Informe de Trabajo Final

Curso: Procesamiento Avanzado de Señales e Imágenes

Integrantes:

Jorge Ronaldo Godiel Gálvez	U20171C442
Bryan Giomar Valdivieso Becerra	U201610269
Luis Harold Olano Tejada Olano	U201710107

Profesor: Christian Carlos Del Carpio Damián
Sección: LS8B

2021 - 01

Índice

Trabajo Final

1. Descripción del problema general del trabajo propuesto

Para el presente proyecto se requiere la realización de un programa con una interfaz gráfica haciendo uso de técnicas de inteligencia artificial como lo son las Máquinas de vectores de soporte (SVM). En esta interfaz gráfica será posible capturar una imagen a tiempo real de nuestro recinto en el cual se formarán las letras mayúsculas que se pueden observar en la Figura 1 con palitos de fósforos a los cuales se les cortó la cabeza.

Figura 1. Letras por detectar. Fuente: CanStock

Sin embargo, no todas las letras utilizan palitos de fósforos completos y solo utilizan la mitad de los fósforos, como lo son las letras G, J, M, Q, V, W y Y. Además de este detalle, se presentan ciertas limitantes en nuestro programa, como lo es que el programa detecta 3 letras mayúsculas a la vez, por lo que la posibilidad de solo detectar 1 o 2 letras queda descartada. La segunda limitante del proyecto es que las letras deben tener cierto espacio entre ellas, esto es para que el programa las detecte correctamente. Finalmente, la tercera limitante es que las letras no podrán estar rotadas y se leerán si es que se colocan de manera vertical. Esto sucede porque el programa no posee un algoritmo de rotación de imágenes, lo cual añadiría bastante la complejidad del proyecto.

En base a estas especificaciones y limitaciones es que se desarrolló el proyecto de adquisición e identificación de letras mayúsculas solicitado.

2. Objetivo general y objetivos específicos

2.1. Objetivo General

- ✓ Captura, procesamiento y reconocimiento de tres letras formadas con palitos de fósforos

2.2. Objetivos específicos

- ✓ Configuración de cámara para uso en Matlab
- ✓ Recuantización a escala de grises de un bit
- ✓ Identificación de objetos y clasificación por tipo de letra
- ✓ Creación de base de datos para el SVM

- ✓ Elaboración de Matriz Confusión
- ✓ Elaboración de GUIDE
- ✓ Visualización de resultados

3. Formulación de la hipótesis del trabajo propuesto

   Es posible realizar la adquisición y reconocimiento de tres letras mayúsculas formadas por palitos de fósforos mediante la implementación de máquinas de vectores de soporte implementadas en Matlab.

4. Descripción de los antecedentes de solución y los inconvenientes encontrados para dar solución al problema propuesto

   - ✓ Antecedentes de la solución

     Gracias al trabajo realizado en el laboratorio 1 del curso, ya se tenía listo un código que realizaba la adquisición e identificación de palitos de fósforo, con o sin cabeza, cortados o enteros y de cabeza roja o cabeza verde. Además, la interfaz gráfica del laboratorio fue utilizada como base para la realización de este proyecto.

   - ✓ Inconvenientes encontrados

     Debido a las limitaciones expuestas en la descripción del problema general propuesto, se presentaron letras las cuales generaban cierto error en nuestras primeras pruebas, por lo que se tuvo que añadir pruebas a nuestra base de datos para que nuestra SVM pueda tener un correcto entrenamiento. Además, debido a que no utilizamos un proceso de dilatación en el procesamiento de la imagen, los palitos de fósforo deberán estar correctamente colocados ya que una pequeña separación entre estos puede provocar un error en la identificación de las letras.

5. Descripción los métodos y técnicas algorítmicas que se han elegido para la solución del problema propuesto

   A continuación, se presenta la descripción de los métodos y técnicas utilizadas en el trabajo, esto incluye el recinto de adquisición de imágenes, introducción teórica, diagramas de bloques y explicación paso a paso de funcionamiento e interfaz visual.

   5.1. Recinto de adquisición de imágenes

   Con el fin de conseguir la captura las imágenes con una calidad óptima se construyó un recinto que nos permitiera mantener una iluminación y capacidad de enfoque adecuado sin importar la hora del día. Esto nos permitirá disminuir las condiciones que podrían generarnos una mayor variabilidad en nuestros datos de captura. El recinto cuenta con dimensiones de 40x55x40 cm para que se pueda manipular con facilidad los fósforos al formar las letras.

Figura 2. Vista interna y externa del recinto.

Para un mejor contraste de la imagen se utilizó una tela con poca reflectancia para disminuir el ruido por iluminación. La imagen fue capturada por una cámara web FULL HD 1080P.

Figura 3. Cámara Web empleada

5.2. Introducción teórica

Las técnicas para el procesamiento de imágenes se han desarrollado para cumplir dos principales objetivos: la mejora de la información pictórica para el ser humano y para el almacenamiento, transmisión y representación de la percepción autónoma de los dispositivos. Esto se ha logrado mediante el dimensionado bidimensional de la imagen obtenida a partir de una función f(x,y), donde (x,y) son la componente espacial del plano y es llamada intensidad o nivel de gris en ese punto. En este punto, denotamos que una imagen digital hace referencia a un proceso de digitalización, es decir, un conjunto de datos cuantificados que describen a una imagen basada en su ubicación y en su valor de intensidad.

Para describir de forma eficaz las imágenes vistas por una cámara se han desarrollado modelos de color que pueden transmitir con veracidad la forma en que el ser humano percibe e interpreta los colores. Los modelos más comunes son el RGB, CMY, CMYK, HSI. Para este trabajo se optó por aplicar el modelo de color RGB para la captura de las imágenes.

