas de neuronas pero solo se puede ajustar los pesos de la capa de entrada. Los modelos a clasificar se representan mediante vectores cuyos valores son 0 o 1 e igualmente las salidas del perceptrón son valores entre 0 y 1.

FIGURA II-N°10: PERCEPTRÓN



Fuente: Los Autores

La salida de la neurona se calcula como la sumatoria de las entradas multiplicada por sus respectivos pesos (ecuación 1.3). Posteriormente se aplica una función de transferencia y se obtiene la salida en valores de 1 y 0.

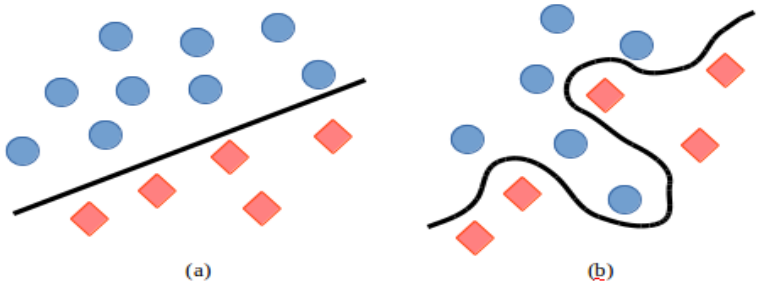
$$S = \sum_{i=0}^{n} x_i * W_i \qquad \textbf{(1.3)}$$

Debido a que la red no puede ajustar los pesos de la capa intermedia, tiene la dificultad de clasificar patrones nolinealmente separables (Figura I-N°10.b). Esta es la principal limitante de la red perceptrón detallada en la publicación Perceptrons por Minsky y Paper en 1969.

La separabilidad lineal consiste en clasificar patrones mediante una línea recta (Figura I-N°10.a). Operaciones básicas como AND y OR son patrones linealmente separables,

operaciones como XOR no lo son, por lo que el perceptrón es incapaz de realizar esta función.

FIGURA II-N°11: SEPARABILIDAD LINEAL Y NO SEPARABILIDAD LINEAL



Fuente: Los Autores

Los usos de la red perceptrón es la clasificación de patrones sencillos y la predicción.

2.12.4.2. ADALINE

La estructura de la red Adaline básica es similar a la perceptrón, salvo a la función de transferencia de la neurona que es la función umbral. Luego de realizar la sumatoria de las entradas por su pesos se aplica la siguiente función de umbralización:

$$1 \, si \, S = \sum X_i * W_i \geq 0 \qquad \textbf{(1.4)}$$
$$-1 \, si \, S = \sum X_i * W_i < 0$$

Esta función limita la salida a valores de 1 y -1. La regla de aprendizaje es similar a la de perceptrón y ajusta sus pesos en función del error de salida.

Las principales aplicaciones de la red adaline es la asociación de patrones linealmente separables, filtros de ruido y filtros adaptativos.

2.12.4.3. MADALINE

La red madaline posee dos capas, una capa de entrada y una capa de salida. La primera capa esta formada por neuronas del tipo adaline. A pesar de contar con dos capas, solamente puede ajustar los pesos de las neuronas adaline. Los pesos de la capa de salida tiene un valor igual a 1, y cada neurona entrega su valor de +1 o -1 a la capa siguiente.

La salida de las neuronas se calculan mediante una regla de mayorías, si más de la mitad de neuronas adaline tiene una salida de 1, la salida total del sistema será 1. Caso contrario la salida será -1.

Los principales usos son filtros y eliminación de ruidos en señales portadoras.

2.12.4.4. RETROPROPAGACIÓN

La red de retropropagación es una red multicapa, que cuenta con un algoritmo efectivo para ajustar los pesos de las capas intermedias. Esto permite crea una red con un número infinito de capas, aumentando la capacidad de aprendizaje de la red.

La red puede resolver exitosamente los problemas linealmente no separables por lo que constituye un gran avance en la ciencia computacional. La red de retropropagación fue la red que impulsó el actual desarrollo de la inteligencia artificial. Fue presentado por Paul Werbos en 1974.

Las aplicaciones de la red de retropropagación es la clasificación, reconocimiento de patrones, entre otras.

CAPÍTULO III

3. DISEÑO DE LA RED NEURONAL ARTIFICIAL

En el presente capítulo se determinará la red neuronal más adecuada para nuestro trabajo, y se detallarán las características más relevantes. Se escogerán las muestras para su entrenamiento y se codificará la RNA.

3.1. SELECCIÓN DE LA RED NEURONAL

Para la selección de la red neuronal se utilizará el análisis documental. Las características de las redes se han detallado en el capítulo II, sección 2.12.4 y serán analizadas.

La red seleccionada debe ser capaz de reconocer patrones alfanuméricos complejos, presentes en las licencias vehiculares. Se estima que cada carácter debe tener más de 500 entradas, por lo que la cantidad de datos a entrenar serán bastante extensas.

Las características más relevantes están detalladas en la Tabla III-N°3.

Debido a la naturaleza de nuestro trabajo, se ha optado por la red de retropropagación. Esta red presenta características favorables como la gran cantidad de datos con los que pueden trabajar, el número variable de capas, una amplia variedad de funciones de transferencia y la capacidad de clasificar patrones no lineales.

TABLA III-N°3: CARACTERÍSTICAS DE LAS REDES NEURONALES

TIPO DE RED	APLICACIONES	Número de Capas	Funciones de Transferencia
Perceptrón	Clasificación de Patrones Sencillos, Predicción	2 capas, solo puede ajustar 1 capa	Tangente hiperbólica, sigmoidea. umbralización
Adaline	Filtros de Ruido, Filtros Adaptativos	2 capas, solo puede ajustar 1 capa	Umbralización
Madaline	Filtros, Eliminación de ruido	2 capas, solo puede ajustar 1 capa	Umbralización
Retropropagación	Clasificación patrones complejos, minería de datos	N capas	Tangente hiperbólica, sigmoidea. umbralización

Fuente: Los Autores

3.2. ESTRUCTURA DE UNA RED RETROPROPAGACIÓN

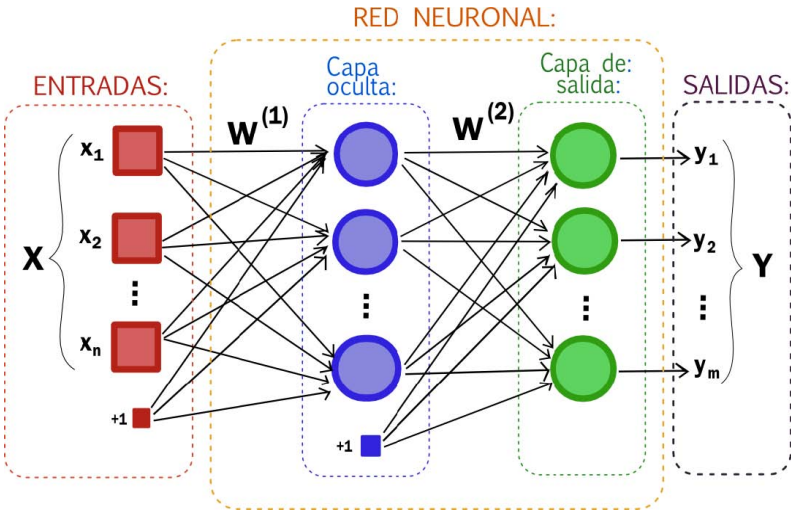
Las redes de retropropagación están formadas por varias capas, las capas más externas son las capas de entrada y salida, y las capas llamadas ocultas son aquellas que ocupan una ubicación intermedia, pudiendo tener el número necesario de capas según la cantidad de datos a aprender.

Cada una de estas capas puede poseer el número de neuronas que sean requeridas para la aplicación, así como tener diferentes funciones de activación en cada neurona. Es importante mencionar también que debe existir en cada capa, exceptuando la capa de salida, una entrada cuyo valor sea igual a 1, denominado frecuentemente como bias.

Todas las entradas de la red neuronal deben estar conectadas con cada una de las neuronas de la capa de entradas mediante sus respectivos pesos, esto quiere decir que al

existir un número *n* de entradas, habrá *n* pesos que correspondan a la capa de entradas por neurona.

FIGURA III-N°1: RED MULTICAPA



Fuente: http://ceres.ugr.es/~alumnos/esclas/imagenes/index_img_4.jpg

Se realiza una sumatoria del producto de los valores de entrada por sus respectivos pesos, el resultado ingresa a la función de activación de la neurona y se obtiene la salida de una neurona (Figura III-N°2).

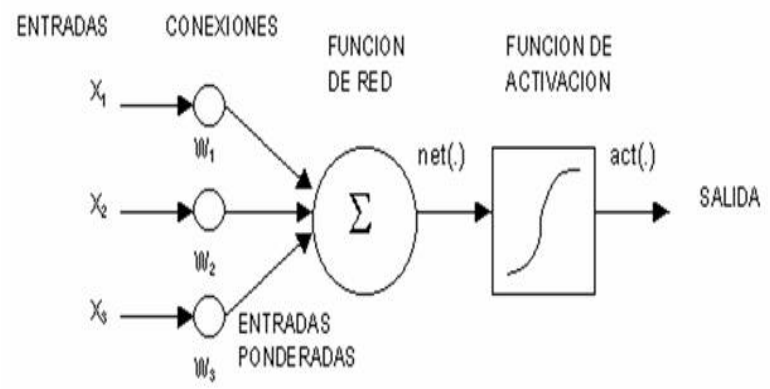
De igual manera, las salidas de las neuronas ya sea de las capas de entrada o intermedias deben estar todas conectadas a cada una de las neuronas de la capa siguiente.

Las entradas a la red neuronal serán vectores que contengan toda la información necesaria, sin existir restricciones de los valores que pueda poseer. También deberá existir una entrada cuyo valor sea 1, que será el bias de entrada. Habrá tantos vectores de entrada como número de datos a entrenar, que serán mostrados a la red para su proceso de aprendizaje o clasificación.

Cada vector de entrada, deberá contar con su respectivo vector de salida, según dicta el modelo de aprendizaje supervisado. El valor esperado en las capas de salidas tendrá una correspondencia con la función de activación en la capa de salida. Se recomienda que el

número de neuronas en la capa de salida sea acorde al número de elementos a clasificar.

FIGURA III-N°2: NEURONA Y COMPORTAMIENTO



Fuente: http://www.angelfire.com/fang/sencillito/index_archivos/image005.jpg

Las dificultades que presenta este tipo de red es que no existe una regla a seguir para determinar el número de capas o neuronas que ésta deba tener para cumplir su objetivo. La elección del tamaño de la red deberá ser determinado mediante un proceso de prueba y error. Debe entenderse como error un deficiente rendimiento de la red, o la no convergencia.

En el caso de que la red presente un funcionamiento deficiente, la solución es cambiar el tipo de función de activación, o aumentar el número de muestras para mejorar la precisión de la red. En el caso de que ocurra la no convergencia es recomendable aumentar el número de neuronas o capas, hasta alcanzar la convergencia. Un error que se debe evitar es que las muestras a clasificar sean muy diferentes entre sí.

Debido a que una red neuronal describe el funcionamiento aproximado de una red biológica, debe esperarse resultados que contengan un cierto margen de error, pues a diferencia de la creencia popular la inteligencia artificial todavía tiene limitaciones para emular la inteligencia biológica.

3.3. ALGORITMO DE ENTRENAMIENTO

La red de Retropropagación nos permite ajustar los pesos de las capas de entrada y ocultas de una manera exitosa. El algoritmo de retropropagación consta de una fase de propagación hacia adelante a partir del vector de entrada. La salida generada es comparada con la salida deseada y ese error se retropropaga a través de la red.

El Error de retropropagación en cada una de las neuronas va a depender de los pesos en cada una de las capas, y a su vez servirá para ajustar sus pesos. El algoritmo de ajuste de pesos presenta reglas distintas para ajustar los pesos en las capas ocultas y en la capa de entrada.

