Artículo **Revista de Cómputo Aplicado**
Junio 2018 Vol.2 No.6 11-19

Algoritmo Perceptrón de Reconocimiento de Imágenes para Detección de Problemas en Cultivos de la Uva

Perceptron Algorithm of Recognition of Images for Detection of Problems in Grape Cultivation

RUIZ-AVILA, Luis Fernando†*, SIERRA-LEYVA, José Luis, FUENTES-VÁZQUEZ, Jhonnathan Alexis y MENDIVIL-REYES, Cinthia Valeria

Instituto Tecnológico de Nogales / Tecnológico Nacional de México

ID 1er Autor: *Luis Fernando, Ruiz-Avila* / **ORC ID**: 0000-0002-9085-3693, **Researcher ID Thomson**: I-3595-2018, **arXiv Author ID**: FernandoR, **CVU CONACYT ID**: 904555

ID 1er Coautor: *José Luis, Sierra-Leyva* / **ORC ID**: 0000-0003-0782-3547, **arXiv Author ID**: Jose_Sierra, **CVU CONACYT ID**: 904333

ID 2do Coautor: *Jhonnathan Alexis, Fuentes-Vázquez* / **ORC ID**: 0000-0002-9273-4014, **Researcher ID Thomson**: I-1196-2018, **arXiv Author ID**: jhonyalexis11, **CVU CONACYT ID**: 903928

ID 3er Coautor: *Cinthia Valeria, Mendivil-Reyes* / **ORC ID**: 0000-0001-9796-1665, **Researcher ID Thomson**: I-4336-2018, **CVU CONACYT ID**: 904697

Recibido Abril 20, 2018; Aceptado Junio 30, 2018

Resumen

En la actividad agrícola se presentan daños causados por enfermedades, falta o exceso de riego o humedad, mal control de plagas, entre otros factores. Diversos estudios han permitido registrar y clasificar información que permita encontrar soluciones a esta problemática. Se propone la implementación de un algoritmo perceptrón de reconocimiento de imágenes para detección de problemas en cultivos de la uva, el cual es un sistema de procesamiento de imágenes del cultivo, que utiliza una cámara de alta resolución en un vehículo aéreo no tripulado (dron), que toma imágenes cuyos tonos RGB de: colores particulares, arrugas en las hojas, u otros datos, de esta forma es posible establecer parámetros de entrenamiento e implementar un algoritmo IA perceptrón de reconocimiento de imágenes para comparar con datos ya conocidos y tomar la acción más conveniente. Se han hecho pruebas con un prototipo a escala y una cámara Arduino a un cultivo de uva, que arrojó con éxito que contenía necrosis en la epidermis, una plaga que produce desecación en sarmientos, reduciendo su desarrollo.

Perceptrón, Cultivos, Dron

Abstract

At agriculture you can find damages caused by diseases, lack of water or excess of irrigation, poor pest control, among other factors. Several studies registered and classify information that allows finding solutions to such problem. We propose the implementation of an Image Recognition Perceptron Algorithm to detect problems in grape crops. This is a crop image processing system, that uses a high resolution camera in an unmanned aerial vehicle (drone), which takes images in RGB, observing tones of particular colours, wrinkles in the sheets, among other data, that helps us to establish training parameters and implement an Artificial Intelligence algorithm for recognizing images to compare them with known data and take the most convenient action. We have performed tests with a scaled Arduino prototype equipped with a camera in a grapes crop, the test showed that the sheets contained necrosis in the epidermis, a plague that produces desiccation in shoots, reducing growing.

Perceptron, Crops, Drone

Citación: RUIZ-AVILA, Luis Fernando, SIERRA-LEYVA, José Luis, FUENTES-VÁZQUEZ, Jhonnathan Alexis y MENDIVIL-REYES, Cinthia Valeria. Algoritmo Perceptrón de Reconocimiento de Imágenes para Detección de Problemas en Cultivos de la Uva. Revista de Cómputo Aplicado. 2018, 2-6: 11-19.

* Correspondencia al Autor (Correo Electrónico: luis.fernando.ruiz_avila.1995._@hotmail.com)
† Investigador contribuyendo como primer autor.

Introducción

El cerebro procesa información y partir de ella, es capaz de dar una respuesta, es un modelo perfecto de aplicar en las técnicas de software y todo sucede en las neuronas (Szeliski, 2010).

La neurona por el proceso de sinapsis es capaz de emitir una respuesta por axones, en analogía, una función matemática con ciertas entradas es capaz de producir diversas respuestas. (Ramesh, Rangachar, & Brian G., 1995).

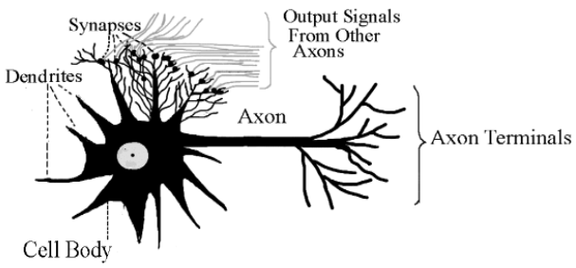

Figura 1 Neurona Biológica
Recuperado de: Intelligent Control Systems... Pág. 43

La visión artificial por computadora +en base a redes neuronales, modela mediante técnicas matemáticas la forma tradicional y apariencia de los objetos para recuperar la forma física y de color de los objetos (González G. F., 2009).