5.2.1. Modelo RGB

Es el modelo más usado en la actualidad por los monitores y cámaras de video, este describe las imágenes en sus componentes de color primario rojo, verde y azul. Este modelo se basa en un sistema de coordenadas cartesianas donde en tres esquinas se encuentran los colores primarios y en las otras se encuentran los colores secundarios, es decir, celeste, magenta y amarillo. Además, este modelo expresa la escala de grises a partir de una línea que une dos puntos en el curvo, el blanco y el negro. Para este modelo presentado los valores de R, G y B se expresan en un rango de 0 a 1, para modelos de 8 bits se puede expresar de 0 a 255 y en modelos de 24 bits hasta 16777216.

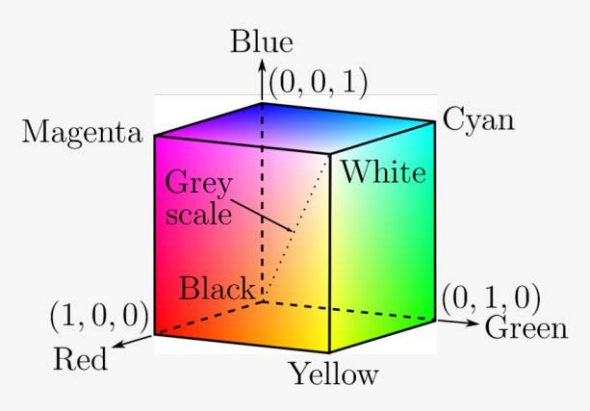

Figura 4. Modelo de color RGB

En la presente imagen podemos visualizar la composición de la imagen captura RGB a partir de sus componentes R,G,B.

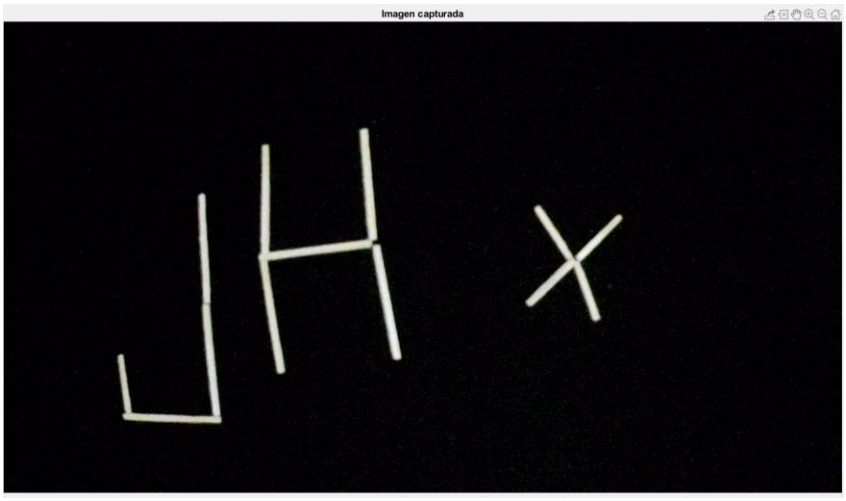

Figura 5. Captura de imagen RGB. Elaboración propia.

5.2.2. Descripción del proceso de binarización

Debido a que el sensado de cualquier señal presenta una carga computacional, se suele utilizar profundidades de bits diversas para describir las imágenes adquiridas. En ese sentido, la imagen sensada obtendrá una resolución descrita por la cantidad de pixeles usada y por el tono de color. En el presente trabajo se describió la imagen adquirida con un r de 1 bit de profundidad, lo que significa que expresa una tonalidad de gris en cada pixel. Se presenta la imagen recuantizada a 1 bit de tonalidad.

Figura 6. Recuantización de una imagen en escala de grises a r=1. Elaboración propia.

La recuantización se realiza a partir de la siguiente ecuación:

$$I_Q(x,y) = round(\frac{I(x,y)(2^{rp}-1)}{255}) \quad I_F(x,y) = round(\frac{255 \times I(x,y)}{(2^{rp}-1)})$$

5.2.3. Vecindad y conectividad

Se reconoce vecindad como el espacio que rodea a un pixel. Para un pixel p con coordenadas (x,y) existen cuatro vecinos horizontales y verticales que puede ser expresado por (x+1,y),(x-1,y),(x,y+1),(x,y-1). Este tipo de relación es conocido como 4-conectividad $N_4(p)$

	(x,y-1)	
(x-1,y)	(x,y)	(x+1,y)
	(x,y+1)	

Figura 7. 4-Conectividad. Elaboración propia.

Los vecinos en diagonal de p, a los que se le denota como $N_D(p)$ presentan las siguientes coordenadas (x+1, y+1),(x+1,y-1),(x-1,y+1),(x-1,y-1) y en conjunto con los vecinos de 4-conectividad son conocidos como 8-conectividad $N_8(p)$.

(x-1, y-1)	(x, y-1)	(x+1, y-1)
(x-1, y)	(x,y)	(x+1, y)
(x-1, y+1)	(x, y+1)	(x+1, y+1)

Figura 8. 8-Conectividad. Elaboración propia.

Este análisis fue aplicado para identificar los fósforos en la imagen y ser tratados como objetos.