La magnitud del cambio de los pesos según el error estará siempre regulada por un factor de aprendizaje, este valor se recomienda ser bajo, en un rango de décimas. Un valor bajo del factor de aprendizaje conlleva a que la red converja, a costa de aumentar el número de épocas. Un valor alto puede volver a la red inestable, y nunca alcanzar el valor adecuado de sus pesos.

De forma simplificada, el algoritmo consta de selección del dato de entrada, propagación hacia adelante, retropropagación de error y ajuste de pesos. Este proceso debe realizarse con todos los datos de entrada el número de veces necesarios hasta que el error en la red esté dentro de un rango aceptable.

3.3.1. PROPAGACIÓN HACIA ADELANTE

Una vez que los datos de entrada sean cuidadosamente seleccionados para el entrenamiento, estos deben descomponerse o presentarse como vectores cuyos valores serán computados en la red neuronal.

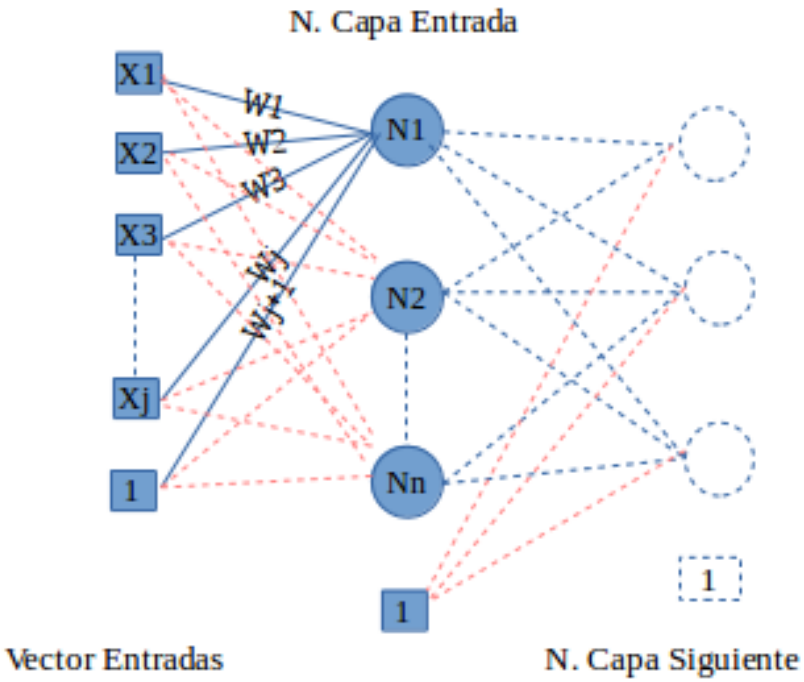
Para iniciar el proceso, se debe inicializar los pesos de todas la capas de forma aleatoria. Éstos pesos preferentemente deben ser inicializados con valores bajos, en el rango de décimas. El valor debe ser aleatorio, considerando también la posibilidad de pesos con valores negativos. Un error a evitar es inicializar sus pesos con valores iguales a cero,

debido a que las fórmulas de ajuste de sus pesos, requieren de un valor inicial que sea diferente de cero.

Una vez ubicados los valores de entradas, se calcula el valor de las salidas de la primera capa por cada neurona, éstas salidas se convierten en entradas de la segunda capa, y así sucesivamente hasta llegar hasta la capa de salida.

Para explicar de una manera más clara el algoritmo de propagación, consideremos una red cuya entrada sea una vector con *j* datos de entrada y una capa de entrada con *n* neuronas.

FIGURA III-N°3: CAPA DE ENTRADA



Fuente: Los Autores

Cada dato del vector de entrada se encuentra conectado mediante su respectivo peso a cada neurona de la capa siguiente. El vector de pesos representado por W tendrá la misma dimensión que el vector de entrada más el 1 del bias. El valor de la activación de la neurona será calculado mediante la sumatoria del producto de las entradas por sus

pesos.

La sumatoria de las entradas de la neurona N_i se calcula como:

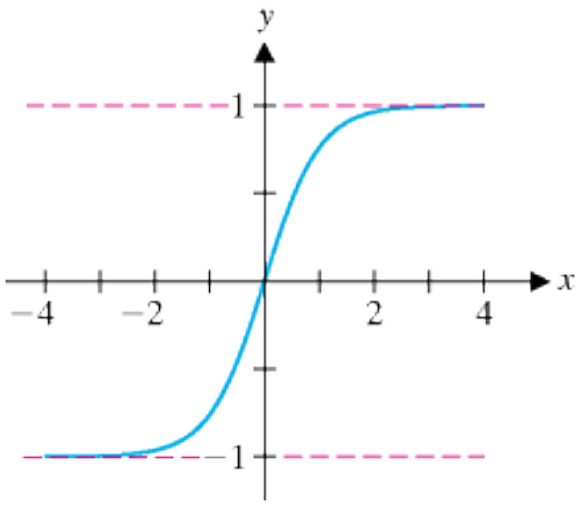
$$S_i = \sum_{i=1}^{i=j+1} Xi \cdot Wi \qquad \textbf{(3.1)}$$

Donde W es el vector de pesos correspondiente a la capa de entrada. La sumatoria obtenida, genera una salida igual a:

$$Sal_i = G(S_i) \qquad \textbf{(3.2)}$$

La función G es la función de activación de la neurona. La función escogida para el presente trabajo fue la función de tangente hiperbólica, debido a que en la práctica presentaba menores tiempos de procesamiento, así como una convergencia más rápida con un número menor de épocas. La función de activación tiene como objetivo acotar los resultados en la salida de la neurona, para evitar en redes de mayor magnitud la aparición de números infinitos debido al desborde de la memoria asignada para valores decimales.

FIGURA III-N°4: TANGENTE HIPERBÓLICA



Fuente:http://www.mhhe.com/math/calc/smithminton2e/cd/folder_structure/text/chap06/section09/figure_0646b.gif

Los valores obtenidos en las neuronas de la capa de entrada, se convierten automáticamente en entradas para las capas subsiguientes, y el proceso continúa hasta llegar a la capa de salida.

3.3.2. RETROPROPAGACIÓN

Una vez calculadas las salidas de la red, se las compara con el vector de la salida deseada para poder encontrar el error. El error se obtiene mediante la diferencia de la salida deseada y la salida calculada, siendo *i* el número de salidas se tiene que el vector de error se calcula como:

$$Es_i = T_i - Sal_i \qquad \textbf{(3.3)}$$

Donde T es el vector que contiene los valores ideales de la salida para la muestra de entrada.

El paso posterior consiste en retropropagar el error, desde la capa de salida hasta la capa de entrada, mediante el producto de sus respectivos pesos. Tomando una red de ejemplo con *j* salidas, y con una capa anterior de *n* neuronas (Figura III-N°5), el error retropropagado se calcula de la siguiente manera:

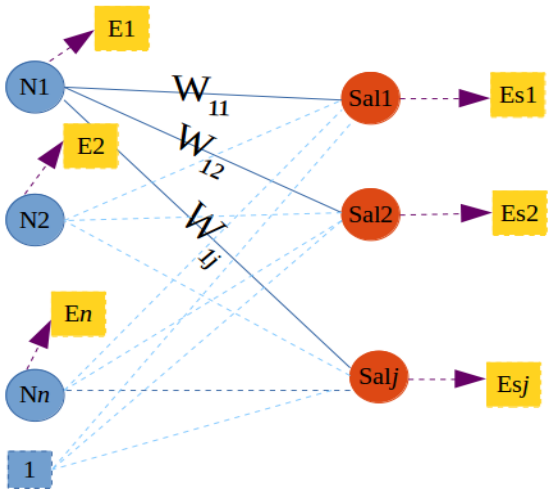
$$E_i = \sum_{k=0}^{k=j} Es_k * W_{ik} \qquad \textbf{(3.4)}$$

3.3.3. APRENDIZAJE

Luego de haber retropropagado el error y que cada neurona tenga su respectivo error, comienza el proceso de aprendizaje o corrección de los pesos. Existen diferentes fórmulas para ajustar los pesos de las capas ocultas y los pesos de las capas de entrada.

Para minimizar el error en la salida, el algoritmo se basa en el cálculo del gradiente del error. El gradiente tiene como objetivo indicar el sentido en el que el campo escalar del error, varía más rápidamente.

FIGURA III-N°5: RETROPROPAGACIÓN DEL ERROR



Fuente: Los Autores

El gradiente será calculado en base a la función de transferencia de la neurona. Para una función de activación del tipo tangente hiperbólica, se tiene un gradiente:

$$\nabla \tanh(E) = \frac{1}{\cosh^2(E)} \qquad \textbf{(3.5)}$$

La fórmula para actualizar los pesos en las capas intermedias u ocultas es la siguiente:

$$W_{i+1} = W_i + n * E_i * Sal_i \frac{1}{\cosh^2(E_i)} \qquad \textbf{(3.6)}$$

La fórmula **(3.6)** involucra a las salida en propagación hacia adelante de la neurona (Sal_i), el error hallado por la retropropagación E_i , *n* como factor de aprendizaje, y el gradiente de tangente hiperbólica ($1/\cosh^2(E_i)$).

Como se mencionó anteriormente, el factor de aprendizaje influye mucho en el aprendizaje de la red. Un valor bajo, atenúa la modificación de los pesos, aumentando el número de épocas en el entrenamiento, pero asegurando una convergencia de la red. Un valor alto, causa una oscilación en el valor de los pesos, lo que provocará que el error

nunca se minimice y la red nunca presentará las salidas deseadas.

El valor que mejor desempeño alcanzó en nuestra aplicación, fue un valor de 0.01. Se probaron valores como 1 o 0.5, pero a lo largo del tiempo, no se alcanzaba el correcto funcionamiento de la red.
Para el ajuste de los pesos en la capa de entrada, se tiene la siguiente fórmula:

$$W_{i+1} = W_i + n * E_i * X_i \qquad \textbf{(3.7)}$$

Los pesos en la capa de entrada, dependen del valor del vector de entrada (X_i).

3.3.4. CONVERGENCIA

La convergencia es el estado en el que la red ha aprendido todos los patrones de entrada, y tiene un margen de error aceptable al propagar los patrones de entrenamiento. A una red le puede tomar varias horas llegar a aprender, y cientos o miles de épocas. Cuánto más grande sea la base de datos para el entrenamiento se requerirá una red más grande.

Es una buen práctica que se comience a entrenar la red con pocas muestras pues asegurará la convergencia, e ir observando el tiempo que le toma a la red aprender. Esta estimación nos ayudará a tener una proyección acerca de cuánto será el tiempo que le tome a la red aprender todas las muestras de entrenamiento.

En la práctica se pudo observar el comportamiento de la red frente a la cantidad de patrones a aprender. Cuando el número de patrones era bajo, la red convergía en pocas iteraciones, y en tiempos relativamente bajos. Por otro lado, cuando el número de muestras aumentaba, el número de iteraciones aumentaba, al igual que el tiempo de aprendizaje.

El número de muestras es proporcional a la precisión que se desee obtener. En el desarrollo del trabajo se observó que la red aumenta su grado de acierto conforme aumentaba el número de patrones para su entrenamiento.