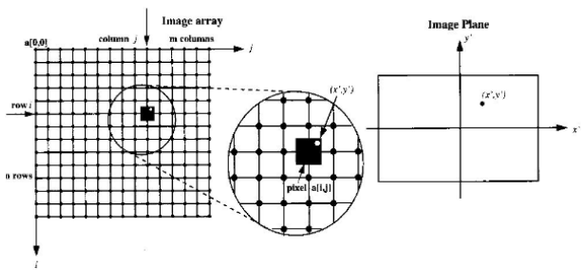

Figura 2 Visión de máquina
Recuperado de Relationship between image plane and image array indices.

Usando visión artificial se adquieren las imágenes, se hace el procesamiento previo, segmentación, representación y descripción de los datos obtenidos, que servirán a su vez como las entradas de datos para ser reconocidas e interpretadas por nuestro modelo de réplica del cerebro que es un perceptrón multicapa que será ajustado para proveernos de información útil. (Giraldo, 2011)

En la actividad agrícola se presentan daños causados por enfermedades, falta o exceso de humedad, mal control de plagas, entre otros factores. Diversos estudios han permitido registrar y clasificar información que permita encontrar soluciones a esta problemática, como el detectar plagas al tener el conocimiento de la plaga es posible redimensionarla en datos que en nuestro caso son patrones de color y textura que pueden ser obtenidos de una imagen.

Se propone un perceptrón de reconocimiento de imágenes para detección de problemas en cultivos, que utiliza una cámara de alta resolución en un vehículo aéreo no tripulado (dron), que toma imágenes cuyos tonos RGB de: colores particulares, arrugas en las hojas, u otros datos, con las que es posible establecer parámetros de entrenamiento e implementar un algoritmo IA perceptrón de reconocimiento de imágenes para comparar con datos ya conocidos y tomar la acción más conveniente.

Se han hecho pruebas con un prototipo a escala y una cámara Arduino a un cultivo de uva, que arrojó con éxito que contenía necrosis en la epidermis, plaga que produce desecación en sarmientos, reduciendo su desarrollo.

El Perceptrón Multicapa

Un perceptrón multicapa tiene tres diferentes zonas en las cuales procesa la información de diferentes maneras, en las capas de entrada, se reciben las variables de entrada, capas ocultas donde ocurre la mayor parte del procesamiento y la capa de salida que es la obtención de resultados. (García, 2006)

Se ha modelado una Neurona N que permite recoger las dendritas, entradas de datos, el axón la salida de datos y el cuerpo, se produce el proceso de sinapsis, una función de activación que produce una salida. Cada entrada se asocia a un peso correspondiente, de esta forma se representa en (1) donde n es la función y de salida y w son los pesos por cada entrada. (Aurelio, Humberto, & Serguei, 2002)

$$n = x_1 w_1 + x_2 w_2 + x_3 w_3 + \cdots + x_i w_i \qquad (1)$$

ISSN-2531-2952

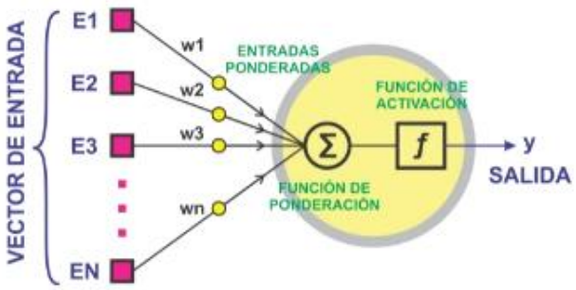

Figura 3 Modelo De Neurona Artificial
Recuperado de: Redes Neuronales Artificiales. Morse Blas, et al. Santa Fe 2015

Así, se establece una relación con otras neuronas donde las entradas se representan por patrones de un determinado color en la imagen y se procesa para obtener datos específicos, con los cuales se forman conexiones entre las entradas de una neurona y las salidas de otras para tener un procesamiento más óptimo, en base a conocimientos previos de las afecciones que pueden presentarse en un cultivo de uva.

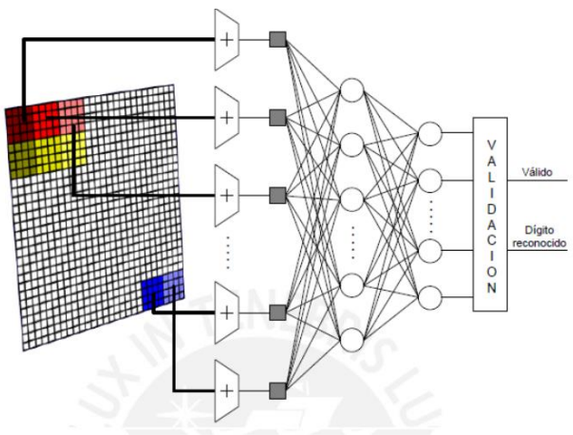

Figura 4 Representación De Colores
Recuperado de: (Hernández, 2014)