5.3. Diagrama de Bloques

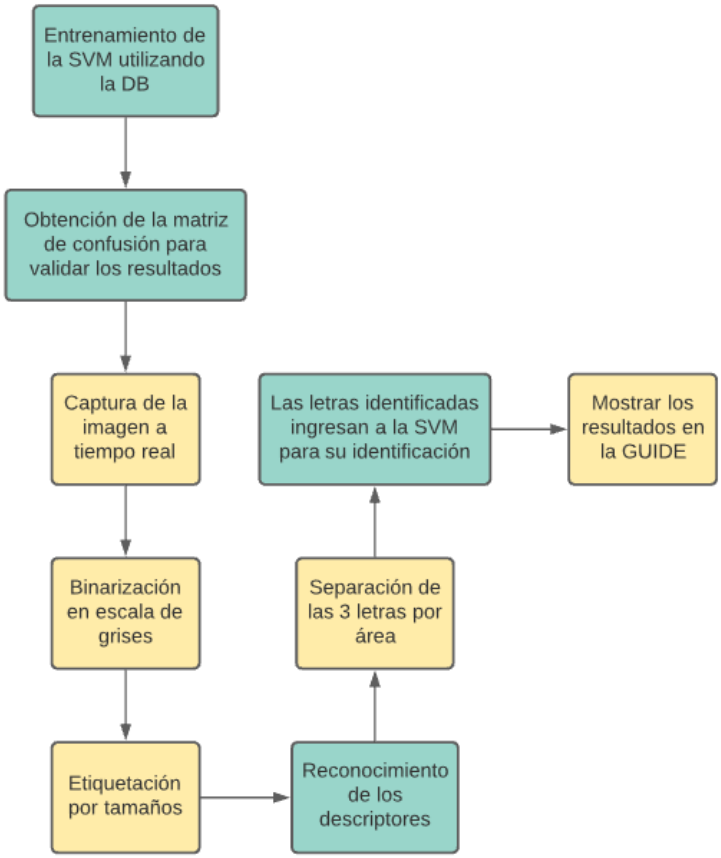

Figura 9. Diagrama de Bloques del programa. Fuente: Elaboración propia

5.4. Explicación paso a paso de funcionamiento e interfaz visual

Figura 10. Interfaz Visual. Fuente: Elaboración propia.

Al ejecutarse el programa aparecerá la interfaz que se puede ver en la Figura 10, este cuenta con dos botones, el primero “Entrenar” desplegará una ventana que contendrá la matriz de confusión del sistema, el segundo botón “Capturar” realizará la captura de la imagen, generará la imagen recuantizada a 1 bit y las gráficas de las letras reconocidas por el programa por separado, a su vez que se presentará en el cuadro de resultados las letras reconocidas.

6. Resultados de detección

Para poder visualizar correctamente los resultados de detección se realizaron múltiples pruebas y así verificar la performance de la SVM en múltiples escenarios.

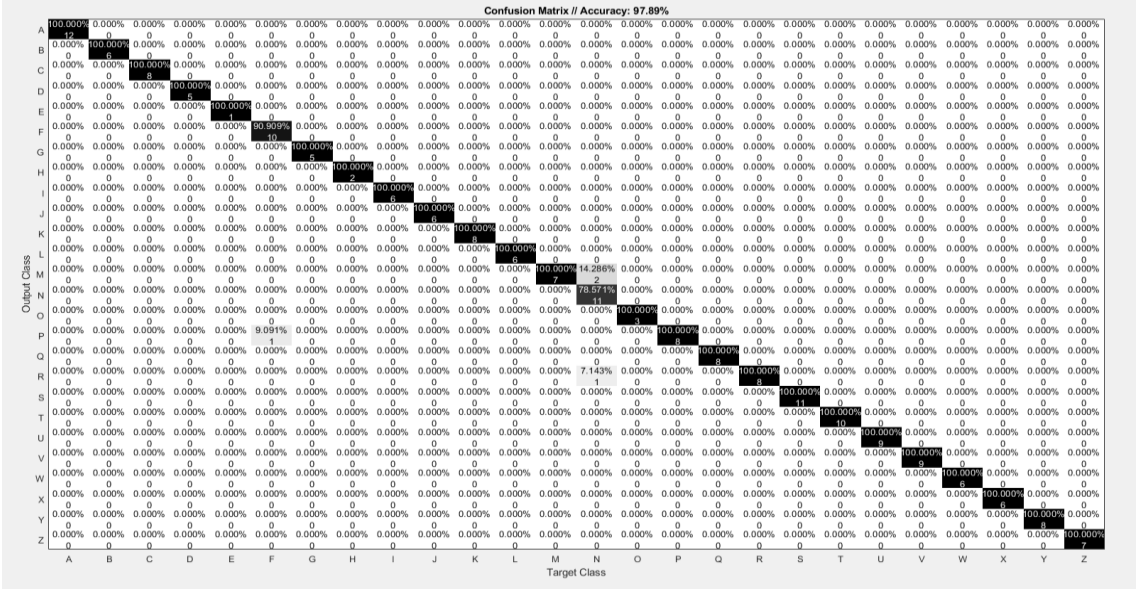

Figura 11. Matriz de Confusión obtenida. Fuente: Elaboración propia.

Figura 12. Primera prueba de identificación de letras. Fuente: Elaboración propia.

