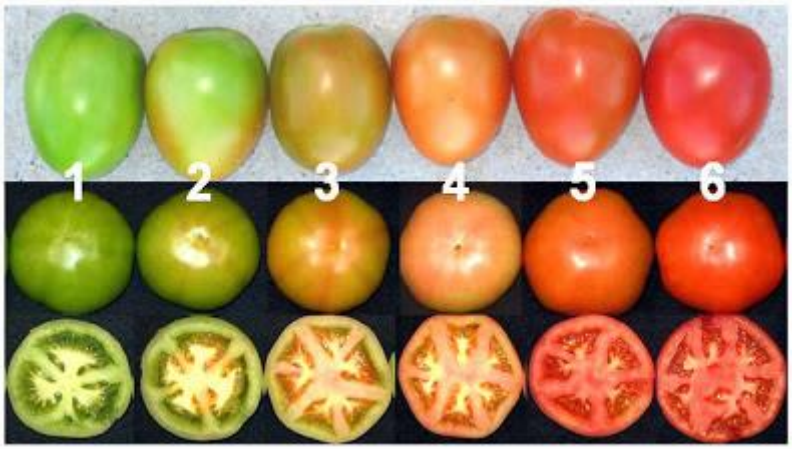
Universidad Rafael Landívar
Campus Quetzaltenango
Facultad de Ingeniería
Inteligencia Artificial

Avances Proyecto IA:
Tomato Analyzer

Barrientos Soto, Jorge Mario 15615-14
Guzmán Maldonado, Sergio Humberto 15106-15
Viernes 04 de mayo de 2018

Tomate

Es una especie de planta herbácea y es nativa de centro y sudamérica, su uso como comida se habría originado en méxico, es cultivada en el mundo entero para su consumo tanto fresco como procesado de diferentes modos(salsa, puré, enlatado, deshidratado). Una fruta es la parte de la planta que contiene las semillas, por lo tanto, botánicamente hablando, el tomate es una fruta, y nos decidimos por esta fruta por su fácil acceso y disponibilidad durante cualquier época del año, y también por la fácil identificación de sus grados de madurez y el apoyo que tenemos de distintos conocidos que son agrónomos, y se le podrían realizar consultas sobre esta fruta.



La mayoría de los tomates del tipo saladette o bola son cosechados en el estado de maduración conocido como estrella o rayado [etapa de color 2], aunque algunos pocos son cosechados en el estado cambiante [etapa de color 3 (Turning)] o en el estado rosa [etapa de color 4 (Pink)] (en la imagen). Sin embargo, cuando se incrementa la demanda de tomates en los mercados nacionales y de exportación, los frutos pueden ser cosechados en estado verde maduro (etapa de color 1) que es cuando están fisiológicamente maduros aunque por fuera todavía no muestran coloraciones típicas de la maduración.

Grado de Madurez

La madurez del tomate está dada por la siguiente tabla, donde se han establecido 5 etapas o 5 grados de madurez, donde se puede visualizar el porcentaje de color verde y el porcentaje de color rojo que tiene el tomate en dicho grado de madurez. Aunque los resultados no son constantes, dicho porcentaje se acerca a los niveles de color de verde y rojo.

Grado de madurez	Color (%)	
	Verde	Rojo
1	100	0
2	75	25
3	50	50
4	25	75
5	0	100

Tiempo para maduración del Tomate según el grado de madurez

TABLA 2. Valores promedio para el pH medido en el jugo y la resistencia de la pulpa al penetrómetro en frutos de tomate de tres híbridos cosechados en diferentes grados de madurez.

Híbrido	Grado de maduración	pH	Firmeza ($lb \cdot pulg^{-2}$)	Tiempo para maduración (d)
Marimba	1	4,45 ab	9,17 ab	15 e
	2	4,46 ab	9,08 ab	9 bc
	3	4,58 bc	9,17 ab	8 b
	4	4,68 bc	9,06 ab	4 a
	5	4,87 c	9,10 ab	
Sofía	1	4,34 ab	10,00 bcd	15 e
	2	4,48 ab	10,20 cd	14 e
	3	4,38 ab	9,99 bcd	11 d
	4	4,36 ab	9,79 abcd	10 cd
	5	4,43 ab	9,68 cd	
Bravona	1	4,22 a	10,56 e	26 f
	2	4,58 cd	9,52 abc	15 e
	3	4,55 abc	9,80 abcd	9 bc
	4	4,49 ab	9,41 abc	9 bc
	5	4,67 bc	8,88 a	
CV (%)		5,24	6,83	38,56

CV, coeficiente de variación.
Promedios con letras iguales no presentan diferencias significativas, según la prueba de Tukey ($P<0,01$).

Librerias a utilizar
https://pythonhosted.org/neurolab/ //Para la construcción y entrenamiento de la red neuronal.
https://opencv.org/ //Para detección y mapeo de las imagenes.

Descripción del Entorno del Proyecto

Accesible: Ya que conocemos todos los datos de la fruta, en este caso el tomate.
No Determinista: Ya que no le importa lo hecho anteriormente.
Episódico: Ya que primero deberá recortar la imagen y localizar en qué parte se encuentra la fruta.
Discreto: Se conoce el número exacto de salidas, las cuales será 5 salidas según el grado de madurez, una salida que indique que el tomate está podrido y una última que indique error o que no se reconoce el tomate.
Estático: No hay otro agente que intervenga.

Episodio 1: Convertir la imagen a datos que puedan ser usados como entrada.
Accesible: Ya que conocemos todos los datos de la fruta, en este caso el tomate.
No Determinista: No le importa lo hecho anteriormente.
No Episódico: Solo se mapean los datos de entrada de la imagen.
Discreto: Se conoce el número de salidas, que será una matriz de las dimensiones de la imagen.
Estático: No hay otro agente que intervenga.