Como se aprecia en (2) y (3), Y es el resultado y f es una función de h y h es la actividad de la neurona, la sumatoria de n el número de entradas que posee el modelo, w indica el peso de la neurona y x es la entrada de valor de umbral asignado. (González P. J., 2015), el peso sináptico en ese punto, es la fuerza en que se unen dos neuronas, las entradas son señales del entorno, el umbral es un excedente de voltaje en la neurona, para que ocurra la activación y que se produzca una señal de salida.

$$y = f(h(\{Actividad\ Neuronal\})) \quad (2)$$

$$h = \sum_{xi \in X}^{n} wi\ x\ i + 0 \quad (3)$$

Procesamiento De La Imagen

La capa de entrada solo recibirá los datos obtenidos en el procesamiento de la imagen con sus respectivos patrones para el aprendizaje e incluye los siguientes puntos.

Preprocesado De La Imagen

Tratamiento sistemático de los pixeles de la imagen con filtros y transformaciones geométricas para realzar datos específicos, incluyen la mejora del contraste, eliminar el brillo, iluminación y restauración de la imagen (La Serna Palomino & Román Concha, 2009)

Pixeles: Vecindad

Estas son las relaciones que forman los pixeles en una matriz de pixeles contiguos dependiendo de los datos que estén aportando, en la vecindad para todo punto P de coordenadas (x, y) se dice que un pixel q pertenece a sus cuatro vecinos y se describe como en (4) si y solo si q tiene coordenadas como en (5).

$$q = q \in N_4(p) \quad (4)$$

$$(x - 1y)\ o\ (xy - 1)\ o\ (x + 1y)\ o\ (xy + 1) \quad (5)$$

Para todo punto p de coordenadas (x, y) se dice que un pixel q pertenece a sus ocho vecinos si se describe en (6) si y solo si q tiene coordenadas como en (7). (Vélez Serrano, Moreno Díaz, Sánchez Calle, & Sánchez-Marín, 2003)

$$q = q \in N_8(p) \quad (6)$$

$$(x - 1y)\ o\ (xy - 1)\ o\ (x + 1y)\ o\ (xy + 1)\ o\ (x - 1\ y - 1)\ o\ (x + 1\ y - 1) o\ (x + 1\ y + 1) \quad (7)$$

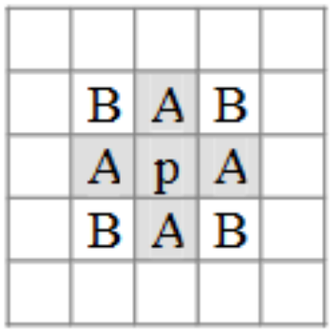

Figura 5 Vecindario del punto p
Recuperado de: Visión Artificial. José F. Vélez S., et al. Madrid. 2003

Relaciones De Distancia

La distancia euclidiana entre pixeles (8), distancia entre el punto P de coordenadas (x, y) y el punto q de coordenadas (s, t) (Vélez Serrano, Moreno Díaz, Sánchez Calle, & Sánchez-Marín, 2003)

$$d(p,q) = \sqrt{(x-s)^2 + (y-t)^2} \qquad (8)$$

Detección de Patrones De Color

Para la detección de zonas de color y otros patrones de toma de datos existe el modelo de color RGB (Red, Green, Blue), de los cuales se toma cada uno de los pixeles, para detectar patrones de color en grupo (Alegre, Pajares, & de la Escalera, 2016)

Detección De Patrones: Transformada De Fourier De Imágenes Digitales

Herramienta que describe fenómenos que ocurren a menudo y aproximar fenómenos de funciones no lineales, en este caso, patrones de color a escala de grises distintivos en la imagen. En niveles de gris I(x, y), así los niveles para una función bidimensional, los factores Ic(n, m), que teniendo en cuenta las bases trigonométricas de las series de Fourier donde los coeficientes $\{a_n\}$ y $\{b_n\}$, son una serie funcional de la forma (9) de la función definida para f(x) con intervalos [-π, π] a la serie trigonométrica que tiene de coeficientes (10) y (11):

$$\frac{a_0}{2} + \sum_{n-1}^{\bowtie} a_n \cos(nx) + \sum_{n-1}^{\bowtie} a_n \operatorname{sen}(mx) \qquad (9)$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,dx \quad \forall_n = 0,1,2\ldots \qquad (10)$$

$$b_m = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(mx)\,dx \quad \forall_m = 0,1,2\ldots \qquad (10)$$

La suma del conjunto de funciones sinusoidales de diferencias frecuencias promedia por unos coeficientes con la finalidad de converger a f(x) el conjunto de señales sinusoidales es una base para el dominio de la frecuencia, los coeficientes de que corresponden la serie Fourier de una función son la transformada de Fourier de la función.