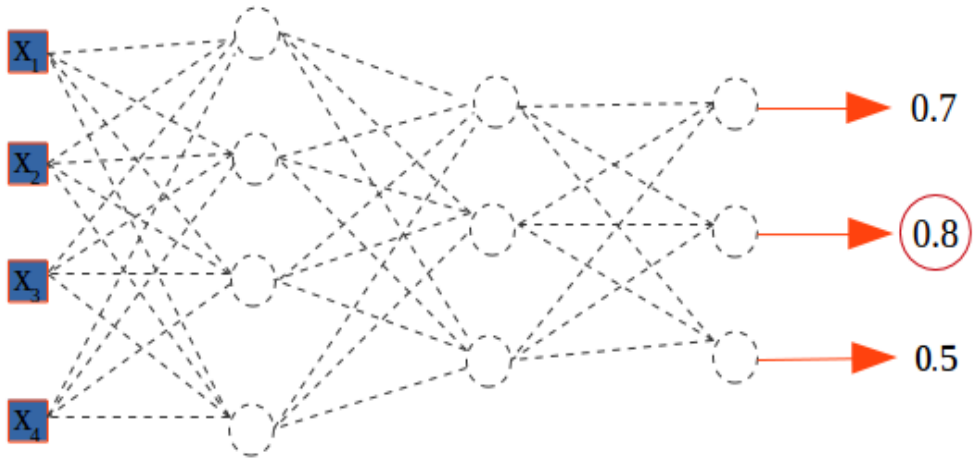
3.3.5. CLASIFICACIÓN

Una vez que se cumpla con la meta del entrenamiento, la red estará lista para entrar en funcionamiento. Es muy importante, una vez finalizado el ajuste de pesos, que los mismos sean almacenados, para evitar que la red entrene cada vez que se ejecute nuestro programa. Se han obtenido excelentes resultados al almacenar los pesos en archivos te texto plano con formato .txt, por la facilidad de manejo de los archivos y por los bajos tiempos de lectura y escritura. Al iniciar el programa, los pesos de entrenamiento deben ser leídos para poner a punto la red.

Cuando exista un dato a clasificar, el mismo deberá ser sometido al proceso de preparación y adecuación previo al ingreso a la red. Ésta entrada generara valores de salidas de acuerdo al valor fijado en los pesos. El proceso de clasificación consiste en encontrar el mayor valor presente en el vector de salida (Figura III-N°6).

Existe la posibilidad de que se presenten más de una salida que posea el valor máximo. En ese caso se debe suponer que la red no ha tenido éxito en clasificar los patrones de entrada. Se puede buscar una solución al problema mediante la utilización de los recursos brindados por la biblioteca de OpenCV para el tratamiento de imágenes.

FIGURA III-N°6: CLASIFICACIÓN



Fuente: Los Autores

Se puede usar técnicas como la erosión, dilatación o esqueletonización del patrón de entrada, para que de esta manera aumentar las probabilidades de éxito en su clasificación.

3.4. CODIFICACIÓN DE LA RED DE RETROPROPAGACIÓN

A continuación se explicará a detalle los aspectos más relevantes de la codificación de la red neuronal en en lenguaje C++. El proyecto fue desarrollado bajo el sistema operativo Linux-Ubuntu, que difiere en cierta medida con la codificación en Windows, en especial con el nombre de las librerías a utilizar y algunas funciones propias del sistema.

Como editor y compilador se utilizó el programa Qt Creator, de distribución gratuita y puede ser descargada de su página web oficial. Para visión artificial se usaron las librerías de OpenCV, igualmente gratuitas.

Cuando la red neuronal adquiere un tamaño considerable, es preferible tener a mano un computador cuyo procesador sea de última generación, pues conforme la red crece, aumenta el número de cálculos.

3.4.1. SELECCIÓN DE DATOS DE ENTRENAMIENTO

Un factor muy importante que desencadenará en el éxito o fracaso de la red es la correcta selección de los datos para su entrenamiento. El objetivo de la red neuronal será identificar los caracteres alfanuméricos presentes en las placas automovilísticas.

Por tal motivo, se necesitará una base de datos con imágenes tanto de números como letras mayúsculas. Luego de una exhaustiva búsqueda y análisis de muestras, se seleccionó una base de datos gratuita disponible en internet, que puede ser descargada mediante el enlace:

http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/EnglishFnt.tgz

Las muestras escogidas aseguran una gran variedad de tipos de letras por cada caracter. Además la numeración con la que cuentan las imágenes facilitan la creación de un código único para el tratamiento de todas las imágenes sin necesidad de modificar la dirección de la fuente.

3.4.2. TRATAMIENTO DE LA IMAGEN

Una vez que la imagen sea leída, ésta debe ser tratada para poder representarse como un vector de entrada adecuado para la red. Una imagen matemáticamente se representa como una matriz de puntos, en cada punto puede poseer un canal si la imagen está en escala de grises, o tres canales si la imagen presenta colores rojo, verde y azul.

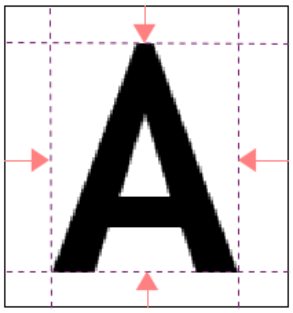
Debido a que la mayoría de las funciones para el tratamiento de imágenes funcionan con la imagen en escala de grises, será necesario pasar nuestra imagen a color, a escala de grises.

FIGURA III-N°7: IMAGEN EN ESCALA DE GRISES

Fuente:http://docs.opencv.org/_images/Histogram_Calculation_Theory_Hist0.jpg

Las imágenes escogidas, están rodeadas de espacios blancos que deber ser eliminados mediante un proceso de barrido en el eje horizontal y vertical de la imagen a fin de desechar características no deseadas de la imagen.

FIGURA III-N°8: RECORTE DE IMAGEN



Fuente: Los Autores

La imagen recortada, debe pasarse a una imagen en escala de grises y posteriormente debe ser binarizada. yo le voy a mandar solo los que son para grabar

Debido a los diferentes tamaños presentes en las muestras se ha decido trabajar con un tamaño estándar de las imágenes. La nueva dimensión de la imagen debe evitar la pérdida de información. Luego del análisis del tipo de muestras, se llegó a la conclusión de que un tamaño de 25x25 píxeles representa correctamente una imagen sin el deterioro y pérdida de información (Figura III-N°9).

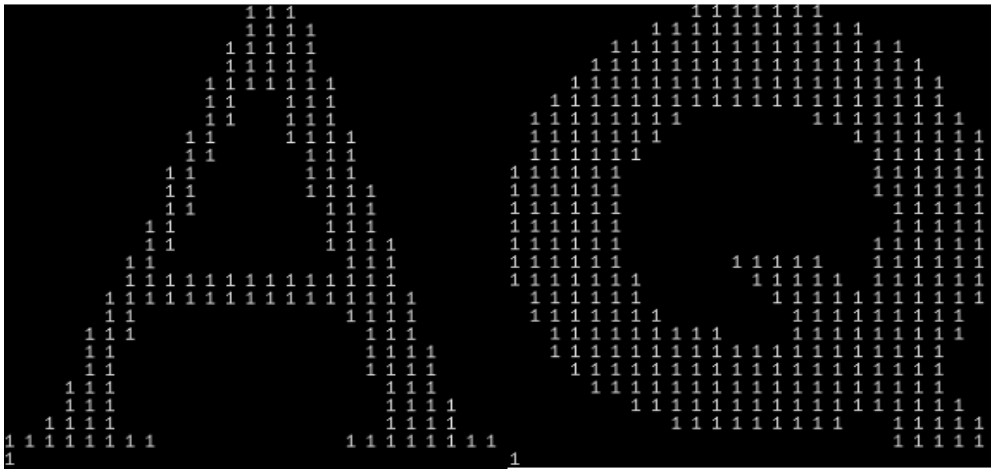
El vector de entrada se formará a partir de la imagen redimensionada y binarizada. El vector estará formado por 626 posiciones de la matriz de 25x25 más la entrada igual a 1 del bias, que ocupará la última posición.

FIGURA III-N°9: MUESTRAS REDIMENSIONADAS

Fuente: Los Autores

El vector de entrada tendrá valores comprendidos entre cero y uno, siendo cero un píxel en blanco y uno un píxel de color negro. Para generar el vector de entrada se debe hacer un barrido por las filas de la imagen e ir almacenando los datos secuencialmente en el vector.

FIGURA III-N°10: MUESTRAS DE ENTRENAMIENTO



Fuente: Los Autores

En la Figura III-N°10, se ha omitido la impresión de los valores iguales a cero para poder visualizar de mejor manera las letras. Como se puede observar el nivel de detalle es bastante aceptable, por lo que aseguramos datos de calidad para el entrenamiento.

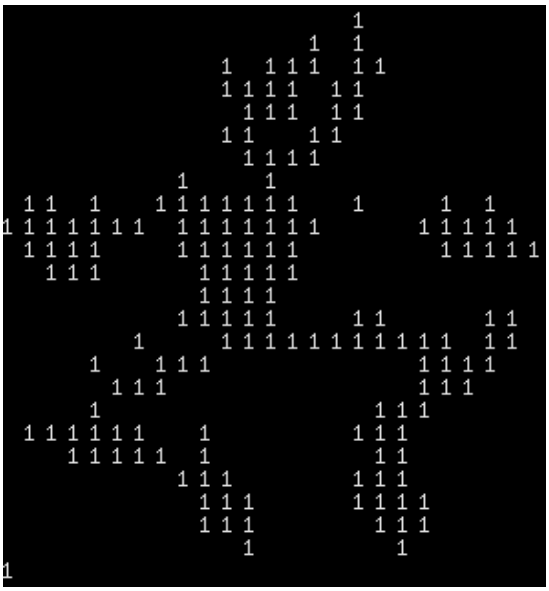
De la base de datos escogida se deben obviar datos que sea muy diferentes a las muestras de entrenamiento, pues se puede provocar que la red no converja debido a la presencia de datos muy diferentes.

FIGURA III-N°11: MUESTRA NO ADECUADA

Fuente: Los Autores

A más de los caracteres alfanuméricos, también debe ser entrenada la red a que aprenda a identificar imágenes que no sea parte del conjunto objetivo. Para ello se deben escoger imágenes aleatorias, con el fin de que la red sepa excluir muestras que no correspondan al conjunto de interés.

FIGURA III-N°12: MUESTRA ALEATORIA



Fuente: Los Autores

Como objetivo se tiene que la red sea capaz de aprender 200 tipos diferentes de letras por cada uno de los caracteres alfanuméricos y datos aleatorios, dando 37 tipos de caracteres a clasificar. Cada muestra posee 626 datos de entrada, lo que genera un total de 4'632.400 datos.

3.4.3. CODIFICACIÓN Y DIMENSIONAMIENTO DE LA RED

Se posee como información preliminar la cantidad de datos que debe aprender la red, por lo que se apuntó a la creación de una red de dimensiones relativamente altas.

Se consideró como una primera estimación una red de tres capas, con 500 neuronas de entrada, 100 neuronas en la capa intermedia y 37 en la capa de salida. Los pesos de sus

respectivas capas se han representado mediante matrices, las filas de la matriz representará a los pesos y las columnas representarán el número de neuronas.

FIGURA III-N°13: DECLARACIÓN DE PESOS DE RNA

```
double pesos1[num_entradas][neu_capa1];
double pesos2[neu_capa1+1][neu_capa2];
double pesos3[neu_capa2+1][neu_capa3];
```

Fuente: Los Autores

La matriz pesos 1, pertenece a su capa 1, como constantes se define el número de entras y el número de neuronas. Debido a que los datos de entrenamiento se generaron con el valor a 1 del bias, no es necesario adicionarla, cosa que no sucede con los pesos de las capas 2 y 3, donde se puede observar (Figura III-N°13) que se adiciona el número 1 del bias.

Previo a la realización del aprendizaje es necesario leer el archivo de entrenamiento en donde se encuentras los datos de entrada. Los datos serán almacenados en una matriz, cuyas filas representa cada uno de los datos del caracter, y las columnas representan los diferentes caracteres.

Es necesario también crear vectores para almacenar los valores de las salidas y además la matriz de salidas deseadas, así como también vectores para almacenar los errores retropropagados y definir el factor de aprendizaje de la red (Figura III-N°14).