Episodio 2: Recortar la imagen, dejando solamente el contorno de la fruta, donde los bordes queden lo más cerca al contorno posible.
Accesible: Ya que conocemos todos los datos de la fruta, en este caso el tomate.
No Determinista: No le importa lo hecho anteriormente.
No Episódico: Solo se mapean los datos de entrada de la imagen.
Discreto: Se conoce el número de salidas, que será una matriz de las dimensiones de la imagen.
Estático: No hay otro agente que intervenga.

Episodio 3: Saca una muestra de los colores del tomate. Se generarán máscaras donde se separan los colores.
Accesible: Ya que conocemos todos los datos de la fruta, en este caso el tomate.
No Determinista: No le importa lo hecho anteriormente.
No Episódico: Es un único episodio, que se encarga de sacar muestras.
Discreto: Se conoce el número de salidas, que será una matriz de las dimensiones de la imagen.
Estático: No hay otro agente que intervenga.

Episodio 4: Toma todos los datos anteriores y a través de una Red Neuronal obtiene el nivel de madurez que se encuentra la fruta. Es la etapa más importante de todas.
Accesible: Ya que conocemos los datos de la muestra.
No Determinista: No le importa lo hecho anteriormente.
No Episódico: Es un único episodio, una Red que determinará el nivel de madurez.
Discreto: Se conoce el número de salidas, serán 7 salidas, 5 según el nivel de madurez, otra que indique que el tomate está podrido y una última salida que será la del error.
Estático: No hay otro agente que intervenga.

Episodio 5: Con los datos de salida de la red neuronal se determinará el tiempo en que la fruta, tomate, será consumible para el ser humano. Además de otros datos de interés como la calidad de la fruta y el tiempo que le falta para el consumo óptimo.

Accesible: Se conocen todas las entradas, que será en base a la red neuronal del episodio anterior.
No Determinista: No le importa lo hecho anteriormente.
No Episódico: Solo determinan los datos relacionados al tomate
Discreto: Se conoce el número de salidas, que serán los datos del tomate (Calidad y tiempo de consumo).
Estático: No hay otro agente que intervenga.

Primera Prueba

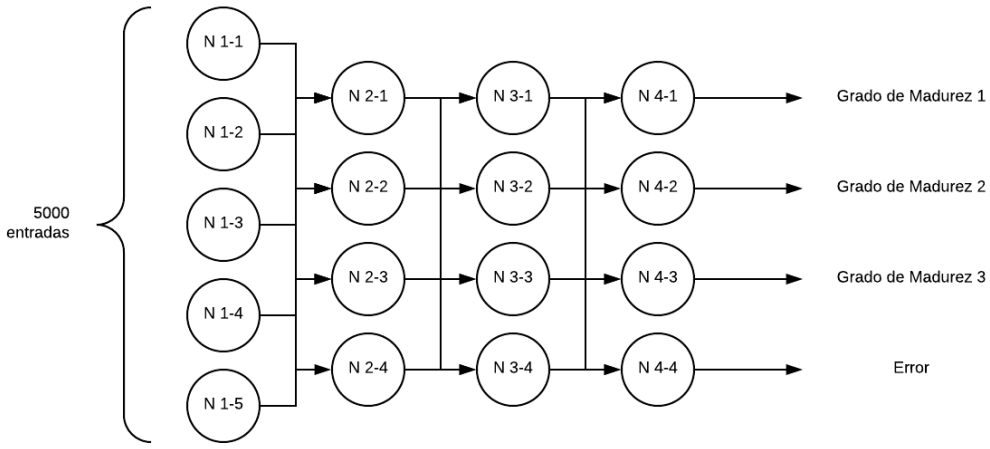
Para esta primera prueba se usará una imagen de 50 pixels por 50 pixels. Por el momento solo se tendrán tres niveles de maduración, esto con el objetivo de que el diseño de la RNA sea un poco más simple.

Nivel 1	Nivel 2	Nivel 3
Inmaduro	Medio Maduro	Maduro

Nota: Se Usaron 9 patrones de entrenamiento, 3 por cada nivel de madurez descritos anteriormente.

Diseño RNA

INPUTS: 100*100 = 5000 pixels de entrada
OUTPUTS: 3 salidas
Número de Capas: 4 Capas