En un conjunto de N valores que en nuestro caso son patrones de color distintivos con relaciones de distancia y vecindad entre los colores de cada pixel de la imagen que formarían una señal discreta o muestreada de la imagen, implican tomar en cuenta la resolución y calidad de la imagen. Estos estarán dados por 11:

$$F = \{f(0), (f1), \ldots . f(N-1)\} \qquad (11)$$

Cambiando las integrales por sumatorias se consigue pasar de una definición continua a discreta y es necesario hacer los cambios de variables sobre los coeficientes (a_n,$b_{m)}$, para que la función en un intervalo general se defina como [0, N-1] en vez del intervalo [-π, π], y utilizando la fórmula de Euler se puede compactar la notación transformando el coeficiente (a_n,$b_{m)}$, en un numero complejo, así para una señal temporal f(k) el coeficiente enésimo ab de la serie de Fourier discreta es (12):

$$ab(n) = \frac{1}{N}\sum_{K=0}^{N-1} f(k) e^{-j\frac{2\pi}{N}kn} \quad \forall_n = 1.,N-1 \qquad (12)$$

Formando así una señal discreta en el dominio de la frecuencia de (11) a (13):

$$F = \{ab(0), ab(1), \ldots . ab(N-1)\} \qquad (13)$$

Para el tratamiento de imágenes se amplía la fórmula para imágenes cuadradas en escalas de grises, así los coeficientes I_c (n, m) se calculan según la ecuación (14) normalizando de 0 a 255. (Vélez Serrano, Moreno Díaz, Sánchez Calle, & Sánchez-Marín, 2003)

$$I_c(n,m) = F\big(I(x,y)\big) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I(x,y)\, e^{-j\frac{2\pi xn}{N}} e^{-j\frac{2\pi my}{N}} \quad \forall_{n,m} = 0,1.,N-1 \qquad (14)$$

Preprocesado de la imagen

De la información tomada de la imagen, se determina si a una planta le hace falta agua o se excede, o presenta alguna plaga; se requiere un preprocesado que arroje una matriz de datos para establecer patrones, que conjuntan la base de aprendizaje para el perceptrón multicapa, mediante la generación hacia atrás del error que se pueda producir (Giraldo, 2011). Al final de estos pasos se obtiene una matriz de datos, arrojando patrones distintivos de la imagen.

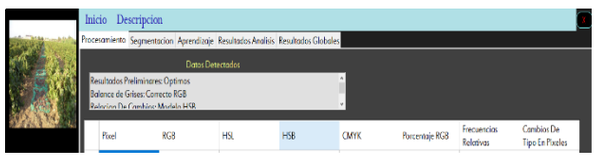

Figura 6 Captura. Etapa De Preprocesamiento
Fuente: Elaboración Propia

ISSN-2531-2952

Segmentación

En la segmentación se obtienen bordes, líneas, curvas, que indican patrones de datos, que realzan la falta de humedad en las hojas, comparando con los datos de color obtenidos en el procesamiento inicial. (La Serna Palomino & Román Concha, 2009). Se dividen en zonas y se etiquetan los datos, pixel por pixel. La segmentación se adecua a medida.

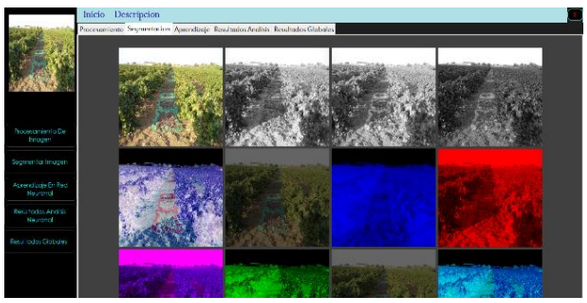

Figura 7 Captura. Etapa De Segmentación

Método De Entrada De Datos

Una vez obtenidos los datos se crea un patrón, clasificando los patrones de acuerdo a si tienen la afección distintiva, al ser procesada, la neurona definirá el patrón de error y la correlación con la función de activación definida por la constante variante de umbral en cada neurona generalizando y disminuyendo el error hasta un mínimo aceptable.

Función De Activación

Es la función que activa o inhibe las señales recibidas y muestra resultados, utiliza la función de tangente hiperbólica (15) se adecua a la forma de entrenamiento (Ali & Mohammad, 2001).

$$f(x) = \frac{1}{1+e^{-ax}} \qquad 0 \leq f(x) \leq 1 \qquad (15)$$

Aprendizaje, Backpropagation y Descenso Del Gradiente

Este algoritmo se basa en la función de error, arroja datos en una curva, que deben alcanzar un mínimo y un máximo de error, el propósito, mediante el descenso del gradiente, es evitar mínimos locales de la curva. (Anil K., Jianchang, & K. M., 1996).

1ro, se define una amplitud del error para determinar el grado de aceptabilidad que se desea tener, valores desde 0.1 a 0.9. 2do, se elige un patrón de entrada, la matriz de datos, para que cumpla con los requerimientos, a esto se le conoce como aprendizaje asistido, donde se determinan una serie de valores que se desea alcanzar. 3ro, las capas ocultas del perceptrón comienzan a propagar las señales adentro de la red. 4to, se procede a calcular δ_i^L para la capa de salida que es como se define la fórmula de activación en (16).

$$\boldsymbol{\delta_i^L = f'(net_i^L)[d_i^L - o_i^L]} \qquad (16)$$

Donde $\mathbf{net_i^L}$ y $\mathbf{o_i^L}$ representan la entrada y salida (respectivamente) de la i-ésima unidad en la lésima capa y $\mathbf{f'}$ es la derivada de la función de activación $\mathbf{f(x)}$ en este caso (17).