En el algoritmo de propagación hacia adelante, se ha escogido un error aceptable de la red del 10% (ecuación 3.8), para ello se usa una función de umbral a la salida con valores de 0.8 y -0.8. Los valores deseados tienen un valor de 1, y el resto de salidas un valor de -1, por lo tanto la distancia entre ambos valores es igual a 2.

$$E = \frac{1 - 0.8}{2} * 100 = 10 \qquad \textbf{(3.8)}$$

FIGURA III-N°14: VECTORES DE SALIDA Y MATRIZ DE ENTRADA

```
double entradas[num_entradas][num_datos];//Matriz a entrenar

double salida1[neu_capa1+1];//[neuronas c1+1bias]
double salida2[neu_capa2+1];//[neuronas c2+1bias]
double salida3[neu_capa3+1];//[neuronas c3+1bias]

double error1[neu_capa1];//[neuronas capa 1][capas]
double error2[neu_capa2];//[neuronas capa 2][capas]
double error3[neu_capa3];//[neuronas capa 3][capas]

double factor_aprend=0.01;  //factor aprendizaje
```

Fuente: Los Autores

En la propagación debe calcularse el error, para posteriormente retropropagarlo en el caso de que exista. La función de propagación hacia adelante depende del patrón de entrada y esa será la variable de entrada de la función (Figura III-N°15).

El código de retropropagación (Figura III-N°16) en la red es similar al de propagación hacia adelante, con ciertas variantes. La función usa el vector de error, por lo tanto no posee patrones de entrada.

El código de aprendizaje de la red (Figura III-N°17) diferencia la capa de entrada del resto de capas. El ajuste de los pesos en la capa de entrada es similar a la red perceptrón, y no necesita el cálculo del gradiente del error.

Para las capas siguientes sin distinción se aplica la fórmula (3.6).

Luego de aplicar el algoritmo al total de muestras escogidas, se pudo observar que la red no convergía, por lo que se decidió aumentar el número de capas a 5, quedando la red con una dimensión de 500 neuronas en la capa de entrada. 100 neuronas en la segunda, tercera y cuarta y finalizando con las 37 neuronas de salida.

FIGURA III-N°15: PROPAGACIÓN HACIA ADELANTE

```
void propagacion_adelante(int var_entrada)
{
//capa1
for(int i=0;i<neu_capa1;i++)  //numero neuronas
{
    suma=0;
    for(int j=0;j<num_entradas;j++) //numero de pesos por entradas
        suma+=entradas[j][var_entrada]*pesos1[j][i];
    salida1[i]=tanh(suma);  //funcion transferencia
}
//capa2
for(int i=0;i<neu_capa2;i++)  //numero neurona
{
    suma=0;
    for(int j=0;j<(neu_capa1+1);j++) //pesos salida1+1bias
        suma+=salida1[j]*pesos2[j][i];
    salida2[i]=tanh(suma);
}
//capa3
for(int i=0;i<neu_capa3;i++)  //numero neurona
{
    suma=0;
    for(int j=0;j<(neu_capa2+1);j++) //pesos salida2+1bias
        suma+=salida2[j]*pesos3[j][i];
    salida3[i]=tanh(suma);

    if(salida3[i]>=0.8)     //valor umbral permite a
        salida3[i]=1.0;     //la red un error del 10%
    if(salida3[i]<(-0.8))
        salida3[i]=-1.0;
    //calculando el error en capa 3
    int aux=var_entrada;
    while(aux>=num_caracteres)  //numero de letras
        aux=aux-num_caracteres; //halla la salida deseada
    error3[i]=sal_deseada[i][aux]-salida3[i];

    }
}
```

Fuente: Los Autores

FIGURA III-N°16: RETROPROPAGACIÓN

```
void retroprop()
{
//Retrop. Error capa2
for(int i=0;i<neu_capa2;i++) //neuronas c2
    {
    suma=0;
    for(int j=0;j<neu_capa3;j++) //neuronas c3
        suma+=pesos3[i][j]*error3[j];
    error2[i]=suma;
    }
//Retrop. Error capa1
for(int i=0;i<neu_capa1;i++) //neuronas c1
    {
    suma=0;
    for(int j=0;j<neu_capa2;j++) //neuronas c2
        suma+=pesos2[i][j]*error2[j];
    error1[i]=suma;
    }
}
```

Fuente: Los Autores

Luego de realizar las pruebas necesarias, la red presentó nuevamente las dificultades de la no convergencia, por lo que se probaron nuevas combinaciones de capas y neuronas.

Luego de una búsqueda exhaustiva se encontró la red ideal para nuestro propósito, y la misma consistía en tres capas, 1000 neuronas en la capa de entrada, 250 neuronas en la capa intermedia y 37 en la capa de salida.

Teniendo 1000 neuronas en la capa de entrada y 626 entradas obtenemos un total de 626.100 pesos en la capa de entrada. Con 1000 neuronas en la capa de entrada más un bias y 250 en la capa intermedia dan un total de 250.250 pesos. De forma similar calculamos el número de pesos en la capa de salida, obteniendo un total de 9.287.

FIGURA III-N°17: APRENDIZAJE

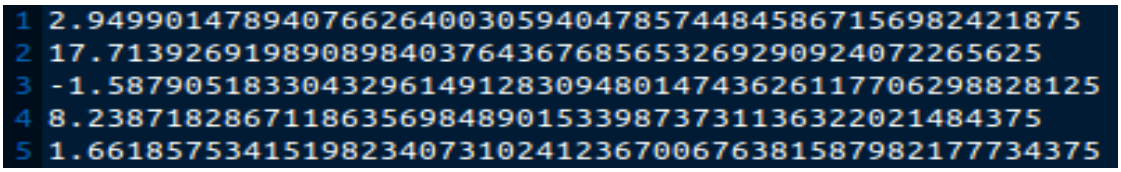
```
void aprender(int variable)
{
    //Ajuste pesos de Capa Entrada
    for(int j=0;j<neu_capa1;j++) //neuronas por capa1
    {
        for(int k=0;k<num_entradas;k++) //pesos Capa entrada+1bias
            {
                pesos1[k][j]+=factor_aprend*error1[j]*entradas[k][variable];
            }
    }
    //Ajuste Pesos Capa 2
    for(int j=0;j<neu_capa2;j++) //neuronas por capa2
    {
        for(int k=0;k<(neu_capa1+1);k++) //neuronas c1+1bias
        {
            pesos2[k][j]+=factor_aprend*error2[j]*salida1[k]/pow(cosh(salida2[j]),2);
        }
    }
    //Ajuste Pesos Capa 3
    for(int j=0;j<neu_capa3;j++) //neuronas por capa3
    {
        for(int k=0;k<(neu_capa2+1);k++) //neuronas c2+1bias
        {
            pesos3[k][j]+=factor_aprend*error3[j]*salida2[k]/pow(cosh(salida3[j]),2);
        }
    }
}
```

Fuente: Los Autores

En cuanto más grande sea la red, mayor será el tiempo que le tome en cada época. Conforme comience a entrenarse la red, irán disminuyendo el número de muestras que la red no pueda clasificar. Con la disminución de los errores, le tomará menos tiempo a la red acabar una época. Entonces las primeras épocas le tomará mayor tiempo a la red que las últimas.

El proceso de entrenamiento le ha tomado a nuestra red un tiempo aproximado de 6 horas, por lo tanto es de vital importancia una vez haya terminado el aprendizaje almacenar los pesos. Para evitar errores se debe asegurar que en los procesos de lectura y escritura se utilicen los pesos con todos sus decimales.

FIGURA III-N°18: VALORES TÍPICOS DE LOS PESOS



Fuente: Los Autores

CAPÍTULO IV

4. DISEÑO E IMPLEMENTACIÓN

En el capítulo se detallará la creación del software a utilizar en el parqueadero tarifario y el ANPR. Se detalla la creación del escenario de pruebas a escala.

4.1. DISEÑO DEL SOFTWARE ANPR

El software ANPR consiste en el reconocimiento de patrones de las placas vehiculares. Tiene como objetivos el seccionamiento de los caracteres de la placa vehicular para posteriormente utilizar un software OCR capaz de reconocer cada uno de los patrones. El OCR a utilizar será la red neuronal diseñada y detallada en el Capítulo III.

Para el tratamiento de la imagen se usarán las librerías dedicadas a la visión artificial brindadas por OpenCV, se analizarán diferentes tipos de algoritmos de seccionamiento así como técnicas para aumentar el grado de aciertos por parte de nuestra red.

4.1.1. SEGMENTACIÓN DE LA PLACA VEHICULAR

El primer paso consiste en segmentar la región de interés que contenga la placa vehicular. Según el nuevo reglamento de la fabricación de matrículas en Ecuador, la placa tiene dimensiones acotadas de 154mm de largo, por 404mm de ancho.

FIGURA IV-N°1: DIMENSIONES DE LA PLACA VEHICULAR

Fuente: http://upload.wikimedia.org/wikipedia/commons/thumb/2/2b/Matr%C3%ADcula_Ecuador_Especificaciones.svg/463px-Matr%C3%ADcula_Ecuador_Especificaciones.svg.png

Para disminuir la información presente en la imagen se ha utilizado una transformación morfológica denominada Black Hat, que consta de la diferencia de la imagen aplicada una operación morfológica de cierre y la imagen original.

La operación de cierre remueve regiones que contienen agujeros pequeños o regiones en negro.

FIGURA IV-N°2: OPERACIÓN MORFOLÓGICA DE CIERRE

Fuente:http://docs.opencv.org/_images/Morphology_2_Tutorial_Theory_Closing.png

Posteriormente se realiza la operación morfológica de gradiente, útil para encontrar el borde de un objeto. Para realizar las operaciones morfológica la imagen debe estar binarizada. Con esto logramos resaltar los lugares en donde exista mayor probabilidad

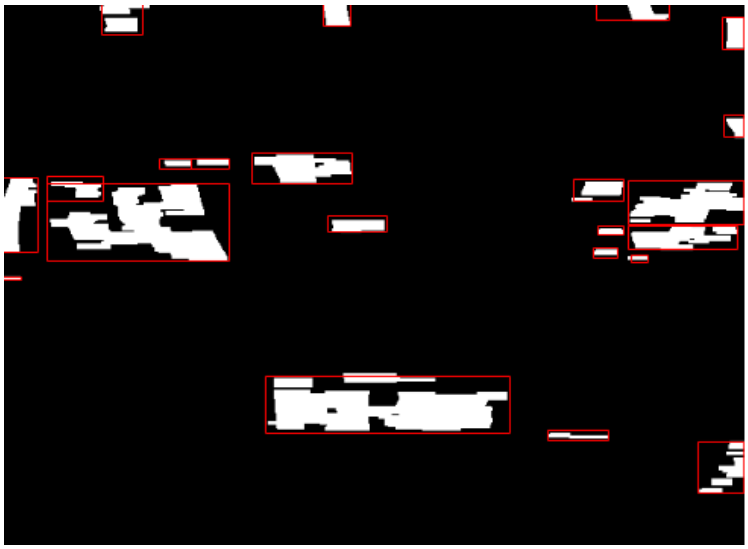
de contener una placa. El resultado final se observa en la Figura IV-N°3.

Para segmentar la placa vehicular se utiliza el método de localización de objetos basado en el algoritmo para detectar blobs, los mismos se puede delimitar en rectángulos que contenga el área de interés (Figura IV-N°4).

FIGURA IV-N°3: OPERACIÓN MORFOLÓGICA DE CIERRE Y GRADIENTE

Fuente: Los Autores

FIGURA IV-N°4: DETECCIÓN DE BLOBS



Fuente: Los Autores

Debido a que en una imagen existirán muchos objetos basura, se deben excluir mediante el uso de información de las características de la imagen objetivo. Un primer filtro a utilizar es la dimensión de la placa.

Las fotografías de las imágenes de muestra de los automóviles fueron tomas en un rango de 1.5m a 2.5m en las que las placas presentan medidas que varían dependiendo de la distancia que se encuentre la placa y la cámara. Mediante varias pruebas, se ha determinado el rango mínimo y máximo de las dimensiones de largo y ancho que presentan las placas en la imagen.