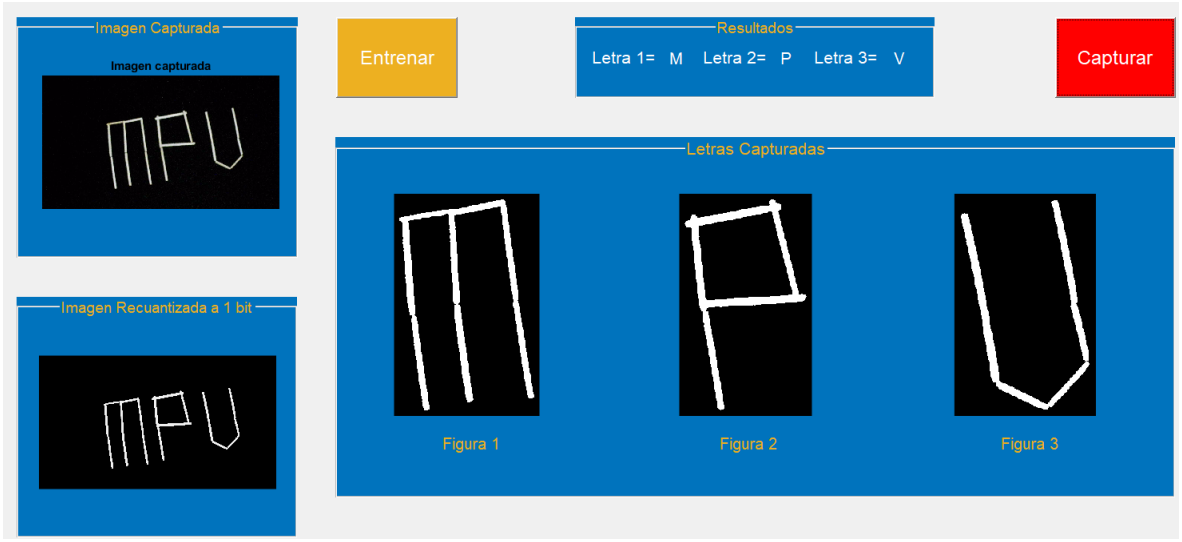

Figura 13. Segunda prueba de identificación de letras. Fuente: Elaboración propia.

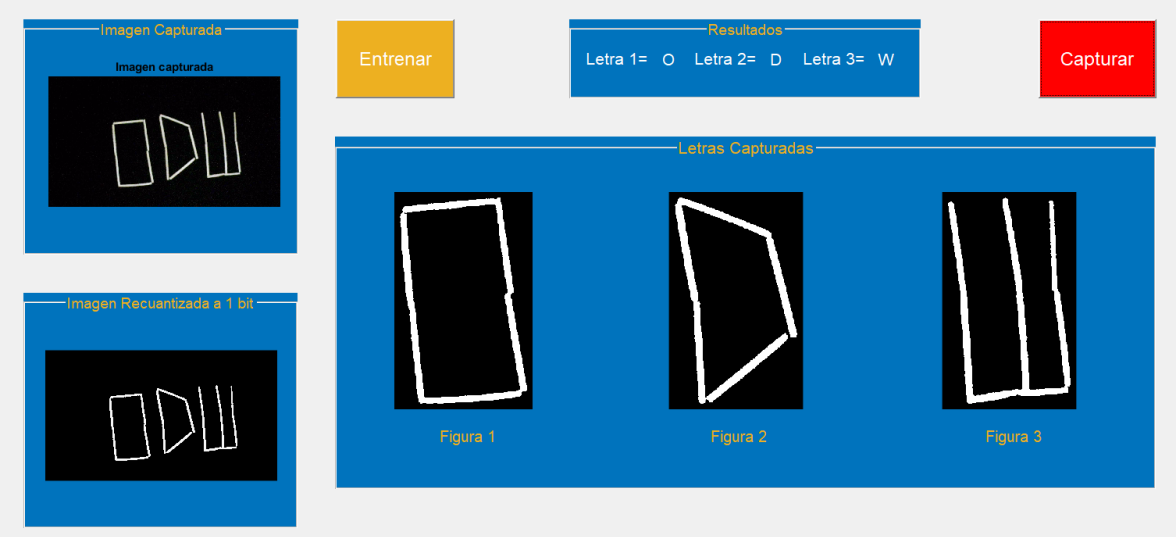

Figura 14. Tercera prueba de identificación de letras. Fuente: Elaboración propia.

Figura 15. Cuarta prueba de identificación de letras. Fuente: Elaboración propia.

Figura 16. Quinta prueba de identificación de letras. Fuente: Elaboración propia.

Figura 17. Sexta prueba de identificación de letras. Fuente: Elaboración propia.

Figura 18. Séptima prueba de identificación de letras. Fuente: Elaboración propia.

Figura 19. Octava prueba de identificación de letras. Fuente: Elaboración propia.

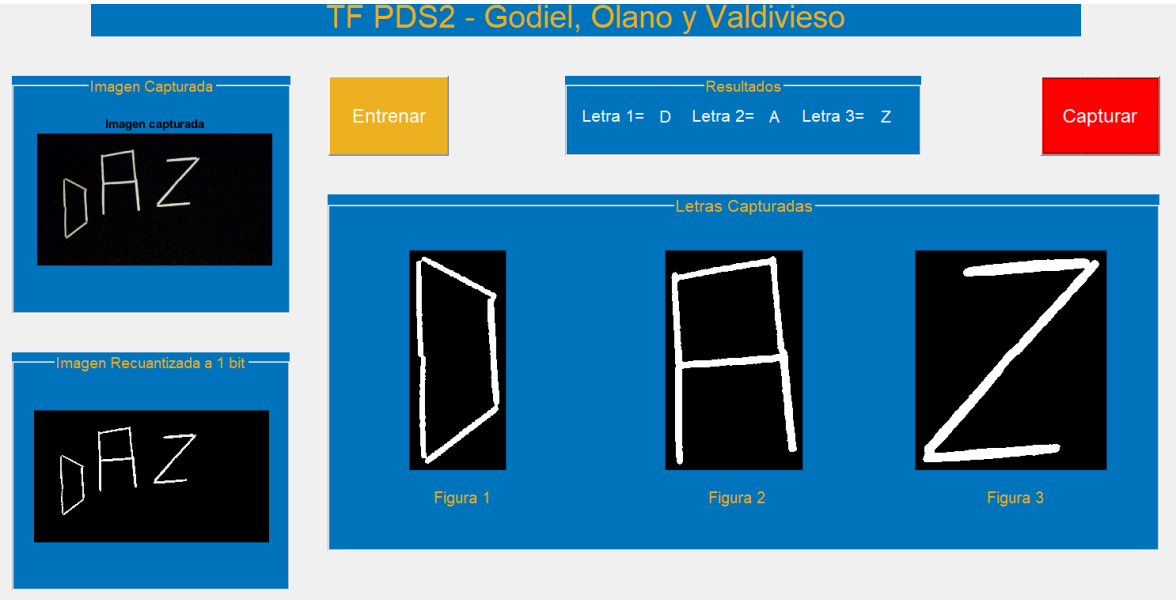

Figura 20. Novena prueba de identificación de letras. Fuente: Elaboración propia.