$$\boldsymbol{f(x) = f(x)[1 - f(x)]} \qquad (17)$$

5to, ahora para el Perceptrón multicapa en la función de activación se tendría (18)

$$\boldsymbol{\delta_j^L = [z_j^{(q)}(1-z_j^{(q)})]\,(\,t_j^{(q)} - z_j^{(q)})} \qquad (18)$$

6to, se procede Calcular las deltas (δ) para las capas previas por propagación del error hacia atrás la ecuación (19).

$$\boldsymbol{\delta_i^L = f'(net_i^l)\sum_k v_{ik}^{l+1}\,\delta_k^{l+1}} \qquad (19)$$

Para l = (L − 1)..., 1, donde los $\mathbf{u_{ik}}$ representan los parámetros libres de la red o pesos. Una vez hecho esto, se actualizan los pesos (20) y (21):

$$\boldsymbol{\nabla v_{ki}^l = \eta \delta_i^l o_k^{l-1}} \qquad (20)$$

$$\boldsymbol{u_{mj}^{(r+1)} = u_{mj}^{(r)} + \eta \delta_j^L y_n^{(q)}} \qquad (21)$$

y para las capas ocultas ecuación (22)

$$\boldsymbol{w_{nm}^{(r+1)} = w_{nm}^{(r)} + \eta \delta_m^l x_n^{(q)}} \qquad (22)$$

Posteriormente se itera hasta el segundo paso y se repite para el siguiente patrón hasta alcanzar el mínimo de error (fijado a priori) o hasta alcanzar el número máximo de iteraciones.

ISSN-2531-2952

Metodología A Desarrollar

En base a un perceptrón multicapa, y la descripción biológica de una neurona, se puede aplicar la misma relación a un modelo matemático de conexiones. Definiendo primeramente cuáles serán las entradas de los datos, que en este caso son las técnicas visión artificial. El procesamiento en esta etapa de los datos y la obtención de los resultados iniciales para el procesamiento en la etapa de reconocimiento ocupa la mayor parte del proceso este proceso es descrito a continuación en la siguiente imagen.

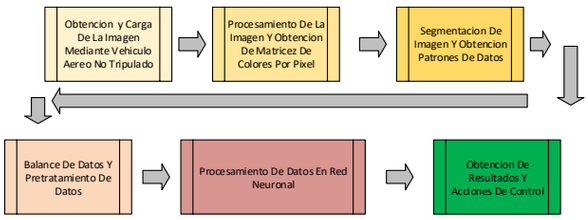

Tabla 1 Diagrama de bloques
Fuente: Elaboración Propia

La primera fase del proceso consiste en la obtención de las capturas del cultivo, por medio del vehículo aéreo no tripulado, el cual es imprescindible para la toma de fotogramas que abarquen un área más grande del terreno y en consecuencias más datos son registrados en la red neuronal permitiendo una mayor cantidad de datos, que son más confiables y se pueden tomar como muestras representativas del cultivo. Estas fotografías son guardas en una SD-CARD (Secure Digital Card), para su posterior análisis. La segunda fase se basa en la aplicación de técnicas de procesamiento de imágenes digitales para la obtención por filtros de color, vecindad, relaciones de distancia y posición, pixel por pixel, para obtener matrices de datos, las cuales serán clasificado para determinar las entradas de datos de la red neuronal.

```
for (int i = 0; i < mapa.Height; i++)
{
    for (int j = 0; j < mapa.Width; j++)
    {
        colores = mapa.GetPixel(j, i);
        f.Add(new int[] {colores.R,colores.G,colores.B});
        f2.Add(new int[] { Convert.ToInt32(mapa.GetPixel(j, i).GetSaturation()),
            Convert.ToInt32(mapa.GetPixel(j, i).GetBrightness()),
            Convert.ToInt32(mapa.GetPixel(j, i).GetHue()) });
        Color nuevo = colores;
        nuevo= Color.FromArgb(205,nuevo);
        nuevo = Color.FromArgb(100,nuevo.B,nuevo.B,nuevo.B);
        mapa.SetPixel(j, i, nuevo);
        mapa.SetResolution(10f,10f);
        mapa.MakeTransparent();

    }
}
```

Figura 8 Obtención de color natural de la imagen, para la aplicación de técnicas de procesamiento de donde formaremos matrices de datos
Fuente: Elaboración Propia

La tercera fase consiste en aplicar segmentación de imágenes a partir transformada de Fourier de imágenes digitales con los datos obtenidos y registrados en las matrices de datos formando así un análisis más completo de las capturas tomadas en escalas de grises, y comparando los resultados con las frecuencias de muestras ya almacenadas formando así un conjunto de N valores de las muestras tomadas, que en nuestro caso son patrones de color distintivos con relaciones de distancia y vecindad entre los colores de cada pixel de la imagen que formarían una señal discreta o muestreada de la imagen, para determinar variaciones en picos altos y bajos de frecuencia y así determinar las zonas más conflictivas para el análisis.