El siguiente filtro será verificar la razón entre el ancho y el largo del objeto:

$$r = \frac{ancho}{largo} = \frac{404}{154} = 2.63$$

Luego de aplicar los filtros, se obtendrá la placa aislada del resto de información. El blob que cumpla las características deberá ser recortado de la imagen original para posteriormente realizar el proceso de reconocimiento de los patrones.

FIGURA IV-N°5: SEGMENTACIÓN DE LA PLACA

Fuente: Los Autores

4.1.2. SEGMENTACIÓN DE LOS CARACTERES DE LA PLACA

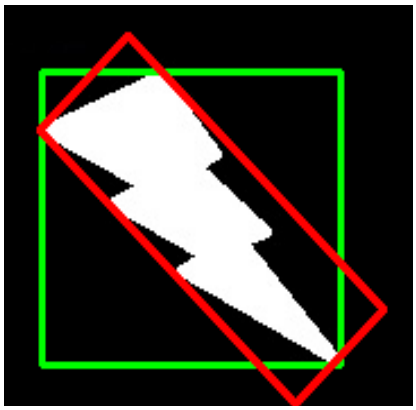
Para poder reconocer los patrones presentes en la placa es necesario aislar cada uno de los caracteres presentes, excluyendo las palabras Ecuador y otros símbolos presentes en la placa.

Para realizar la segmentación se han considerado dos funciones propias de OpenCV y serán contrastadas en función de su rendimiento. La primera función denominada boundingRect() encuentra el rectángulo de menor dimensión que rodea por completo el objeto, el rectángulo tiene un grado de inclinación de 0^0. La segunda función minAreaRect() rodea el objeto pero el rectángulo resultante tiene un ángulo de rotación diferente de cero grados.

Para tener una mejor perspectiva del funcionamiento de las dos funciones, se puede observar en la Figura IV-N°6 en color verde la función boundingRect(), y en rojo la función minAreaRect().

Previo al uso de las funciones detalladas anteriormente, se debe realizar un tratamiento de imagen que consta de los procesos de redimensionamiento, cambio a escala de grises, filtros para eliminar ruido, binarización inversa y por último segmentación. Tal como se realizó con la detección de placas, se usarán filtros de dimensiones para seleccionar exclusivamente los caracteres de las placas.

FIGURA IV-N°6: BOUNDINGRECT() Y MINAREARECT()



Fuente:http://opencv-python-tutroals.readthedocs.org/en/latest/_images/ boundingrect. Png

4.1.2.1. REDIMENSIONAMIENTO

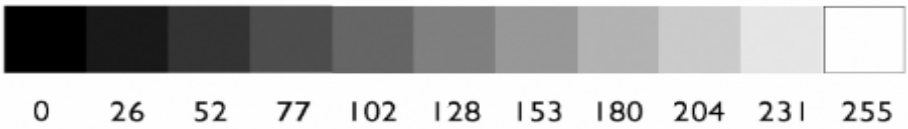
Luego de haber obtenido el recorte de la placa vehicular, es conveniente trabajar con un tamaño estándar de las placas, para que de esta manera se pueda aplicar los filtros de dimensiones y seleccionar así las letras. La función resize() nos permite redimensionar una imagen, las nuevas dimensiones escogidas para la placa han sido de 300x120 píxeles.

Se debe tener cuidado en seleccionar las nuevas dimensiones, debido a que puede distorsionar la imagen o causar pérdida de información.

4.1.2.2. CAMBIO A ESCALA DE GRISES

Los algoritmos a utilizar funcionan con una imagen en escala de grises. Una imagen en escala de grises se representa como una matriz de un sólo canal, o una matriz unidimensional. En cada uno de sus píxeles, presenta valores que van desde 0 el color negro hasta 255 que representa un color blanco.

FIGURA IV-N°7: ESCALA DE GRISES



Fuente: http://www.fotorevista.com.ar/Noticias/Tecnica/08/data/121023194834-1.jpg

La función especializada para el cambio a escala de grises es la función cvtColor() con el argumento CV_BGR2GRAY. El resultado de aplicar la función a nuestra placa se observa en la Figura IV-N°8.

FIGURA IV-N°8: ESCALA DE GRISES EN PLACA

Fuente: Los Autores

4.1.2.3. ELIMINACIÓN DEL RUIDO

Es muy frecuente la presencia de ruido en las imágenes. El ruido es provocado por el polvo, lluvia, distorsiones propias de la cámara u otros factores que influyen negativamente en la imagen. La utilización de filtros lineales nos permiten superar esta dificultad.

El filtro escogido se aplica a cada uno de los píxeles y realiza una suma ponderada de todos los puntos adyacentes. De esta manera es posible aislar los puntos que presenten cambios bruscos y suavizar la imagen.

Como se puede observar en la Figura IV-N°9, mediante el filtro se han atenuado características no deseadas como los puntos de sujeción de la placa y las ondas que cruzan los caracteres. La función blur() nos permite aplicar el filtro.

FIGURA IV-N°9: ELIMINACIÓN DEL RUIDO

Fuente: Los Autores

4.1.2.4. BINARIZACIÓN INVERSA

La binarización inversa es un proceso previo a la utilización de las funciones para detectar objetos, que se basan en la detección de puntos blancos sobre fondos negros. Debido a que los caracteres son negros en fondo blanco, es necesario binarizar la imagen y posteriormente hallar su complemento.

El proceso de binarizar una imagen consiste en reemplazar los valores de los píxeles de la imagen en escala de grises por valores de 0 o 255. Para ello es necesario escoger un valor umbral, donde valores superiores al valor umbral serán asignados un valor de 255 o blanco, y valores inferiores un valor a 0 o negro.

Escoger el valor umbral representará el éxito o el fracaso de la binarización, un valor no adecuado causa la pérdida de información. Éste valor depende mucho de las condiciones de iluminación, por lo que no es recomendable usar valores fijos de umbral.

Existen varios métodos para determinar este valor, uno de ellos es el método de Otsu, que calcula el valor umbral usando métodos estadísticos. El algoritmo presentó un buen rendimiento al momento de realizar las pruebas.

La función de OpenCV que nos permite binarizar la imagen es threshold(), con los argumentos CV_THRESH_OTSU+CV_THRESH_BINARY_INV se aplica el método de Otsu y la binarización inversa.

FIGURA IV-N°10: BINARIZACIÓN INVERSA

Fuente: Los Autores

4.1.2.5. SEGMENTACIÓN MEDIANTE MINAREARECT

Para segmentar las letras y números presentes en las placas, se aplicaron filtros de las dimensiones verticales de las letras, que siempre presentaban un valor constante.

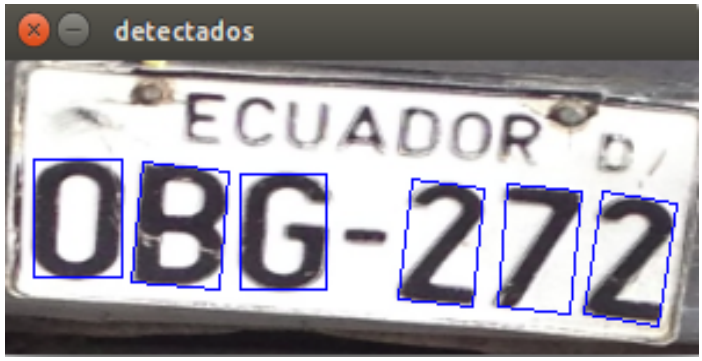
La primera función a analizar será la función minAreaRect(). El resultado de aplicar esta función es un rectángulo que rodea por completo los objetos presentes en la imagen. Lo particular de esta función es que el rectángulo entregado puede tener un ángulo de rotación diferente de cero.

La ventaja consiste en que si la placa tiene algún grado de inclinación, esta inclinación se puede corregir y así aumentar las probabilidades de éxito del reconocimiento del patrón.

En la Figura IV-N°11, se muestra las ventajas de usar el algoritmo minAreaRect(). Como se puede observar la placa tiene un cierto grado de inclinación,

Al momento de realizar las pruebas necesarias, se detectó ciertos problemas con el algoritmo, especialmente en la detección de los números 4 y 1, las letras como la J y Q. En la Figura IV-N°12 se observa una placa sin grado de inclinación, en donde se obtiene un comportamiento no desea con el número 4. Al momento de corregir el grado de inclinación, obtenemos un número 4 distorsionado, que la red neuronal en este caso identifica cómo una letra N.

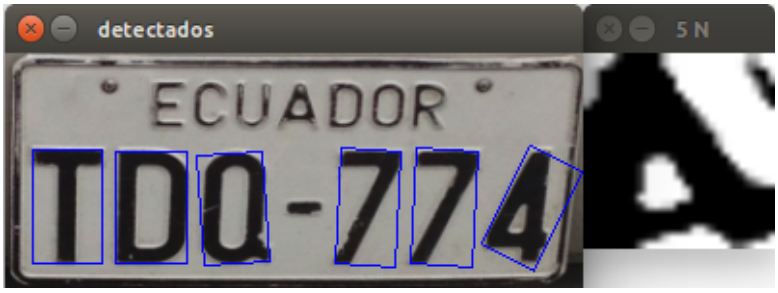
FIGURA IV-N°11: FUNCIÓN MINAREARECT()



Fuente: Los Autores

Debido a esto se optó por descartar esta función, pues de lo contrario se debería modificar nuestra red neuronal para clasificar correctamente el carácter con su nuevo grado de inclinación.

FIGURA IV-N°12: SEGMENTACIÓN DISTORSIONADA

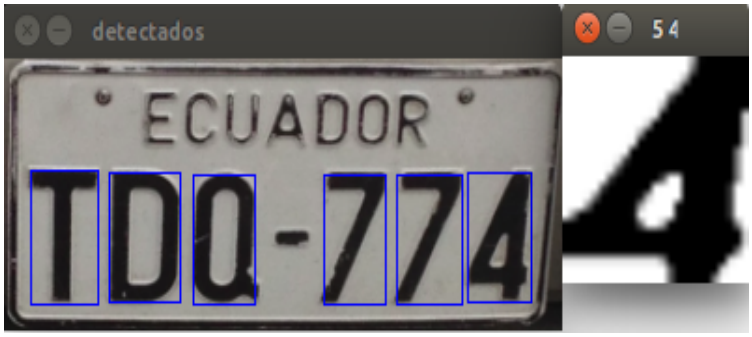


Fuente: Los Autores

4.1.2.6. SEGMENTACIÓN MEDIATE BOUNDINGRECT

El resultado de la función boundingRect() es siempre un rectángulo con una inclinación de 0^0. La obvia desventaja es que si la placa presenta un ángulo de inclinación, este no será corregido y la red puede no detectar el patrón.

FIGURA IV-N°13: SEGMENTACIÓN BOUNDINGRECT()



Fuente: Los Autores

Después de varias pruebas se determinó que el grado de inclinación de la cámara no exceda los 20^0 con respecto a la horizontal.

4.1.2.7. OPERACIONES MORFOLÓGICAS PARA AUMENTAR LA PRECISIÓN DEL ANPR

Con el objetivo de mejorar el rendimiento de la red neuronal, se ha utilizado operaciones morfológicas en los casos en los que la red no identifique correctamente el caracter. Cuando un caracter no es identificado, se aplica las transformaciones morfológicas de erosión, dilatación y esqueletonización, en ese orden hasta que en alguna de ellas se logre el resultado esperado.

El proceso de dilatación consiste en hacer crecer las regiones limitantes, de manera que el caracter aumente su grosor. Por contrapartida la erosión reduce el groso de la imagen, al igual que la esqueletonización.

4.2. SELECCIÓN DE LA INFRAESTRUCTURA DEL PROTOTIPO

A continuación se detalla los materiales usados para la implementación física del proyecto.

4.2.1. CONFIGURACIÓN DE LA CÁMARA IP

La cámara seleccionada para el proyecto ha sido EasyN series F (Figura IV-N°14), debido a las prestaciones que presenta frente a su costo. Tiene una definición de 680x480 y auto iluminación nocturna con una distancia de hasta 10m.