7. Análisis y discusión de los resultados

7.1. Mejores Resultados

Los mejores resultados se presentaron cuando las letras se encontraban en el área central de la tela, ya que la iluminación es óptima en dicha zona. Además, las letras que no tienen similitudes con las demás, como por ejemplo, la L, la X, la T, la I o la J, no presentaron errores en las pruebas realizadas.

7.2. Resultados Intermedios

Los resultados intermedios se presentaron cuando las letras se encontraban entre la zona central de la tela y el límite. En estas zonas la iluminación no era mala, pero tampoco era óptima, por lo que podía generar ciertos errores. Además, las letras que tenían cierta similitud entre ellas como la O, la B, la Q, la R o la P presentaban ciertos errores en ocasiones.

7.3. Peores Resultados

Los peores resultados se presentaron cuando las letras se ubican bastante cerca al límite del alcance de la cámara. En este punto la iluminación no era buena, por lo que la imagen no se procesaba de manera correcta y la SVM no podía identificar de manera correcta la letra formada por los palitos de fósforos.

8. Aplicaciones que puede tener la solución implementada

Los sistemas Machine Learning presentan una gran variedad de aplicaciones tanto en el ámbito médico como en el de seguridad. En el ámbito médico, se han realizado diversas aplicaciones para mejorar el procesamiento de las tareas de tomografías computarizadas e Imágenes por Resonancia Magnética (MRI). (Chin et al., 2018) desarrollo un sistema de reconocimiento de esquizofrenia mediante SVM aplicado en MRI. El sistema es

empleado como herramienta de diagnóstico y pronóstico de las anomalías cerebrales estructurales del paciente. Esto se logró con el entrenamiento del sistema con una base de datos de 127 personas, consiguiendo así una precisión del 86.6% en primera instancia y en combinación con la selección secuencial de región de interés (ROI) obteniéndose un 92.0% en precisión y 89.4% de validación de los datos.

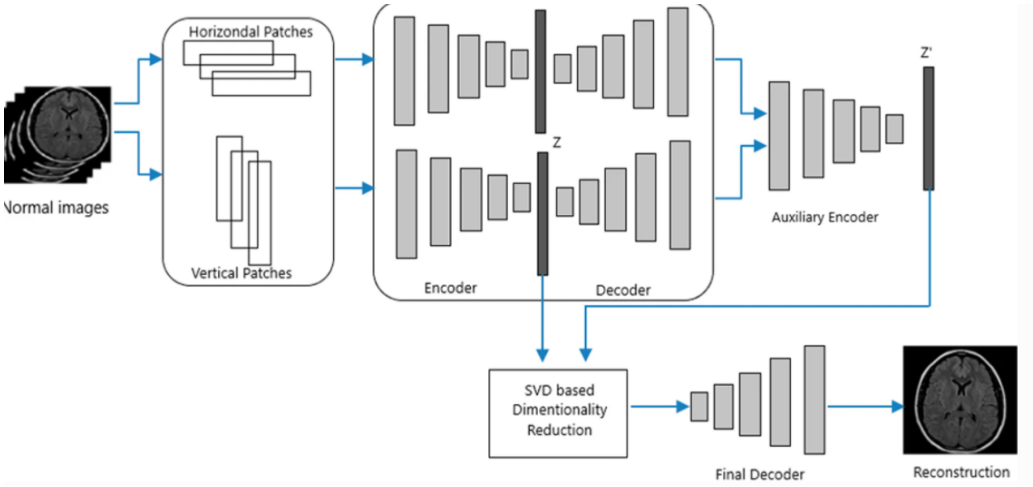

Figura 21. Método del modelo propuesto por (Chin et al., 2018)

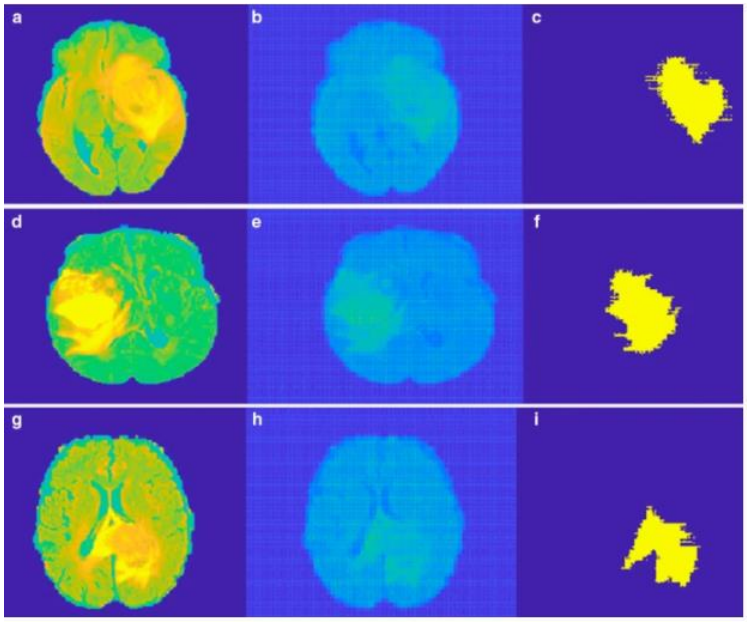

Figura 22. Segmentación del tumor obtenido por MRI. Extraído de (Chin et al., 2018)

En el área de seguridad, (Othman et al., 2018) desarrolla un sistema de detección de intrusiones (IDS) usando máquina de vectores de soporte (SVM), con este modelo monitorea y analiza las firmas de un sistema o red a alta velocidad para prever cualquier ataque. El sistema incorpora Big Data y SVM para un análisis de dato preciso y eficiente usando la plataforma Apache Spark Big Data.