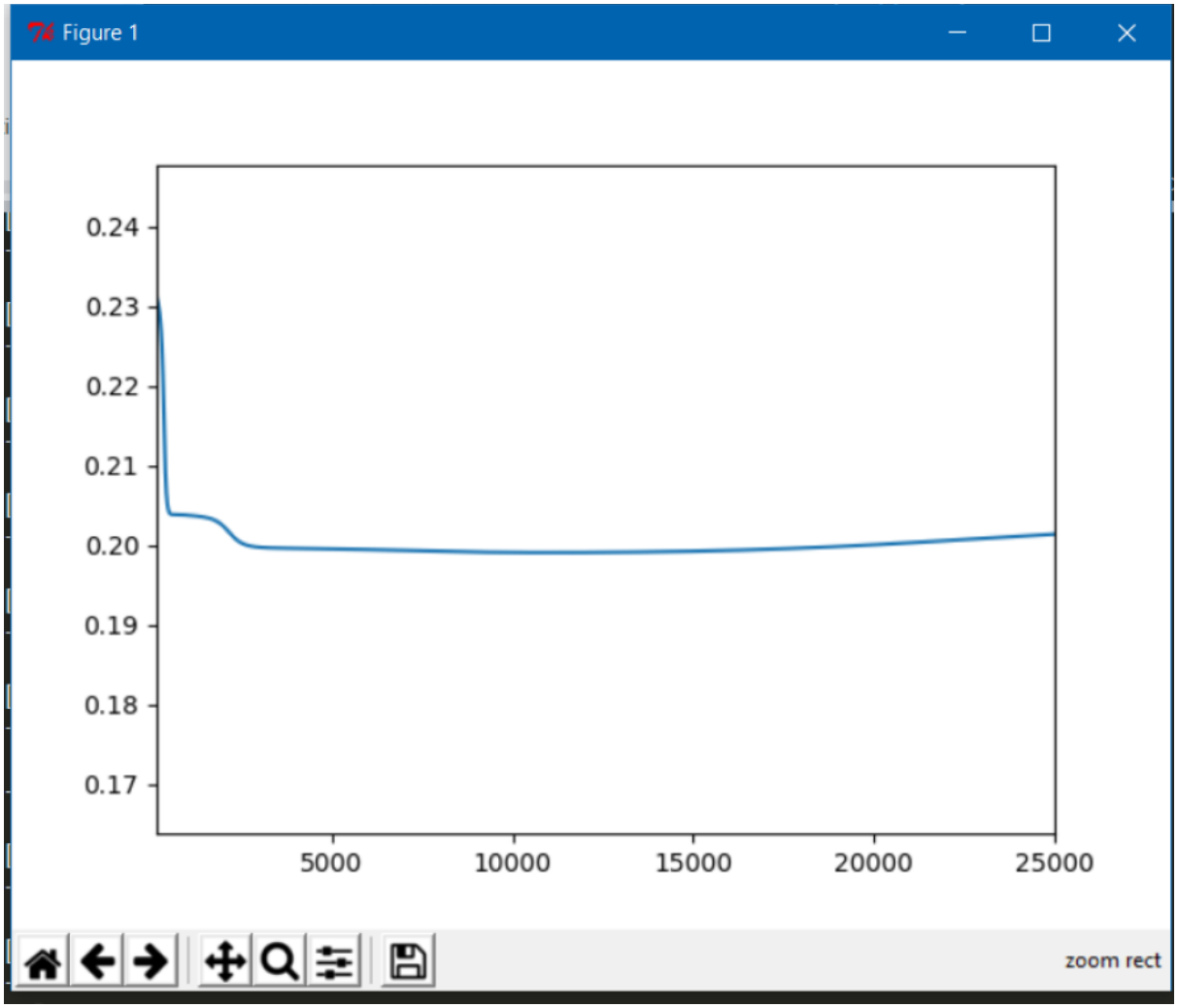


Nota: En la primera capa no se pensó tener el mismo número de neuronas que de entradas en la primera capa debido a la dificultad de cálculo. Es por eso que solo 5 neuronas en la primera capa. También se pensaron en 4 capas ya que en los ejercicios hechos en clase con solo tres capas no alcanzaba a entrenar y se hacía largo el proceso.

Resultados Primera Prueba

Se trato de entrenar usando un error mínimo de 0.2 y una constante de aprendizaje de 0. Se le dió un margen de aproximadamente una hora, sin embargo luego de esa hora al ir por 70,000 épocas se procedió a abortar.

Se Corrió con 25,000 iteraciones, guardando los errores de cada época y se procedió a graficar dichos errores, el resultado fue el siguiente:



Al ver que el error tendía a subir, se procedió a usar una librería para el diseño de la red neuronal, NeuroLab.

Segunda Prueba

Según la primera prueba fallida, se intentó reducir el área de trabajo y por lo tanto las entradas para la RNA que detectará la madurez del tomate.

Según lo que se escribió en la descripción del entorno, lo que se busca es agarrar una imagen donde el tomate pueda estar en cualquier parte y luego enmarcar el área sombreada para recortar.

Para esto se utilizará la librería para python OpenCV. Este proceso involucra varias fases que nos permitirán aislar los objetos dentro de una imagen.

El proceso se divide en 5 fases:

1. Convertir la imagen a escala de grises
2. Filtrar la imagen para eliminar el ruido
3. Aplicar el detector de bordes Canny
4. Buscar los contornos dentro de los bordes detectados
5. Dibujar dichos contornos

Cada una de estas fases dará como resultado una imagen que será la entrada de la siguiente fase. Todo el proceso se hizo utilizando como herramientas **OpenCV y Python.**

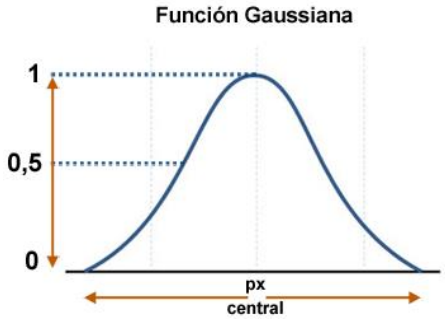
Convertir la imagen en escala de grises con OpenCV

Una vez que has cargado la imagen, lo primero que tienes que hacer es convertir a escala de grises con OpenCV, ya que sino se realiza de esta manera la carga al computador es demasiada.

Primero se intentó realizar sin convertir la imagen a escala de grises. Pero no se consiguió nada, debido a que en sí lo que se estaba haciendo es manejar tres imágenes diferentes R,G,B. Por lo que al ver la carga del CPU se procedió a cancelar ese proceso y utilizar escala de gris.

Filtrado del ruido en una imagen

Las **imágenes digitales no son perfectas**. El ruido inherente de los propios dispositivos o los efectos contraproducentes por iluminación alteran la realidad. En el tratamiento digital de imágenes hay diferentes métodos para eliminar el ruido (promediado, mediana, Gaussiano o bilateral). La que se usó es el método Gaussiano.



Detector de bordes Canny

El proceso para detectar bordes con Canny se divide en 3 pasos.