Para utilizar la cámara web en OpenCV, es necesario instalar las librerías GStreamer, ffmpeg y xine. Para Utilizar las cámaras es necesario conectarlas en red con el computador mediante un router y con el software proporcionado por el fabricante se debe configurar la dirección IP y el puerto. El número total de puertos es de 65535, de cuales podemos escoger cualquier valor. Para acceder a la cámara por medio del explorador basta con escribir en la barra de direcciones la dirección IP seguido del puerto, separados por dos puntos.

FIGURA IV-N°14: CÁMARA EASYN SERIES F

Fuente: http://www.prlog.org/11559502-easyn-series-wireless-ip-camera-internet-pt-wpa-ir-led.jpg

Para obtener la dirección Web completa se debe observar en las propiedades de la imagen que muestra el explorador, la misma nos servirá para acceder a la cámara en OpenCV, para nuestro caso la configuración de las dos cámaras se observa en la figura IV-N°15.

FIGURA IV-N°15: CONFIGURACIÓN DE LAS CÁMARAS IP

```
VideoCapture camara1("http://192.168.1.103:65534/videostream.asf?user=admin&pwd=&resolution=32&rate=0");
VideoCapture camara2("http://192.168.1.101:65533/videostream.asf?user=admin&pwd=&resolution=32&rate=0");
```

Fuente: Los Autores

4.2.2. DISEÑO DEL SISTEMA DE ACCESO PARA PAQUEADEROS TARIFADOS

En el sistema de parqueaderos tarifados hay que tener en cuenta algunas variables como son el número de sitios libres disponibles, el número de placa, los tiempos de ingreso y salida, así como la tarifa por hora.

Para almacenar los datos de una mejor manera se creó un vector de estructuras que contiene las variables. La dimensión del vector puede ser modificada fácilmente en función de las condiciones reales.

Cuando se detecta un automóvil en la entrada, se consulta si existen lugares disponibles, si es el caso se captura la imagen de la cámara de entrada y mediante el sistema ANPR se obtiene los caracteres, se almacena el tiempo de ingreso y a la vez se indica el lugar disponible.

De manera similar, cuando se detecte un automóvil en la salida, se procede a encontrar los datos de placa, se busca una coincidencia con el número de placa en el vector de estructuras y se procede a calcular el valor a pagar. El valor se lo calcula haciendo una diferencia entre el tiempo de salida y el tiempo de entrada, multiplicado por la tarifa.

El valor de la tarifa por hora puede ser modificada dependiendo de las condiciones económicas de la región, y el costo para el usuario será exacto sin redondeos.

El sistema tiene una interfaz gráfica sencilla y de fácil uso, minimizando la interacción del operador en la aplicación. En la misma que se puede visualizar las variables mencionadas anteriormente.

Debido a la presencia de dos cámara, una de entrada y otra de salida, se ha considerado un diseño en el que en la misma ventana, se puedan visualizar las dos simultáneamente. En la Figura IV-N°16, se encuentra el diseño escogido para la interfaz gráfica.

4.2.3. SELECCIÓN DEL MATERIAL DE CONSTRUCCIÓN Y ESCALA

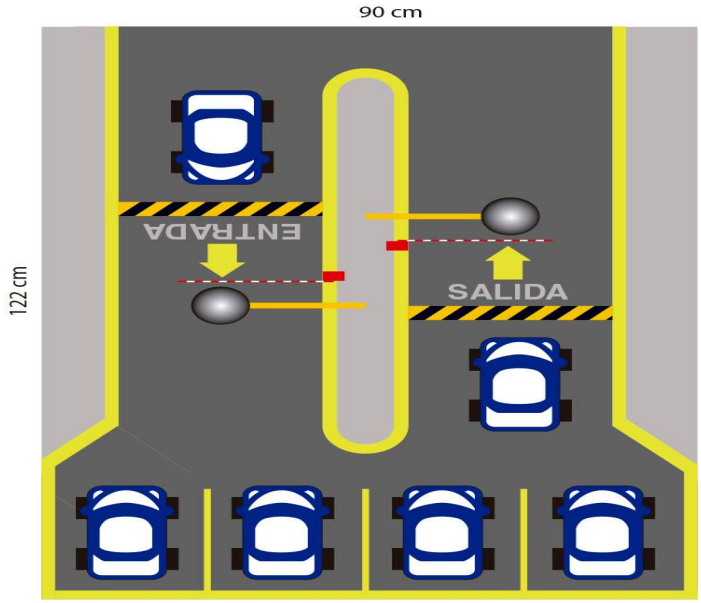
Para realizar las pruebas de funcionamiento del proyecto se ha realizado una maqueta en madera triplex reforzado, en las que se han instalado las cámaras en la parte de ingreso y salida del parqueadero.

La dimensión óptima ha sido de 1.22x0.9 metros (Figura IV-N°17), y los materiales de construcción serán madera, plástico, cartón y varillas metálicas, según sea necesario. Las dimensiones de las placas serán de 20x52 milímetros, con una escala de 7:1.

FIGURA IV-N°16: INTERFAZ GRÁFICA

Fuente: Los Autores

FIGURA IV-N°17: DISEÑO DEL SISTEMA DE PRUEBAS



Fuente: Los Autores

CAPÍTULO V

5. PRUEBAS Y RESULTADOS

En el capítulo se describirá las pruebas a las que fue sometida la red, así como resultados de su entrenamiento. También se evalúa el comportamiento general del sistema.

5.1. RESULTADOS DE ENTRENAMIENTO

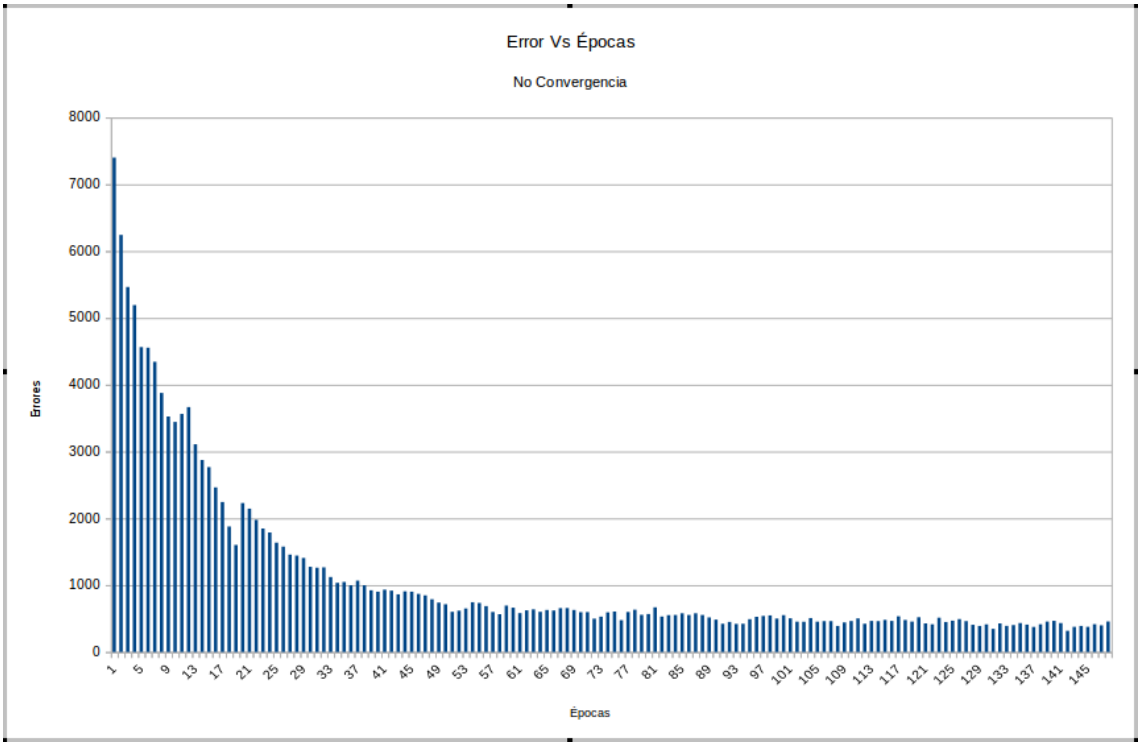
Observar el comportamiento de la red neuronal es de mucha utilidad para realimentar el conocimiento acerca de las RNA.

Como se mencionó anteriormente en el proceso de entrenamiento, con un primer diseño no se obtuvo éxito, debido a la falta de neuronas en la capa de entrada. En el proceso de entrenamiento no se pudo llegar a la convergencia. Podemos observar el comportamiento de la red en la Figura V-N°1, mientras aumentan el número de épocas.

Conforme inicia el proceso de entrenamiento la red tiene más errores o patrones por aprender y le tomará más tiempo ajustar el número de pesos, razón por la cual las primeras épocas le toma más tiempo a la red.

El número de errores disminuye drásticamente en las primeras épocas, bajando de 7400 a 1800 en sólo 17 épocas. Pero en las posteriores épocas la disminución comienza a decaer y llegar a un valor en el que la red no mejora. Depende de la experiencia del desarrollador determinar cuándo una red no llegará a la convergencia y detener el proceso, de no ser así la red ajustará sus pesos en infinitas iteraciones sin mejora alguna.

FIGURA V-N°1: NO CONVERGENCIA DE LA RED



Fuente: Los Autores

Se puede aproximar el comportamiento del error en función de sus épocas para un mejor estudio, y se obtiene la siguiente ecuación:

$$E(epocas) = 7000 * e^{\left(\frac{-epocas}{20}\right)} + 400 \qquad \textbf{(5.1)}$$

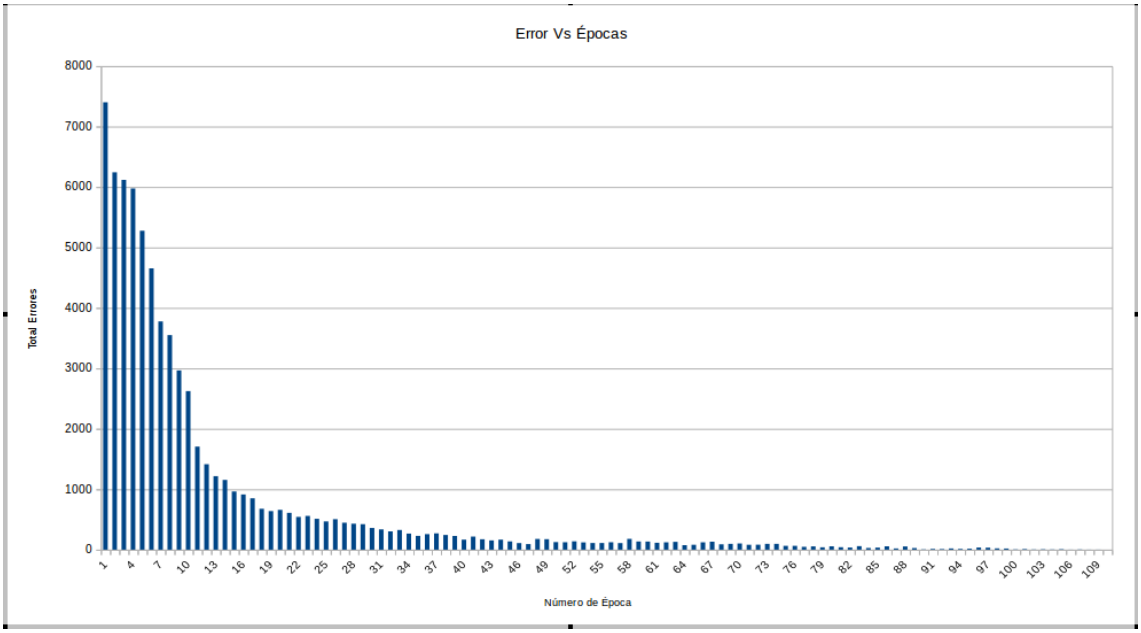
De la ecuación **5.1** se puede observar que conforme sus épocas tienden a infinito existirá un valor asintótico igual a 400, por lo tanto nunca se minimizará el error a 0.