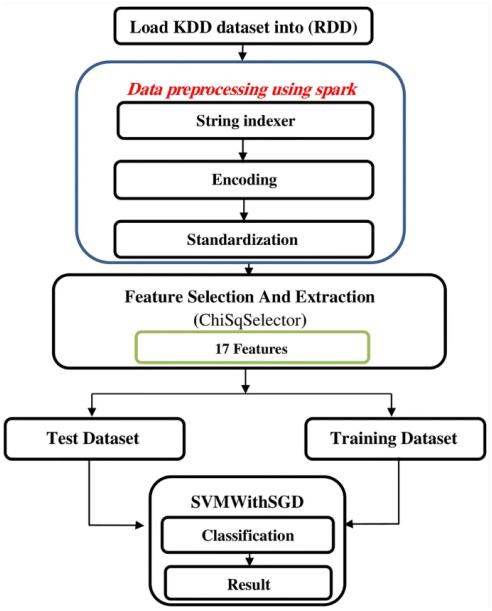

Figura 23. Modelo del sistema IDS. Extraído de (Othman et al., 2018)

De esta forma, el sistema que hemos desarrollado puede ser utilizado en sistemas de detección de placas automovilísticas. Este sistema utilizaría cámaras al aire libre ubicadas a la altura del coche en las entradas de estacionamientos o centros comerciales. El sistema podría ser utilizado como método de monitoreo, control y ubicación de los autos para mejorar seguridad en instalaciones de acceso público. Por otro lado, el sistema podría ser utilizado para la identificación del tipo de rubro para designar el costo de peaje en los accesos de la ciudad o para el reconocimiento de multas vehiculares. Esto implicaría un gran avance en la automatización de los sistemas de seguridad pública y privada.

Figura 24. Fotografía de placa automovilística

9. Comentarios y conclusiones

- ✓ Se cumplió con los requerimientos solicitados para este proyecto
- ✓ El SVM tiene la particularidad de no mejorar su tasa de error con cada iteración, esto lo diferencia de algoritmos más complejos como las redes neuronales, por ejemplo, las redes neuronales de tipo Perceptron son redes multicapa que disminuyen el error por cada iteración.

- ✓ El SVM es un tipo de Machine Learning que puede ser utilizado en tareas del ámbito médico como de identificación de data de seguridad, por ejemplo, en el reconocimiento de tumores por medio de tomografías computarizadas o Imágenes por Resonancia

Magnética (MRI) e identificación de detección de intrusos (IDS) respectivamente. Esto demuestra el gran potencial que posee esta técnica para su uso con sistemas computarizados más completos y especializados.

- ✓ El uso de una base de datos de gran tamaño es ideal para reducir el porcentaje de error

10. Referencias bibliográficas

Chin, R., You, A. X., Meng, F., Zhou, J., & Sim, K. (2018). Recognition of Schizophrenia with Regularized Support Vector Machine and Sequential Region of Interest Selection using Structural Magnetic Resonance Imaging. *Scientific Reports*, *8*(1), 13858. https://doi.org/10.1038/s41598-018-32290-9

Othman, S. M., Ba-Alwi, F. M., Alsohybe, N. T., & Al-Hashida, A. Y. (2018). Intrusion detection model using machine learning algorithm on Big Data environment. *Journal of Big Data 2018 5:1*, *5*(1), 1-12. https://doi.org/10.1186/S40537-018-0145-4

GONZALEZ, Rafael C.Woods, Richard E. (2001) Digital image processing. New Jersey : Prentice Hall. (621.367 GONZ).

Rgb Color Model Cube, HD Png Download , Transparent Png Image - PNGitem. (n.d.). Retrieved June 24, 2021, de https://www.pngitem.com/middle/ThRhwmb_rgb-color-model-cube-hd-png-download/

MATHWORKS. (s. f.). Support Vector Machine (SVM) - MATLAB & Simulink. Recuperado 9 de julio de 2021, de https://www.mathworks.com/discovery/support-vector-machine.html