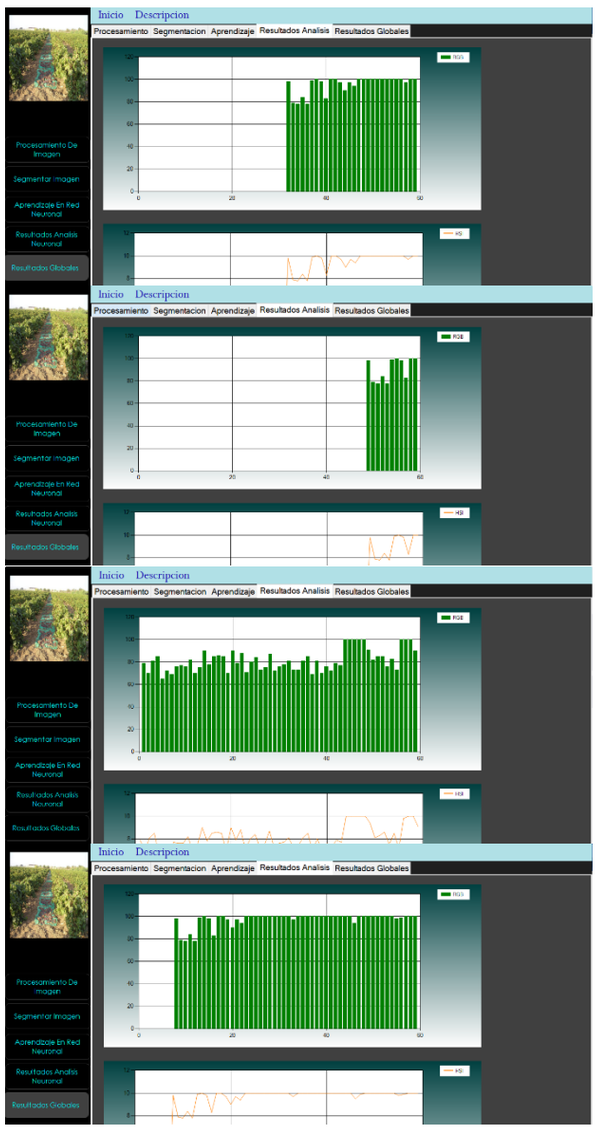

Figura 8 Resultados de datos obtenidos por patrón de color
Fuente: Elaboración Propia

ISSN-2531-2952

La cuarta fase consiste en sacar un balance general de los datos obtenidos, generalizando datos a partir de los tomados para meterlos al análisis neuronal como otro factor de análisis, como ejemplo de esto el porcentaje de color en RGB por pixel, frecuencias relativas en color por pixel y el cambio de color de un pixel vecindad a otro.

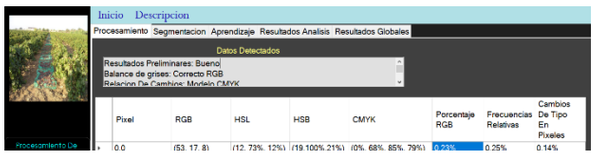

Figura 8 Imagen representativa de resultados obtenidos en la cuarta fase
Fuente: Elaboración Propia

La quinta fase consiste en aplicar a los datos ya recolectados el algoritmo perceptron, consiste en, primer paso otorgar los pesos sinápticos a las neuronas según el conocimiento y datos ya obtenidos anteriormente, se procede a analizar estos datos contenidos en archivos .txt en los cuales se han guardado valores y determinaran el facto de activación de la función de activación para cada neurona.

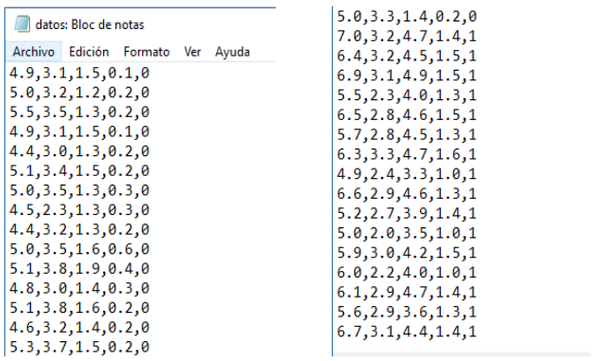

Figura 9 Archivo datos.txt de una de las pruebas obtenidas
Fuente: Elaboración Propia

```
PerceptronVideo.Program                                   LeerPatrones()
static void LeerPatrones()
{
    string data = System.IO.File.ReadAllText(dataPath).Replace("\r", "");

    string[] row = data.Split(Environment.NewLine.ToCharArray());
    for (int i = 0; i < row.Length; i++)
    {
        string[] rowData = row[i].Split(',');

        double[] inputs = new double[inputCount];
        double[] outputs = new double[outputCount];

        for (int j = 0; j < rowData.Length; j++)
        {
            if (j < inputCount)
            {
                inputs[j] = Normalizacion(double.Parse(rowData[j]), inputMin, inputMax);
            }
            else
            {
                outputs[j - inputCount] = Normalizacion(double.Parse(rowData[j]), outputMin, outputMax);
            }
        }

        input.Add(inputs);
        output.Add(outputs);
    }
}
```

Figura 10 Obtención de valores de los archivos
Fuente: Elaboración Propia

Estos datos son normalizados para obtener valores aceptables dentro de la muestra tomada, ya que cada archivo es una muestra.

```
4 referencias
static double Normalizacion(double value, double min, double max)
{
    return (value - min) / (max - min);
}
```

Figura 11 Normalización de valores
Fuente: Elaboración Propia.