Por el contrario, luego de la búsqueda de la red ideal para nuestra aplicación se obtuvieron los siguientes resultados en el proceso de aprendizaje en la Figura V-N°2.

La red describe un funcionamiento similar al caso anterior en las primeras épocas, reduciendo considerablemente el error por época. Pero tiene el desenlace esperado, minimizando el error a cero.

El proceso de aprendizaje no asegura que el error siempre sea disminuido de una época contigua a otra, pero en un promedio el error tiende a decaer. Esto se puede apreciar de una mejor manera observando una ampliación de la Figura V-N°2, en la Figura V-N°3.

FIGURA V-N°2: CONVERGENCIA DE LA RED



Fuente: Los Autores

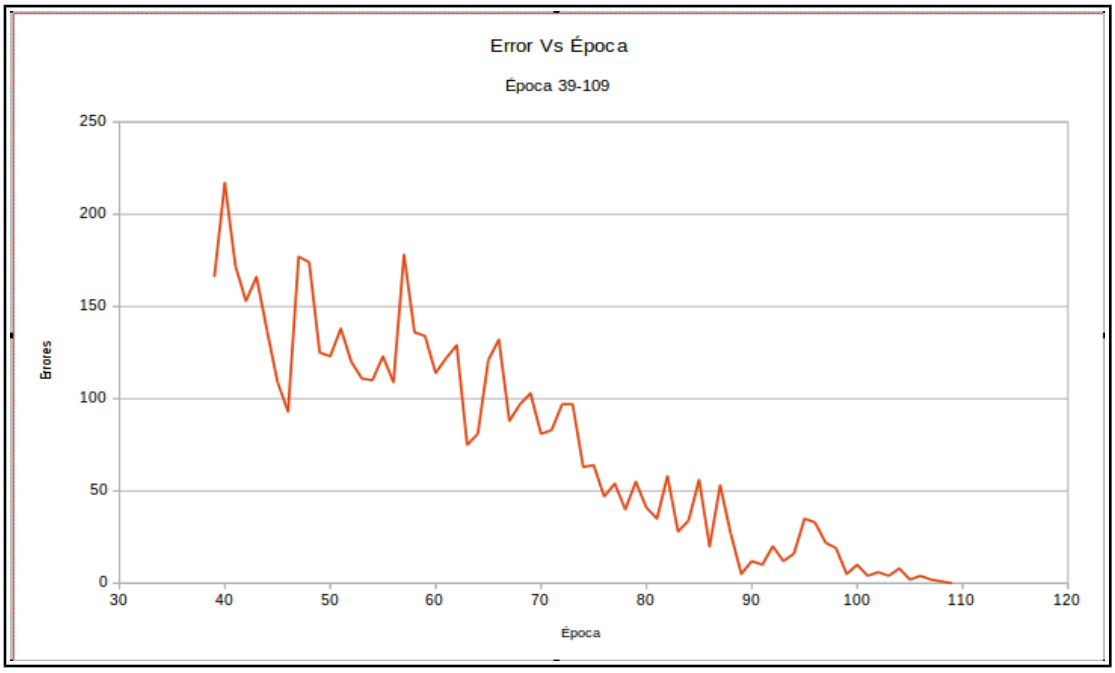
5.2. PRUEBAS DE CLASIFICACIÓN DE CARACTERES

Para evaluar el comportamiento de la red, se consideraron dos opciones: una red entrenada con 100 muestras por caracter, y otra con 200 muestras.

Para la evaluación del desempeño de nuestra red se han utilizado muestras a clasificar de diferentes fuentes a las escogidas para el entrenamiento dando un total de 477.

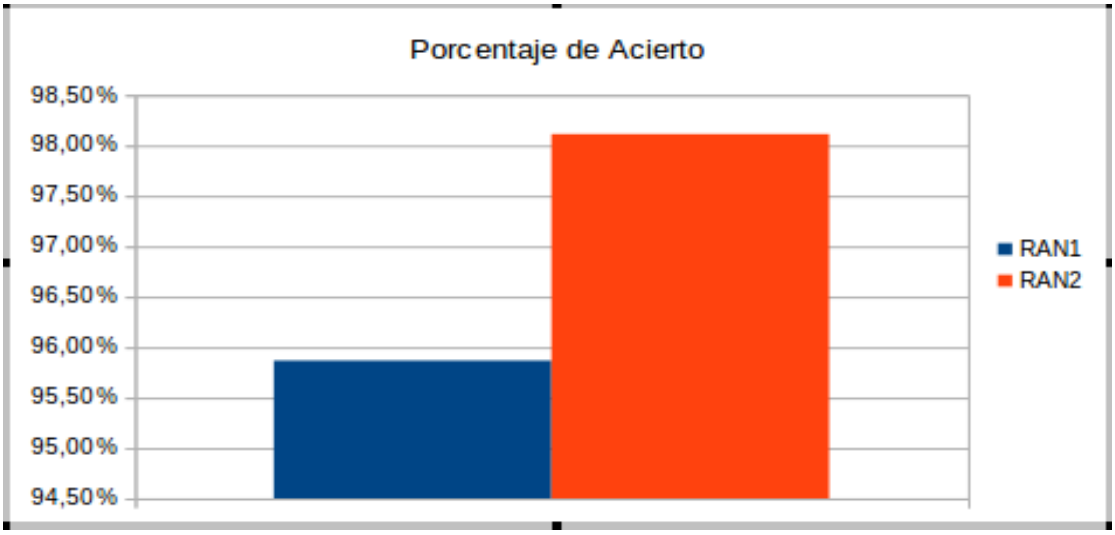
Los resultados obtenido se observan en la Figura V-N°4, la red con 100 muestras (RNA1) tiene un nivel de acierto del 95.867%, mientras que la red con 200 muestras (RNA2) tiene un nivel de eficacia del 98.113%.

FIGURA V-N°3: COMPORTAMIENTO DEL APRENDIZAJE



Fuente: Los Autores

FIGURA V-N°4: RESULTADOS DE ACIERTOS



Fuente: Los Autores

Un ejemplo de caracteres identificados correctamente se puede observar en la Figura V-N°5, y en la Figura V-N°6 se presentan una muestra no identificada satisfactoriamente.

FIGURA V-N°5: MUESTRAS IDENTIFICADAS CORRECTAMENTE

Fuente: Los Autores

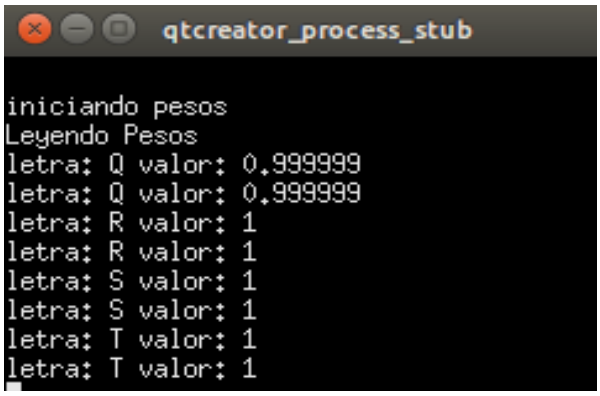
FIGURA V-N°6: MUESTRA NO IDENTIFICADA SATISFACTORIAMENTE

Fuente: Los Autores

Si se observan las salidas que genera la red en la Figura V-N°7, se puede determinar el grado de certeza que presenta al clasificar un carácter. Un valor igual a 1 representa un nivel de certeza del 100%.

De acuerdo a los resultados obtenidos, se puede asegurar que la eficiencia de la red es directamente proporcional a la cantidad de muestras de entrenamiento con una diferencia de 2.246%. Razón por la cuál, se justifica haber apuntado a entrenar a la red con 200 muestras por carácter.

FIGURA V-N°7: RESULTADO DE CLASIFICACIÓN Y SALIDAS



Fuente: Los Autores

5.3. PRUEBAS DE CLASIFICACIÓN EN PLACAS VEHICULARES

Para seleccionar el número de muestras a tomar, se ha seguido el siguiente método estadístico:

$$n = \frac{k^2 Npq}{e^2(N-1) + K^2 pq} \qquad \textbf{(5.2)}$$

Se tiene que:

n corresponde al número de muestras.
k Constante en función del nivel de confianza según distribución normal.
N es el universo.
p y *q* constantes que toman típicamente el valor de 0.5.
e es el error muestral

Se ha escogido un error muestral del 10% que le corresponde a *k* un valor de 1.65, además que el universo o población en el Ecuador es aproximadamente de 290.752 automotores oficialmente registrados, obtenemos un número de muestras igual a:

$$n=\frac{1.65^2*290.752*0.5*0.5}{0.1^2*(290.752-1)+1.65^2*0.5*0.5}=68.05$$

Las muestras fueron tomadas en un rango de distancia entre la placa y la cámara de 1.5m a 2m. Con un ángulo de inclinación de 0^0 con una desviación de más, menos 15 grados.

De los caracteres seccionados correctamente se obtuvo un porcentaje de acierto de 99.256%. En la Figura V-N°8, encontramos ejemplos de placas correctamente identificadas.

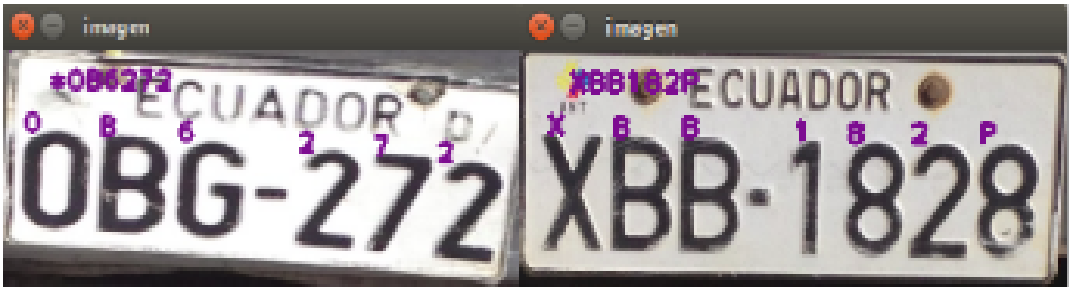
La razón por la cual el porcentaje de éxitos es mayor al obtenido anteriormente es que las placas vehiculares presenta un número muy limitado del tipo de sus fuentes. Además cabe mencionar que las nuevas disposiciones de la ANT de fabricación de placas ha sido pensada para mejorar el rendimiento de los ANPR.

FIGURA V-N°8: PLACAS CORRECTAMENTE IDENTIFICADAS

Fuente: Los Autores

Ejemplos de placas mal identificadas, se muestran a continuación.

FIGURA V-N°9: PLACAS NO IDENTIFICADAS



Fuente: Los Autores

5.4. TIEMPO DE PROCESAMIENTO

Uno de los objetivos más buscados en un programa es la reducción del tiempo de procesamiento. A menor tiempo, se considera que un programa es más eficiente. Las pruebas fueron realizadas con un computador con un procesador Intel I3, se recomienda para disminuir el tiempo de procesamiento un procesador más avanzado. La cantidad de memoria RAM que ocupa el programa es apenas 24Mb.

De las muestras tomadas, el tiempo medio que le lleva identificar los caracteres de una placa es sólo 156.21 milisegundos. Dando un tiempo promedio de 24.39 milisegundos por cada carácter.

En comparación con el trabajo de tesis desarrollado por E. Barragan (2011), el OCR implementado en Labview le tomaba un tiempo de 3.3435 segundos. Con ello se puede justificar la aseveración de que el sistema desarrollado puede competir con los OCR's comerciales. El sistema desarrollado es 21.404 veces más rápido.

La combinación de las dos cámaras IP provoca un retardo en la adquisición de las imágenes de aproximadamente 2 segundos, tiempo que no altera el correcto funcionamiento del sistema.