- Detección de bordes con Sobel
- Supresión de píxeles fuera del borde
- Aplicar umbral por histéresis

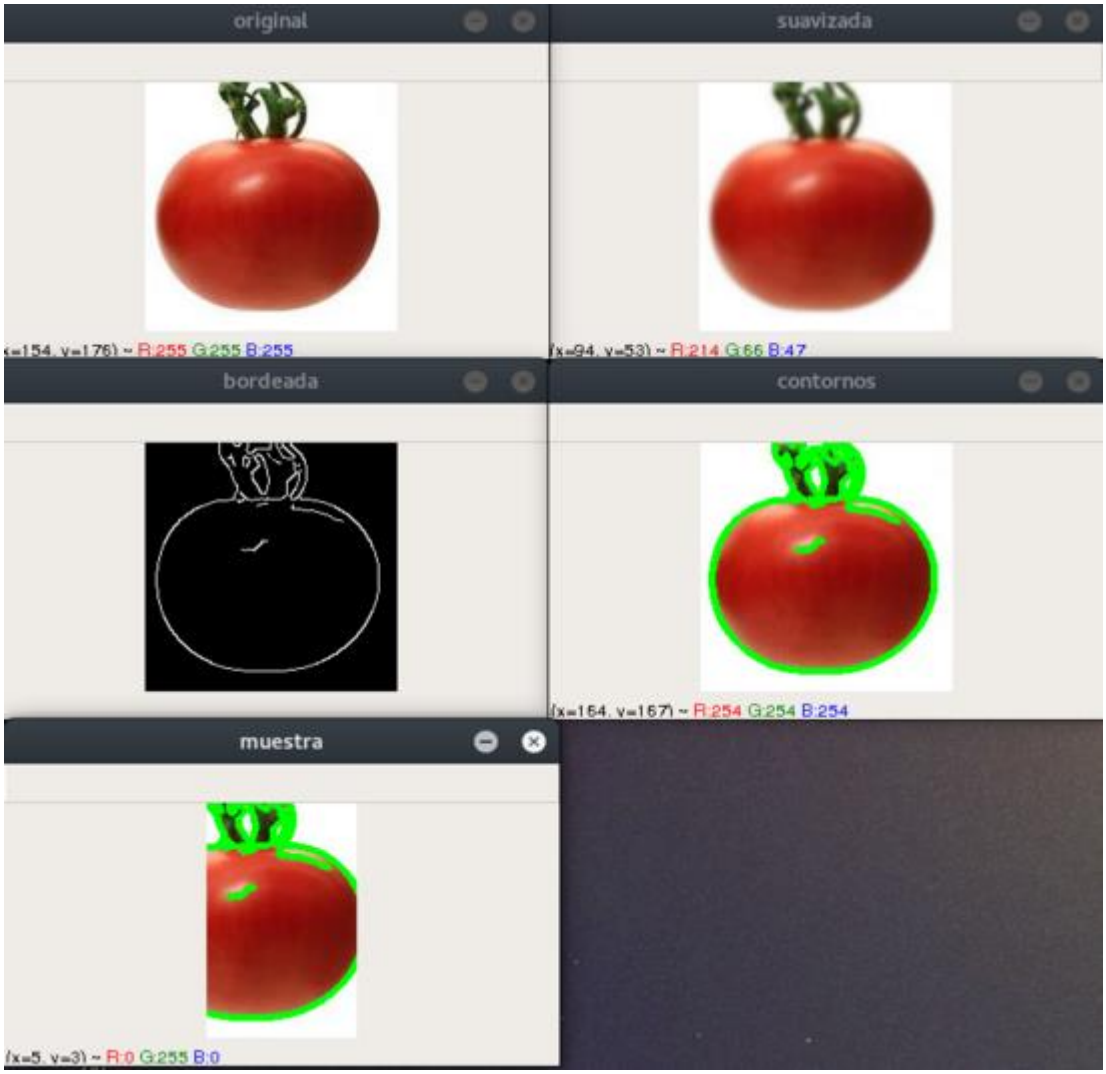
Buscar los contornos de una imagen

Hay que saber la diferencia que existe entre un borde y un contorno. Los bordes, como hemos visto anteriormente, son cambios de intensidad pronunciados. Sin embargo, un contorno es una curva de puntos sin huecos ni saltos es decir, tiene un principio y el final de la curva termina en ese principio.

Dibujar los contornos en una imagen con OpenCV

Por último nos queda dibujar los contornos que hemos encontrado en la imagen. Esto lo haremos a través de otro método de OpenCV.

Los resultados son los siguientes:



A Partir de ahora, solo quedaría recortar la imagen del tomate, eliminando cualquier entrada innecesario para la RNA.

Una vez obtenida la imagen solo del tomate, lo que se buscará es obtener alrededor de 3 a 5 muestras del tomate, esas muestras serán aproximadamente imagenes de 10 píxeles por 10 píxeles.

Para más información acerca del proceso de esta segunda prueba puede visitar el repositorio de Github, donde actualmente están los archivos y codigo para encontrar el contorno del tomate.

https://github.com/jomabaso/IA_tomate

Tercera Prueba

Tratamiento de Imagen OpenCV

Se creo una funcion para encontrar el tomate como tal, a partir del reconocimiento de colores, almacenando los colores que deseamos filtrar en un array de numpy ejemplo:
valor en rgb del color que deseamos.

```
Rojo_debil = np.array([49,50,50])
Rojo_fuerte = np.array([80,255,255])
```

También debemos de convertir la imagen que recibimos del tomate a una forma más reconocible que se explicara en siguiente párrafo, la manera de convertirlo es la siguiente

```
conversion = cv2.cvtColor(imagen, cv2.COLOR_BGR2HSV)
```

Se debe estudiar la imagen para determinar qué píxeles dentro de la imagen están dentro de ese rango, para ello se crea el concepto de máscara, que como tal es un rango de colores.
debemos de crear una mascara, ejemplo:

```
Mascara_roja = cv2.inRange(hsv, Rojo_debil, Rojo_fuerte)
```

donde hsv será el tipo o formato de colores que utilizaremos que significa hue saturation value, lo que representa es el ‘hue’ que es el tono del color, Saturation se refiere a la saturación o la intensidad de la tonalidad que se está analizando, mientras menos saturación más gris, y value que es la luminosidad del color, entonces combinaremos negro y el rango de rojos.