11. Programa comentado

```
function varargout = TF_PDS2_Godiel_Olano_Valdivieso(varargin)
%      TF_PDS2_GODIEL_OLANO_VALDIVIESO      MATLAB      code      for
TF_PDS2_Godiel_Olano_Valdivieso.fig
%            TF_PDS2_GODIEL_OLANO_VALDIVIESO,   by   itself,   creates   a   new
TF_PDS2_GODIEL_OLANO_VALDIVIESO or raises the existing
%      singleton*.
%
%      H   =   TF_PDS2_GODIEL_OLANO_VALDIVIESO   returns   the   handle   to   a   new
TF_PDS2_GODIEL_OLANO_VALDIVIESO or the handle to
%      the existing singleton*.
%
%      TF_PDS2_GODIEL_OLANO_VALDIVIESO('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in TF_PDS2_GODIEL_OLANO_VALDIVIESO.M with the given
input arguments.
%
%          TF_PDS2_GODIEL_OLANO_VALDIVIESO('Property','Value',...)  creates  a  new
TF_PDS2_GODIEL_OLANO_VALDIVIESO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied  to  the  GUI  before  TF_PDS2_Godiel_Olano_Valdivieso_OpeningFcn  gets
called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All  inputs  are  passed  to  TF_PDS2_Godiel_Olano_Valdivieso_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
```

```
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TF_PDS2_Godiel_Olano_Valdivieso

% Last Modified by GUIDE v2.5 09-Jul-2021 03:03:17

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TF_PDS2_Godiel_Olano_Valdivieso_OpeningFcn,
...
                   'gui_OutputFcn',    @TF_PDS2_Godiel_Olano_Valdivieso_OutputFcn,
...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
      gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
      [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
      gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before TF_PDS2_Godiel_Olano_Valdivieso is made visible.
function  TF_PDS2_Godiel_Olano_Valdivieso_OpeningFcn(hObject,  eventdata,  handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%  varargin     command  line  arguments  to  TF_PDS2_Godiel_Olano_Valdivieso  (see
VARARGIN)

% Choose default command line output for TF_PDS2_Godiel_Olano_Valdivieso
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes TF_PDS2_Godiel_Olano_Valdivieso wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = TF_PDS2_Godiel_Olano_Valdivieso_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global trainedClassifierSVM
global validationAccuracy


closepreview;

vid=videoinput('winvideo',1); % se escoge la camara web
preview(vid);
pause(5);
I = getsnapshot(vid); %captura de foto
axes(handles.axes1)%imagen tomada
imshow(uint8(I)); % se muestra la foto tomada
impixelinfo;
title('Imagen capturada');
%Foto en escala de grises sin recuantizar y tamano de la imagen
Igray=rgb2gray(I);
[M,N]=size(Igray); % determina el tamaño de la imagen.

%-------------------Recuantizacion de imagenes en escala de grises------
r=1;  %Para  hacer  procesamiento,  los  valores  de  las  imágenes  tiene  que  %ser
convertidas a datos tipo double.
Igray=double(Igray);
fe=max(max(Igray));
Irecuan=round(Igray*((2^r)-1)/fe);
Irecuan=round(Irecuan*fe/((2^r)-1));
axes(handles.axes2)
imshow(uint8(Irecuan));
impixelinfo;
title('Recuantizacion de imagen escala de grises a r=1');
[Iobjeto,numobjeto]  =  bwlabel(Irecuan,8);  %  obtencion  de  numeros  de  objetos
preliminar

% Variables de calculo de fosforos
AMIN=2000; % tamaño minimo de un palo entero
prom=2200; %tamaño promedio de un palo entero
A=round(prom*1.5);  %Tamaño minimo para detectar un monton de palos
mayores=0;
menores=0;
vectormayor=[];
vectormenor=[];

for i=1:numobjeto
       ip=find(Iobjeto==i);
       Ap=length(ip);
       if Ap<AMIN && Ap>50  %Ap= tamaño palo analizado
       menores=menores+1;
          vectormenor=[vectormenor,Ap];
       end;
       if Ap>A
       mayores=mayores+1;
          vectormayor=[vectormayor,Ap];
       end;
end;
disp(strcat('se han detectado : ',num2str(mayores),'montones de palos'));
```

```
disp(strcat('se han detectado : ',num2str(menores),'palos cortados'));
cuentapalosmayores=0;
listamayores=[];
for j=1:length(vectormayor)
    cantidadmayores=vectormayor(j)/prom; % Tamaño promedio de un palo
       listamayores=[listamayores,cantidadmayores];
end;
sumamayores=sum(listamayores);
totalamontonados=round(sumamayores);
aux1=numobjeto-length(vectormayor)+totalamontonados;
disp(strcat('se han detectado : ',num2str(aux1),' palos en total'));

imshow(uint8(Irecuan));
[V,K]=bwlabel(Irecuan,8);
helper=0;
for i=1:K
 ip=find(V==i);
 AParea=length(ip);

 if AParea>300
       if helper==0
       ip1=ip;
       helper=helper+1;
       elseif helper==1
       ip2=ip;
       helper=helper+1;
       elseif helper==2
       ip3=ip;
       helper=helper+1;
       end
 end
end
helper=0;
%SEPARAR OBJETOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Func1=zeros(M,N);
Func1(ip1)=255;
Func2=zeros(M,N);
Func2(ip2)=255;
Func3=zeros(M,N);
Func3(ip3)=255;
[x1,y1]=find(Func1==255);
[x2,y2]=find(Func2==255);
[x3,y3]=find(Func3==255);
%SEPARAR OBEJETOS cortaobjetos

Areaobj1=Func1(min(x1-10):max(x1+10),min(y1-10):max(y1+10));
Areaobj2=Func2(min(x2-10):max(x2+10),min(y2-10):max(y2+10));
Areaobj3=Func3(min(x3-10):max(x3+10),min(y3-10):max(y3+10));