CONCLUSIONES

1. Se seleccionó a la red de retropropagación para el proyecto. Luego del análisis documental, se ha escogido la red de retropropagación como las más óptima para cumplir con la solución para nuestra problemática. La red de retropropagación tiene la capacidad de trabajar con una gran cantidad de capas y neuronas.

2. Se dimensionó la red de acuerdo a los objetivos planteados. Después de una búsqueda exhaustiva se determinó que el tamaño de la red sea capaz de cumplir con los objetivos planteados. Se encontró una relación directa entre el número de neuronas, especialmente las de la capa de entrada y la cantidad de datos a aprender.

3. Se observó el comportamiento de la red en su fase de aprendizaje. Se pudieron obtener gráficas de disminución del error en los casos en los que la red convergía y en los que la red no convergía.

4. Se investigaron los diferentes tipos de funciones que presenta OpenCV para el procesamiento de imágenes de acuerdo a las necesidades de nuestro proyecto, y en algunos casos se contrastaron en función de su rendimiento.

5. Se obtuvieron resultados favorables clasificando caracteres. En la realización de las pruebas se pudo obtener un grado de eficacia del 98.113% clasificando muestras de varias fuentes, y del 99.256% identificando los caracteres en las placas vehiculares. Además se determinó que la precisión de la red es proporcional a la cantidad de muestras en el entrenamiento.

6. Se creó una solución eficiente y rápida. El tiempo de procesamiento fue relativamente bajo en comparación con otros trabajos realizados mediante el uso de herramientas comerciales. Con un tiempo promedio apenas de 156.21 milisegundos por placa, y 24.39 milisegundos por carácter, se puede afirmar que la herramienta desarrollada hace un uso eficiente de los recursos del computador.

7. Se solucionó el problema presente en los parqueaderos tarifados. Mediante el uso de la tecnología desarrollada se pudo dar una respuesta adecuada a la problemática planteada. Se redujeron los tiempos destinados para el registro de entrada, el registro de salida y facturación.

8. Se creó un modelo a escala. El uso de un modelo a escala facilita mucho la realización del proyecto, además que nos permite obtener una estimación de la funcionalidad del trabajo.

RECOMENDACIONES

1. Se debe seleccionar cuidadosamente el tipo de muestras para el entrenamiento, así como el número de las mismas. Se recomienda comenzar con pocos datos de entrenamiento para poder tener una impresión inicial de las dimensiones de la red que necesitamos.

2. El uso de la función de transferencia que mejor desempeño presentó es la función de tangente hiperbólica. Reduce considerablemente los tiempos que le toma al computador realizar los cálculos de entrenamiento.

3. El uso de archivos de texto para almacenar las variables de entrenamiento así como los pesos tuvieron un buen desempeño. Los tiempos de lectura y escritura fueron reducidos, así como la facilidad de trabajar con ellos.

4. Se debe tener muy en cuenta la selección del mejor algoritmo para segmentar las imágenes, con el algoritmo que detecta objetos mediante rectángulos rotados se obtuvo más errores en la segmentación frente al algoritmo que detecta los objetos en rectángulos con rotación de 0^0.

5. Para minimizar los tiempos de entrenamiento y los tiempos de procesamiento se debe escoger preferentemente un procesador de última generación, así mismo el uso de C++ como lenguaje de programación hace que el uso de memoria RAM sea eficiente.

6. Cuando se deba trabajar con más de una cámara se debe tener en mente como primera opción las cámaras IP, para evitar los problemas de hardware que se pueden presentar en las cámaras Web.

RESUMEN

El trabajo de investigación de desarrollo de un sistema de accesos para parqueadero y cobro tarifario mediante reconocimiento de patrones tiene como objetivo automatizar y mejorar las condiciones de servicio de parqueaderos disminuyendo tiempos de registro y tarifando valores justos en la Facultad de Informática y Electrónica perteneciente a la Escuela Superior Politécnica de Chimborazo.

Mediante análisis documental fueron evaluadas redes neuronales de aprendizaje supervisado y se escogió la red neuronal de retropropagación como más apta para el trabajo.

El sistema se desarrolló bajo el sistema operativo Linux-Ubuntu, usando como compilador QT Creator y para visión artificial las librerías de OpenCV, el lenguaje de programación fue C++. Se codificó la red neuronal y se escogieron muestras para su entrenamiento. Se diseñó el sistema de segmentación y clasificación. Se construyó un modelo a escala de un parqueadero, donde se instalaron dos cámaras IP conectadas en red con el computador.

Los resultados por el sistema de reconocimiento de siete caracteres obtenidos fueron favorables con un porcentaje de 99.256% clasificando caracteres de placas vehiculares. Así mismo el tiempo de procesamiento fue relativamente bajo con un valor medio de 156.21ms.

Se concluye que la red de retropropagación posee un alto desempeño clasificando caracteres y manejando grandes volúmenes de datos. Se ven reducidos los tiempos de ingreso y salida del parqueadero mediante el sistema creado.

Se recomienda a los investigadores de la Facultad de Informática y Electrónica que tomen este trabajo como referencia para trabajar con redes neuronales artificiales.

Palabras clave: <INTELIGENCIA ARTIFICIAL> <OPENCV> <RETROPROPAGACION> <QT CREATOR> <MATRICULAS VEHICULARES> <UBUNTU> <SEGMENTACION> <PARQUEADERO>

ABSTRACT

This research is a development of an accessing system for parking and charging through recognizing prompts, It aims to automatize and better the conditions of service of parking, decreasing register time and charging the correct price in the Faculty of Informatics and Electronics at Superior Polytechnic of Chimborazo.

Using documental analysis neuronal networks of supervising learning were assessed and the most appropriate neural network of Retro-propagation was chosen for work.

The system took into consideration the operative system Linux-Ubuntu by using QT Creator as compiler and for artificial vision the bookstores of OpenCV, programming language was C++. Neural network was coded and samples were chosen for its entertainment. The system of segmentation and classification was designed. A scale model of parking was build including two cameras IP on line with the computer.

The results were positive in 99.256% classifying license plates. Processing time was relatively low with a medium value of 156.21ms.

It is also taken into account that retro-propagation network has a high development classifying characteristics and management of big data volumes. Entering and leaving parking time have been reduced because of the new system.

It is recommended to researches of the Faculty of Informatics and Electronics to consider this work to keep studying neuronal artificial networks.

Key words: <Artificial Intelligence> <OpenCV> <Retro-propagation> <QT Creator> <License plates register> <Ubuntu> <Segmentation> <parking>.

BIBLIOGRAFÍA

1. ÁLVAREZ, M., Análisis, diseño e implentación de un sustema de control de ingreso de vehículos basados en visión artificial y reconocimiento de placas en el parqueadero de la Univesidad Politécnica Salesiana-Sede Cuenca., Tesis Ing. Sistemas., UPS-CUENCA., Cuenca-Ecuador., Escuela de Ingeniería en Sistemas., 2014., pp 63-96.

2. BARRAGÁN, E., Diseño e implementación del sistema de control vehicular utilizando reconocimiento óptico de caracteres en el laboratorio de automatización industrial de la E.I.S., Tesis Ing. Sistemas., ESPOCH., Riobamba-Ecuador., Facultad de Informática y Electrónica., 2011., pp 77-95.

3. BASOGAIN, O., Redes neuronales artificiales y sus aplicaciones., Bilbao-España, Departamento de Sistemas y Automática Escuela Superio de Ingeniería de Bilbao., s.f., pp 1-34.

4. BRONSON, G., C++ para ingeniería y ciencias., 2da ed., México D.F.-México, Cengage learning Editores., 2007., 844 p.

5. GARCÍA, P., Reconocimiento de imágenes utilizando redes neuronales artificiales., Tesis Maestria en Investigación en Informática., Universidad Complutense de Madrid., Madrir-España., Facultad de Informática., 2013., pp 11-20.

6. LELIS, D., Mastering OpenCV with Practical Computer Vision Projects., Birmingham-UK., Packt Publishing., 2012., pp 148-160.

7. MATICH, D., Redes neuronales: Conceptos básicos y aplicaciones., Rosario-Argentina., Universidad Tecnológica Nacional-Facultad Regional Rosario., 2001., 55 p.

8. PAREDES, J., GUERRERO, L., Estudio comparativo entre algoritmos de reconocimiento de borde para identificación de placas de autos., Tesis de Ing. Electrónica, Telecomunicaciones y Redes., ESPOCH., Riobamba-Ecuador., Facultad de Informática y Electrónica., 2012., pp 29-56.

9 PONCE, P., Inteligencia artifial con aplicaciones a la ingeniería., México D. F-México; Alfaomega Grupo Editor., 2010., pp 193-235.

10. RIVERA, J., y BURI, A., Segmentación de imágenes de placas vehiculares usando técnica de crecimiento de regiones., Tesis de Ing. en ciencias computacionales especialización sistemas tecnológicos., ESPOL., Facultad de Ingeniería en Electricidad y Computación., Guayaquil-Ecuador., 2011., pp 47-72.

11. STUART, J. Y NORVING, P., Inteligencia artificial, un enfoque moderno., Traducido del inglés por Juan Corchado., 2da ed., Madrid-España., Pearson Educación., 2004., pp 1-35.

12. VALENCIA, M., YÁNEZ, C., SÁNCHEZ, L., Algoritmo backpropagation para redes neuronales: conceptos y aplicaciones., s.l., Instituto Politécnico Nacional Centro de Investigación Computacional., México D.F.-México., 2006., 14 p.

13. VEGA, J., Desarrollo de un sistema de reconocimiento visual para sistemas Linux embebidos., Proyecto de fin de Carrera., Universidad de Zaragoza., Escuela de Ingeniería y Arquitectura., Zaragoza-España., 2012., 34 p.

14. ANPR
<http://es.wikipedia.org/wiki/Reconocimiento_autom%C3%A1tico_de_matr
%C3%Adculas>.
2015-01-18

15. ARDUINO MEGA 2560.

<http://arduino.cc/en/Main/arduinoBoardMega2560>.

2015-02-02

16. CAMARA IP CON OPENCV

<http://robocv.blogspot.com.au/2012/01/using-your-ip-camera-with-opencv.html>.

2015-04-01

17. CONFIGURACIÓN DE QT Y OPENCV

<http://xrong.org/notes/install-Qt-OpenCV-Ubuntu**>.**

2014-10-14

18. FILTROS EN IMÁGENES

<http://blogrobotica.linaresdigital.com/2011/01/filtrado-de-imagen-suavizado-usando.html>.

2014-12-20

19. OCR

<http://es.wikipedia.org/wiki/Reconocimiento_%C3%B3ptico_de_caracteres>.

2015-01-10

20. OPENCV

<http://docs.opencv.org/doc/tutorials/tutorials.html>.

2014-08-10

21. OPERACIONES MORFOLÓGICAS

<http://es.slideshare.net/balfier/operaciones-morfologicas>.

2015-03-04

22. PROTOCOLO DE COMUNICACIÓN RS232 ENTRE ARDUINO Y QT

<http://forum.arduino.cc/index.php?topic=157883.0>.

2015-04-10

23. REDES NEURONALES ARTIFICIALES

<http://es.wikipedia.org/wiki/Red_neuronal_artificial>.

2014-08-8

<http://es.wikipedia.org/wiki/Propagaci%C3%B3n_hacia_atr%C3%A1s>.

2014-08-15

24. REGLAMENTO DE LICENCIAS VEHICULARES

<http://www.ant.gob.ec/index.php/licencias>.

2014-11-09

25. SEGMENTACION DE LA IMAGEN

<http://docs.opencv.org/doc/tutorials/imgproc/shapedescriptors/bounding_rects_circles/bounding_rects_circles.html>.

2015-02-20

<http://docs.opencv.org/doc/tutorials/imgproc/shapedescriptors/bounding_rotated_ellipses/bounding_rotated_ellipses.html>.

2015-02-21

26. TRATAMIENTO DE LOS DATOS PARA ENTRENAMIENTO

<http://www.nithinrajs.in/ocr-artificial-neural-network-opencv-part-3final-preprocessing/>.

2015-01-12