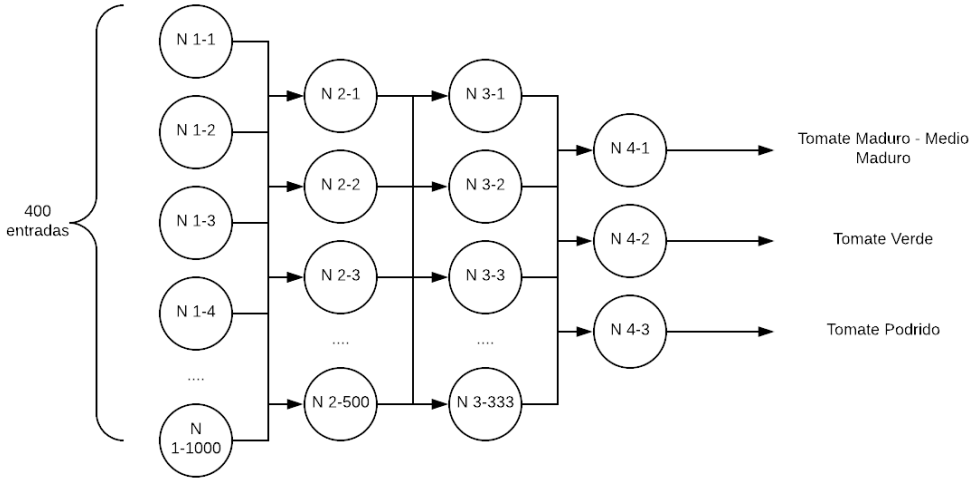
ya que encontramos todos los colores que están dentro de los rangos establecidos
se contornea con la función de cv2.FINDCONTOURS()

A partir de este contorno creamos una elipse para comparar si tiene la forma de un tomate,(considerando los tomates que encontramos normalmente en el mercado) teniendo la elipse determinamos la posición en la que se encuentra en la imagen y con esa posición generamos un cuadro dentro del área de la elipse y recortamos ese cuadro a manera de tener una muestra representativa de los colores del tomate, al recorte resultante se le aplica suavizado a manera de eliminar el brillo innecesario y normalización para determinar de una forma más precisa el valor del píxel que se analiza, luego todos los valores son almacenados en una matriz para su posterior análisis, lo que representa cada dato es el valor del color del píxel y todo este conjunto de datos será la entrada para la red neuronal.

Diseño RNA

Con respecto a la RNA que detectará el nivel de madurez se ha hecho un cambio en el diseño de la Red. Cabe resaltar que la imagen en este punto ya será pasada como una muestra significativa que ayude a determinar la calidad y madurez del tomate.

INPUTS: Imagen de 50*20 pixels.
OUTPUTS: 3 Salidas.



Entrenamiento de la IA

Antes que los datos, imágenes pasen por la RNA descrita anteriormente, se debe de reconocer la imagen del tomate para luego sacar muestras de dichas imágenes. Esto se realiza con OpenCV, y se utilizará la función antes descrita en el tratamiento de imágenes.

Las imágenes que servirán para el entrenamiento se almacenarán en una carpeta llamada “ImagenesTomates”, dentro de esta carpeta se encontrarán tres carpetas donde se clasificaron los diferentes tipos de tomate, al final las imágenes se almacenarán en las siguientes direcciones:

\Proyecto Tomate\ImagenesTomates\TomateMaduro
\Proyecto Tomate\ImagenesTomates\TomatePodrido
\Proyecto Tomate\ImagenesTomates\TomateVerde

Hay un archivo que se llamá tramamientoImagenes.py, que lo que hará es sacar las imagenes de las direcciones descritas anteriormente y las almacenará en las siguientes direcciones, según el tipo de tomate:

\Proyecto Tomate\Recortes\TomateMaduro
\Proyecto Tomate\Recortes\TomatePodrido
\Proyecto Tomate\Recortes\TomateVerde

En las direcciones descritas anteriormente es donde se almacenarán las muestras ya suavizadas y recortadas. Apartir de aca entra el archivo para entrenar entrenamientoNeurolab.py.

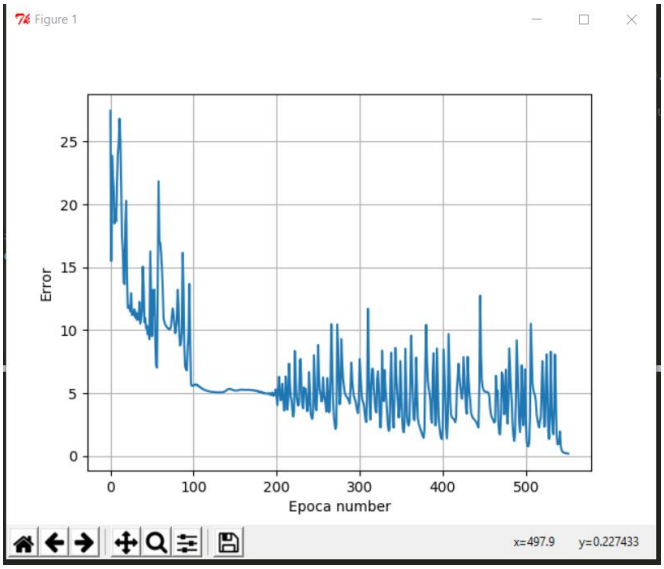
Este archivo de python buscará todas las imágenes ya muestreadas, para luego obtener sus pixeles y normalizar dichos pixeles. Luego ya entra en función la Red Neuronal, con la forma descrita anteriormente.