Una vez hecho esto se procede a iterar con los campos y con el algoritmo de backpropagation para determinar un nuevo aprendizaje a partir de detectar máximos y mínimos en la evaluación de los datos generalizados en las distintas neuronas. Así disminuyendo el grado de error para las capas de neuronas internas y a su vez mejorando el conocimiento.

```
public bool AprendizajeBackPropagation(List<double[]> input, List<double[]> desiredOutput, double alpha, double maxError, int maxIterations)
{
    double err = 99999;
    log = new List<string>();
    while (err > maxError)
    {
        maxIterations--;
        if (maxIterations <= 0)
        {

            return false;
        }

        if (Console.KeyAvailable)
        {
            System.IO.StreamWriter w = new StreamWriter(@"C:\Users\user_\Desktop\bin\Debug\log.txt");


            string pasar = "";
            foreach (var item in log)
            {
                pasar = item + "\n";
                w.WriteLine(pasar);
            }
            return true;
        }

        AplicarBackPropagation(input, desiredOutput, alpha);
        err = ErrorGenerico(input, desiredOutput);
        log.Add(err.ToString());
        Console.WriteLine(err);
    }
    System.IO.File.WriteAllLines(@"C:\Users\user_\Desktop\bin\Debug\logtail.txt", log.ToArray());
```

Figura 12 Método genérico para actualizar valores en las funciones de cada neurona de la red neuronal
Fuente: Elaboración Propia

```
1 referencia
void AplicarBackPropagation(List<double[]> input, List<double[]> desiredOutput, double alpha)
{
    GuardarDeltas();
    for (int i = 0; i < input.Count; i++)
    {
        Activacion(input[i]);
        GuardarSigmas(desiredOutput[i]);
        ActualizarBias(alpha);
        AgregarDeltas();
    }
    ActualizarPesos(alpha);

}
```

Figura 13 Aplicación del algoritmo
Fuente: Elaboración Propia

```
1 referencia
double ErrorIndividual(double[] realOutput, double[] desiredOutput)
{
    double err = 0;
    for (int i = 0; i < realOutput.Length; i++)
    {
        err += Math.Pow(realOutput[i] - desiredOutput[i], 2);
    }
    return err;
}
1 referencia
double ErrorGenerico(List<double[]> input, List<double[]> desiredOutput)
{
    double err = 0;
    for (int i = 0; i < input.Count; i++)
    {
        err += ErrorIndividual(Activacion(input[i]), desiredOutput[i]);
    }
    return err;
}
```

Figura 14 Calculo del error
Fuente: Elaboración Propia

ISSN-2531-2952

En cada iteración además de delimitar con el grado de error y las funciones de activación, también se actualizan la estructura central de la red, es decir los pesos sinápticos las neuronas, así como los valores de la sumatoria de cada capa, para delimitar más aun el error.

```
void ActualizarPesos(double alpha)
{
    for (int i = 0; i < Capas.Count; i++)
    {
        for (int j = 0; j < Capas[i].numberOfNeurons; j++)
        {
            for (int k = 0; k < Capas[i].neurons[j].weights.Length; k++)
            {
                Capas[i].neurons[j].weights[k] -= alpha * deltas[i][j, k];
            }
        }
    }
}
```

Figura 16 Actualización de pesos
Fuente: Elaboración Propia

```
1 referencia
void ActualizarBias(double alpha)
{
    for (int i = 0; i < Capas.Count; i++)
    {
        for (int j = 0; j < Capas[i].numberOfNeurons; j++)
        {
            Capas[i].neurons[j].bias -= alpha * sigmas[i][j];
        }
    }
}
```

Figura 17 Actualización de vías
Fuente: Elaboración Propia.

```
void GuardarSigmas(double[] desiredOutput)
{
    sigmas = new List<double[]>();
    for (int i = 0; i < Capas.Count; i++)
    {
        sigmas.Add(new double[Capas[i].numberOfNeurons]);
    }
    for (int i = Capas.Count - 1; i >= 0; i--)
    {
        for (int j = 0; j < Capas[i].numberOfNeurons; j++)
        {
            if (i == Capas.Count - 1)
            {
                double y = Capas[i].neurons[j].lastActivation;
                sigmas[i][j] = (Neurona.FunctionSigmoide(y) - desiredOutput[j]) * Neurona.SigmoidDerivated(y);
            }
            else
            {
                double sum = 0;
                for (int k = 0; k < Capas[i + 1].numberOfNeurons; k++)
                {
                    sum += Capas[i + 1].neurons[k].weights[j] * sigmas[i + 1][k];
                }
                sigmas[i][j] = Neurona.SigmoidDerivated(Capas[i].neurons[j].lastActivation) * sum;
            }
        }
    }
}
```

Figura 18 Guardar sigmas de la función por capa.
Fuente: Elaboración Propia

Todos estos datos se guardan para disminuir el grado de error que tiene el algoritmo.

```
1,48131520186361
1,48015323947896
1,47899016906586
1,47782599485472
1,47666072113586
1,47549435225941
1,47432689263537
1,47315834673347
1,47198871908317
1,47081801427358
1,46964623695346
1,46847339183109
1,46729948367424
1,46612451731009
1,46494849762518
1,46377142956531
```

Figura 19 Disminución del grado de error al inicio de entrenamiento, iteración 2513
Fuente: Elaboración Propia

Resultados

Los resultados obtenidos en la simulación de aprendizaje del perceptrón y en la utilización del programa, fueron satisfactorios, al probarlo con imágenes a un cultivo de uva, arrojó con éxito la diferencia entre fotografías de hojas que padecían necrosis en la epidermis y las que no lo tenían, con una eficiencia del 95 %.