%GRAFICAR OBJETOS SEPARADOS esto se mete al boton de mostrar procesos de la
%guide
axes(handles.axes3)
imshow(uint8(Areaobj1));
impixelinfo;
axes(handles.axes4)
imshow(uint8(Areaobj2));
impixelinfo;
axes(handles.axes5)
```

```
imshow(uint8(Areaobj3));
impixelinfo;
%DATOS del objeto
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

BW1=im2bw(Areaobj1,0.5);
dato1=regionprops(BW1,'Perimeter','Area','Eccentricity','Orientation','Extent','Mi
norAxisLength','MajorAxisLength');%'Extrema'
BW2=im2bw(Areaobj2,0.5);
dato2=regionprops(BW2,'Perimeter','Area','Eccentricity','Orientation','Extent','Mi
norAxisLength','MajorAxisLength');%'Extrema'
BW3=im2bw(Areaobj3,0.5);
dato3=regionprops(BW3,'Perimeter','Area','Eccentricity','Orientation','Extent','Mi
norAxisLength','MajorAxisLength');%'Extrema'

[M1,N1,P1]=size(BW1);
SEP_A=zeros(M1,N1);
SEP_A(1:(M1-115),:)=1;
SEP_B=zeros(M1,N1);
SEP_B((M1-115):M1,:)=1;
GAA=SEP_B.*BW1;
GAA2=SEP_A.*BW1;
BAJON=regionprops(GAA,'Area');
ALTON=regionprops(GAA2,'Area');
BajoLetra=BAJON(1).Area;
AltoLetra=ALTON(1).Area;
Area=dato1(1).Area;
EjePrincipal=dato1(1).MinorAxisLength;
EjeMenor=dato1(1).MajorAxisLength;
Perimetro=max(dato1.Perimeter);
disp(strcat('Perimetro 1 : ',num2str(Perimetro)));
disp(strcat('Area 1 : ',num2str(Area)));
disp(strcat('EjePrincipal 1 : ',num2str(EjePrincipal)));
disp(strcat('EjeMenor 1 : ',num2str(EjeMenor)));
disp(strcat('AltoLetra 1 : ',num2str(AltoLetra)));
disp(strcat('BajoLetra 1 : ',num2str(BajoLetra)));
disp(strcat(''));

LETRA_1=table(Area,Perimetro,EjePrincipal,EjeMenor,AltoLetra,BajoLetra);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[M2,N2,P2]=size(BW2);
SEP_A=zeros(M2,N2);
SEP_A(1:(M2-115),:)=1;
SEP_B=zeros(M2,N2);
SEP_B((M2-115):M2,:)=1;
GAA=SEP_B.*BW2;
GAA2=SEP_A.*BW2;
BAJON=regionprops(GAA,'Area');
ALTON=regionprops(GAA2,'Area');
BajoLetra=BAJON(1).Area;
AltoLetra=ALTON(1).Area;
Area=dato2(1).Area;
EjePrincipal=dato2(1).MinorAxisLength;
EjeMenor=dato2(1).MajorAxisLength;
Perimetro=max(dato2.Perimeter);
disp(strcat('Perimetro 2 : ',num2str(Perimetro)));
disp(strcat('Area 2 : ',num2str(Area)));
disp(strcat('Ejeprinciplal 2 : ',num2str(EjePrincipal)));
disp(strcat('EjeMenor 2 : ',num2str(EjeMenor)));
```

```
disp(strcat('AltoLetra 2 : ',num2str(AltoLetra)));
disp(strcat('BajoLetra 2 : ',num2str(BajoLetra)));
disp(strcat(''));
LETRA_2=table(Area,Perimetro,EjePrincipal,EjeMenor,AltoLetra,BajoLetra);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[M3,N3,P3]=size(BW3);
SEP_A=zeros(M3,N3);
SEP_A(1:(M3-115),:)=1;
SEP_B=zeros(M3,N3);
SEP_B((M3-115):M3,:)=1;
GAA=SEP_B.*BW3;
GAA2=SEP_A.*BW3;
BAJON=regionprops(GAA,'Area');
ALTON=regionprops(GAA2,'Area');
BajoLetra=BAJON(1).Area;
AltoLetra=ALTON(1).Area;
Area=dato3(1).Area;
EjePrincipal=dato3(1).MinorAxisLength;
EjeMenor=dato3(1).MajorAxisLength;
Perimetro=max(dato3.Perimeter);
disp(strcat('Perimetro 3 : ',num2str(Perimetro)));
disp(strcat('Area 3 : ',num2str(Area)));
disp(strcat('Ejeprinciplal 3 : ',num2str(EjePrincipal)));
disp(strcat('EjeMenor 3 : ',num2str(EjeMenor)));
disp(strcat('AltoLetra 3 : ',num2str(AltoLetra)));
disp(strcat('BajoLetra 3 : ',num2str(BajoLetra)));
LETRA_3=table(Area,Perimetro,EjePrincipal,EjeMenor,AltoLetra,BajoLetra);
T1=LETRA_1;
RESPUESTA_1=predict(trainedClassifierSVM,T1{:,trainedClassifierSVM.PredictorNames}
)
T2=LETRA_2;
RESPUESTA_2=predict(trainedClassifierSVM,T2{:,trainedClassifierSVM.PredictorNames}
)
T3=LETRA_3;
RESPUESTA_3=predict(trainedClassifierSVM,T3{:,trainedClassifierSVM.PredictorNames}
)
set(handles.text10,'String',RESPUESTA_1);
set(handles.text11,'String',RESPUESTA_2);
set(handles.text12,'String',RESPUESTA_3);

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%      str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if            ispc            &&            isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
```

```
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if            ispc            &&            isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%      See ISPC and COMPUTER.
if            ispc            &&            isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global trainedClassifierSVM;
global validationAccuracy;

load ('TF PDS2.mat', 'BasedeDatosTFPDS2')

[q,t]=size(BasedeDatosTFPDS2);
ind=randperm(q,q);
for i=1:q
    BasedeDatosTFPDS2(i,:)=BasedeDatosTFPDS2(ind(i),:);
end
```

```
Tentrena=BasedeDatosTFPDS2(1:round(q*70/100),:); %El 70% para el entrenamiento
Tvalida=BasedeDatosTFPDS2(round(q*70/100)+1:end,:); %El 30% para la validacion

% ENTRENO LA RED
[trainedClassifierSVM, validationAccuracy] = trainClassifierSVM(Tentrena);

% Valido la red con los 30% de datos
%trainedClassifier es la red entrenada
%ysal es la salida de la red cuando se le ingresa un nuevo dato
[a,b]=size(Tvalida);
yfit=[];
for i=1:a
       T=Tvalida(i,1:end-1);
    ysal=predict(trainedClassifierSVM, T{:,trainedClassifierSVM.PredictorNames});
       yfit = [yfit; ysal];
end
res=[Tvalida(:,end),yfit];
test=categorical(Tvalida.Letra);
yt=categorical(yfit);
[cm,order]=confusionmat(test,yt);

figure()
plotConfMat2(cm,{'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P'
'Q' 'R' 'S' 'T' 'U' 'V' 'W' 'X' 'Y' 'Z'})
```