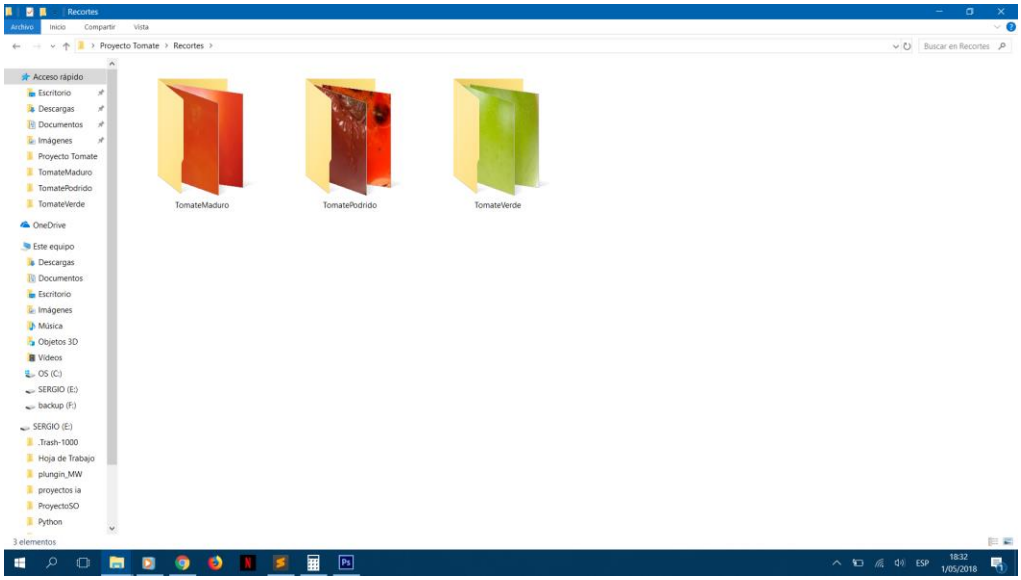
Se generará un archivo cuando la red ya esté entrenada, este archivo tendrá como nombre: “redEntrenada.net".

El archivo de entrenamiento se utilizará en la interfaz gráfica, que será descrita más adelante.

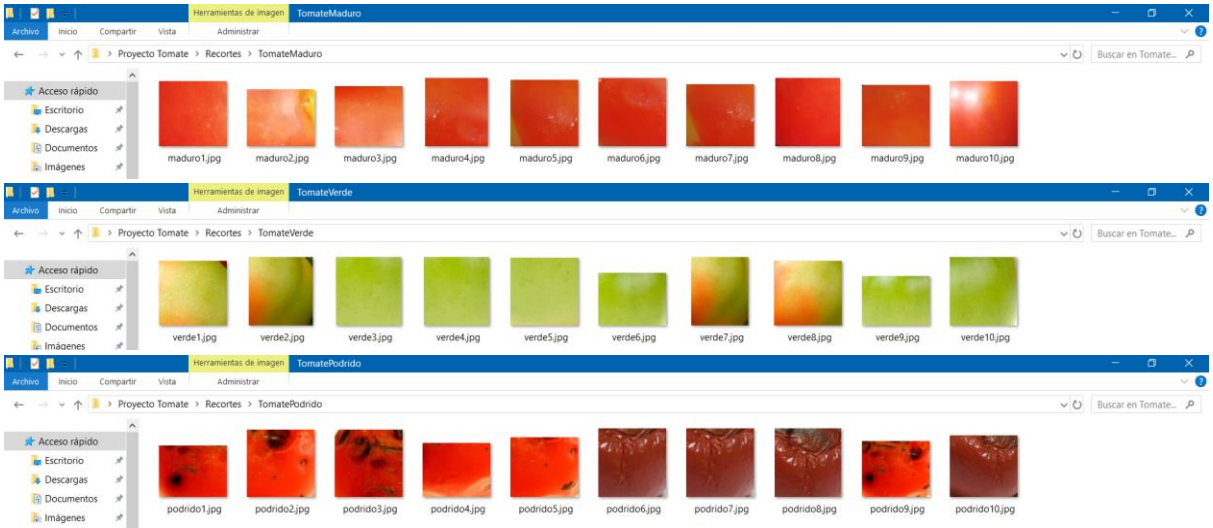
Resultados:

La siguiente gráfica muestra el resultado del error para 5000 iteraciones, con un error mínimo de 0.2 y una constante de aprendizaje de 0.1. Para esta red neuronal se utilizaron fotos de tres tipos de tomates diferentes. Para cada tipo de tomate se utilizaron 10 recortes de 10 imagenes de tomates diferentes.





Donde estas imagenes están separadas en carpetas con el nombre, para que el programa de entrenamiento recorra todas las imágenes que acá se encuentran.