También se pudo predecir a cuáles de las plantas les faltaba humedad o les sobraba y establecer un parámetro de por qué, en base a fechas de riego, haciendo posible que se logre detectar si las plantas necesitan agua o no la necesitan.

Agradecimientos

Agradecemos al Tecnológico Nacional de México, en especial al plantel: Instituto Tecnológico de Nogales, por las facilidades brindadas, tanto para la construcción del prototipo, como para el financiamiento para la presentación del artículo.

Conclusiones

A través de la investigación desarrollada, en la simulación del prototipo de la implementación del software, pruebas concluyentes a partir de imágenes tomadas a cultivos de uva se obtuvo la disminución del error, obtenido de 2.123 que resulta de la evaluación de la función cálculo de error con el aprendizaje obtenido en 30 pruebas, se redujo a 0.70 el grado de error lo que da una aceptabilidad del 93% para el algoritmo, que puede disminuir mucho más a través del aprendizaje.

ISSN-2531-2952

Herramientas como estas son fiables y seguras en su utilización, su aplicabilidad es basta en diversos cultivos de uva, al poder determinar si la planta tiene plagas en nuestro caso necrosis en la epidermis, seria de vital importancia para el productor en etapas tempranas de desarrollo, para poder combatir la plaga.

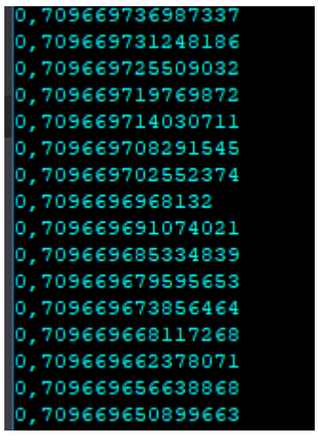

Figura 19 Disminución del grado de error
Fuente: Elaboración Propia

Se observó que la red neuronal alcanzaba o disminuía su grado de error hasta 0.5% después de tener un aprendizaje de 100000 iteraciones de una serie de 1036 datos el error se normaliza y no disminuye más.

Referencias

Alegre, E., Pajares, G., & de la Escalera, A. (2016). *Conceptos y Métodos en Visión por Computador*. España.

Ali, Z., & Mohammad, J. (2001). *Intelligent Control Systems Using Soft Computing Methodologies.* Boca Raton London New York Washington, D.C.: Crc Press Llc.

Anil K., J., Jianchang, M., & K. M., M. (1996). Artificial Neural Networks: A Tutorial. *IEEE*, 37.

Aurelio, V., Humberto, S., & Serguei, L. (2002). Reconocimiento Eficiente De Caracteres Alfanumericos Provenientes De Mapas Raster Por Medio De Clasificadores Neuronales. *Computacion Y Sistemas*, 040-041.

García, J. F. (2006). *Fundamentos para la Implementación de Red Neuronal Perceptrón Multicapa Mediante Software.* Escuela de Ingeniería Mecánica Eléctrica: Universidad de San Carlos de Guatemala Facultad de Ingeniería.

Giraldo, F. N. (2011). Algoritmo de procesamiento de imagenes satelitales con transformada de Hugh. *Vision Electronica*, 1-16.

González, G. F. (2009). *Redes Neuronales Aplicadas Al Análisis De Imágenes Para El Desarrollo De Un Prototipo De Un Sistema De Seguridad.* Pereira: Universidad Tecnológica De Pereira: Programa De Ingeniería En Sistemas Pereira.

González, P. J. (2015). *Reconocimiento de patrones proyectivo-invariantes mediante redes neuronales de variable compleja.* Oviedo, España: Universidad de Oviedo.

Hernández, S. N. (2014). *Implementación de un Sistema de Información para el Reconocimiento de Caracteres Basado en la Red.* Lima, Peru: Pontificia Universidad Católica del Perú - Facultad de Ciencias e Ingeniería.

La Serna Palomino, N., & Román Concha, U. (2009). Técnicas de Segmentación en Procesamiento Digital De Imagenes. *Revista De Ingenieria De Sistemas E Informatica*, 9-16.

Ramesh, J., Rangachar, K., & Brian G., S. (1995). *Machine Vision.* Mcgraw-Hill.

Szeliski, R. (2010). Computer Vision: Algorithms And Applications. *Springer*.

Vélez Serrano, J. F., Moreno Díaz, A. B., Sánchez Calle, Á., & Sánchez-Marín, J. L. (2003). *Visión por computador*. Madrid, España: S.L. - Dykinson.

ISSN-2531-2952