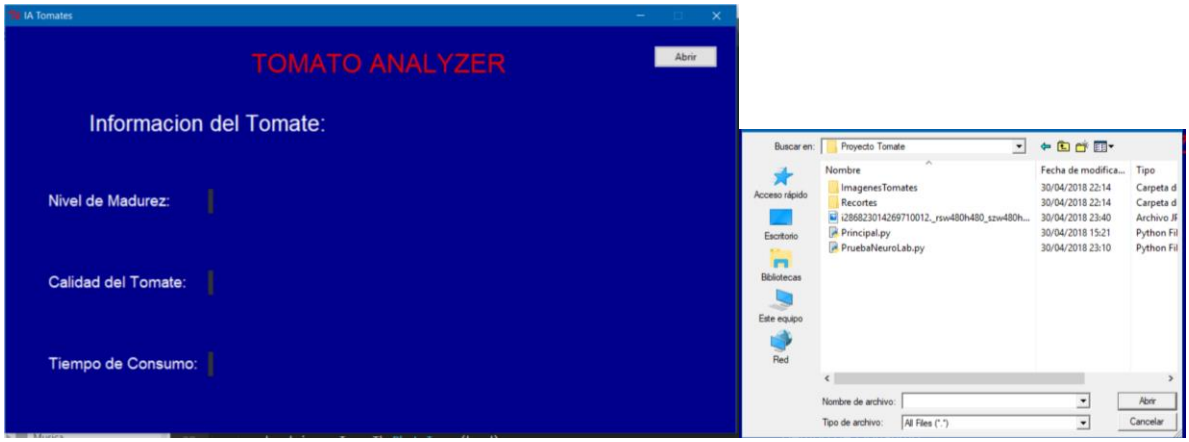
Aca se muestra las salidas del entrenamiento de la RNA.

```
Epoch: 908; Error: 0.454555504241;
Epoch: 909; Error: 0.418291577438;
Epoch: 910; Error: 0.41512112966;
Epoch: 911; Error: 0.382845593052;
Epoch: 912; Error: 0.378086169901;
Epoch: 913; Error: 0.350070539103;
Epoch: 914; Error: 0.344769840913;
Epoch: 915; Error: 0.320757623062;
Epoch: 916; Error: 0.31550561993;
Epoch: 917; Error: 0.295028917185;
Epoch: 918; Error: 0.2901311044;
Epoch: 919; Error: 0.27267697642;
Epoch: 920; Error: 0.26827276955;
Epoch: 921; Error: 0.253359485476;
Epoch: 922; Error: 0.249495870258;
Epoch: 923; Error: 0.236701475249;
Epoch: 924; Error: 0.233376613221;
Epoch: 925; Error: 0.22234445837;
Epoch: 926; Error: 0.219532718362;
Epoch: 927; Error: 0.209966975778;
Epoch: 928; Error: 0.207632934838;
Epoch: 929; Error: 0.199290539336;
The goal of learning is reached
[[-0.01597461  0.91085426 -0.06645995]
 [-0.03409984  0.90899736 -0.06945021]
 [-0.03469288  0.90215962 -0.06827703]
 [-0.01150947  0.90724373 -0.06903845]
```

Interfaz Gráfica de Usuario

Aunque aún haya cosas por pulir en lo que se refiere al tratamiento de imágenes (Detección del tomate, máscaras y normalización de entradas) y al diseño de la RNA y el cómo se tratarán los errores, al ser el último reporte a entregar se quiso mostrar el cómo será la interfaz con la cual el usuario va interactuar.

Donde se mostrarán los resultados del análisis del tomate según la madurez, calidad y el tiempo de consumo. Tenemos el Botón de Abrir en la esquina superior derecha donde al dar clic se le abrirá una ventana para buscar la imagen del tomate que se desea analizar.



Luego de seleccionar dicha imagen y dar clic en abrir, el analizador (IA) actuará y sacará la información que será interpretada y se mostrará en pantalla, junto con la imagen del tomate que se escogió.

Entrega Final

Tratamiento de Imágenes

Para poder obtener una muestra que sea representativa para para la red neuronal, se procedió a encontrar el centro de la figura del tomate.

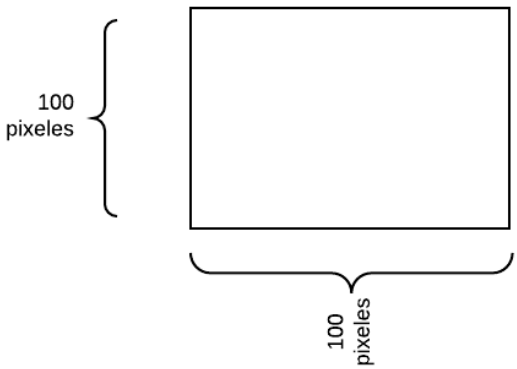
Según lo que se escribió en la descripción del entorno, lo que se busca es agarrar una imagen donde el tomate pueda estar en cualquier parte y luego enmarcar el área sombreada para recortar.

Se aplicó una máscara que luego de encontrar el contorno del tomate, encontró el área del tomate. Una vez a través de esta máscara se procedió a encontrar el centro del objeto, que sería el centro de la máscara que contiene el área del tomate.

Para poder encontrar el centro de la imagen se utilizó la función moments, que ayuda a calcular algunas características como el centro de masa del objeto, el área del objeto, etc.

cv2.moments()

Se obtiene el centro en base a coordenadas en pixeles de la imagen original, a partir de ahora se procedió a recortar. Del centro se le restó 50 píxeles para encontrar la posición a la izquierda, y se le sumó 50 píxeles para encontrar la posición de la derecha. Lo mismo para encontrar la posición superior e inferior. De esta manera la muestra es una imagen cuadrada de 100*100.



Para poder sacar las muestras de las imagenes, estan deben estar guardadas en tres carpetas con el nombre de TomateMaduro, TomateVerde, TomatePodrido, en la carpeta con nombre ImagenesTomates. Luego se ejecuta el archivo:

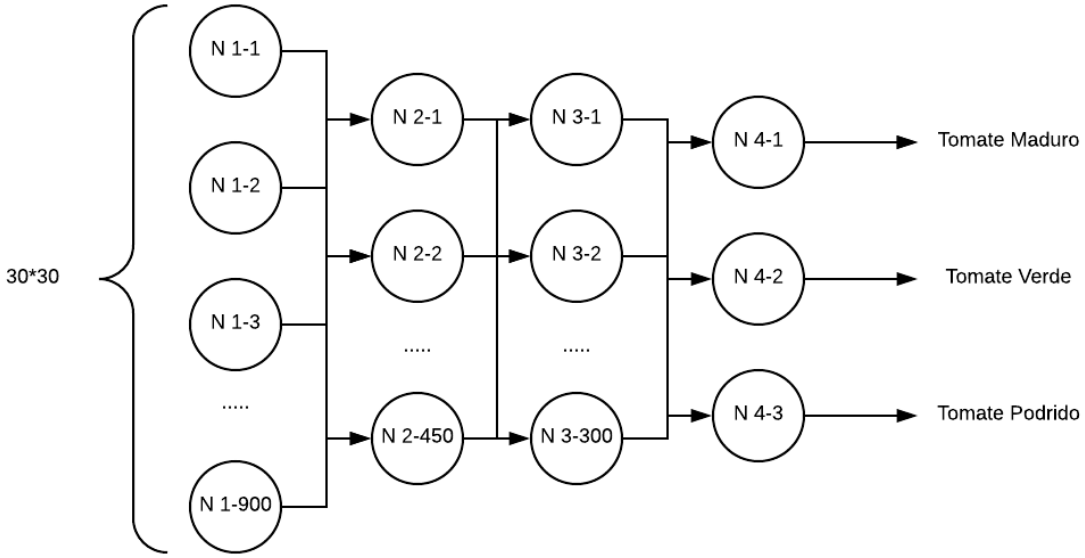
python recortes_entreno.py

Se leera todas las imagenes que esten en estas tres carpetas y las guardara en la carpeta con nombre -RecortesTomates- en tres carpetas con nombre TomateMaduro, TomateVerde, TomatePodrido.

Estas imagenes ya recortadas serán la entrada para la RNA.

Diseño RNA

Los errores se considerarán los errores como cualquier combinación diferente a las especificadas en el análisis de salida de la red neuronal.
INPUTS: Imagen de 30*30 pixels.
OUTPUTS: 3 Salidas.



Para entrenar la RNA se necesitará abrir el archivo - entrenamientoNeuronal.py.

Se tomarán las imágenes de muestra que se encuentran en la carpeta RecortesTomates, para luego redimensionarlas a un tamaño de 30*30, luego se normaliza y se guardará en un archivo csv, junto con las salidas que tendrán en cada patrón, que sería una imagen.

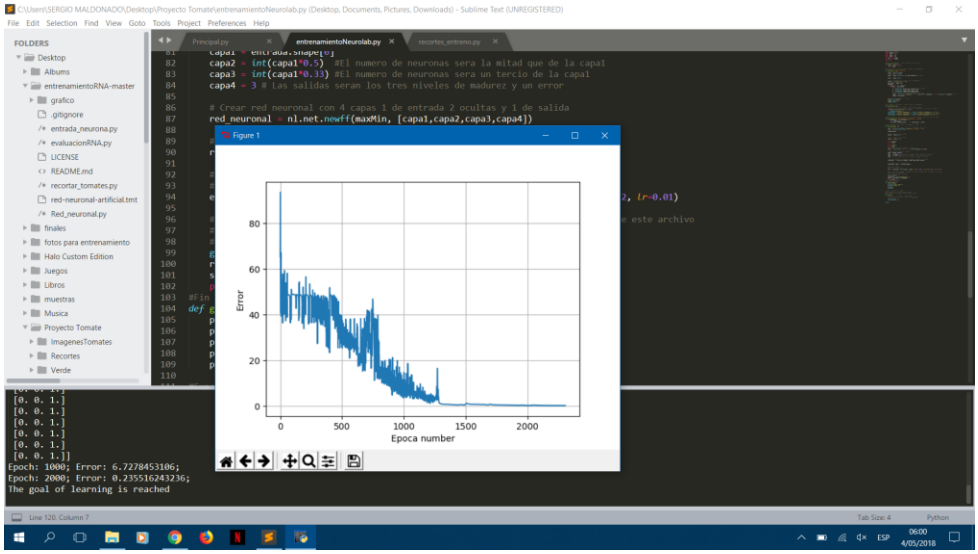
Una vez creado ese archivo, pasará a la creación de la RNA utilizando el tipo de red de NeuroLab Perceptrón Multicapa. Una vez eso, se cargará el archivo csv creado anteriormente y se empezará a entrenar.

Datos del entrenamiento realizado

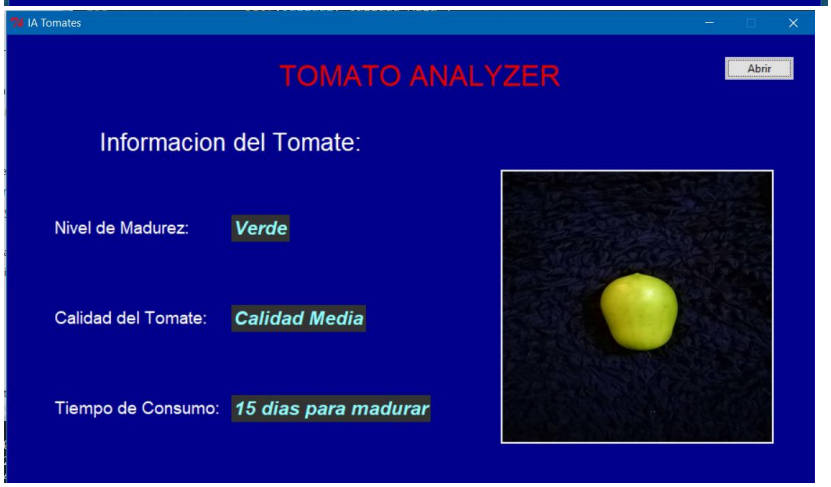
- La RNA se entrenó con 60 imágenes diferentes, correspondientes a los tres tipos de tomates.
- Se utilizó una constante de aprendizaje de 0.1
- Se le dio un margen de 750,000 épocas

Resultados del entrenamiento

La Red fue entrenada con 60 imágenes diferentes de tomates, todas con una dimensión de 30*30 pixeles, al final la RNA se demoró aproximadamente 2600 épocas para entrenar.



Pruebas Finales



Link Repositorio en Github

https://github.com/jomabaso/IA_tomate