Universidad Rafael Landívar.
Facultad de Ingeniería.
Inteligencia Artificial.
Ing. Dhaby Eugenio Xiloj Curruchiche.

# Informe final:
# Detección de patrones por medio de Redes neuronales.

Tzul Pérez, Hugo Adolfo (15222-15)
Xicará Xicará, Wilson Giovanni (15146-15)

Quetzaltenango, 4 de Mayo de 2018.

# CHILE PIMIENTO.

El pimiento es originario de la zona de Bolivia y Perú, donde además de Capsicum frutescens L. se cultivaban al menos otras cuatro especies. Fue traído al Viejo Mundo por Colón en su primer viaje (1493). En el siglo XVI ya se había difundido su cultivo en España, desde donde se distribuyó al resto de Europa y del mundo con la colaboración de los portugueses. Su introducción en Europa supuso un avance culinario, ya que vino a complementar e incluso sustituir a otro condimento muy empleado como era la pimienta negra (Piper nigrum L.), de gran importancia comercial entre Oriente y Occidente (Infoagro 2003). El chile es una Solanácea con seis especies principales y diez especies secundarias. Es una planta anual, herbácea, de crecimiento determinado. Su raíz es pivotante con numerosas raíces adventicias, alcanzando una profundidad de 70-120 cm. La altura de las plantas varía de 0.30 a 1m, según las variedades. La flor del chile es frágil. El fruto es una baya generalmente amarilla o roja en su madurez. Las semillas son aplastadas y lisas, pudiendo contarse de 150-200 por gramo; ricas en aceite y conservan su poder germinativo durante tres o cuatro años.

En la actualidad los nombres de los chiles son bastante confusos ya que con frecuencia el mismo chile recibe otro nombre en un lugar diferente. Además, existe una gran variedad de chiles en el mercado. El chile pimiento elegido para nuestro proyecto recibe el nombre de *Capsicum annuum* y se puede visualizar en la imagen F1.

**Imagen F1.** Apariencia del *capsicum annuum*, lo que comúnmente conocemos como ‘chile pimiento’.

En Guatemala, el chile pimiento tiene ciertas épocas de siembra por tradición y condiciones ambientales que se encuentran entre octubre a enero. Sin embargo, muchos productores han realizado siembras en otras fechas, lo que ha provocado alto índice de plagas y pérdidas económicas.

**Atributos de calidad.**

Una publicación del INE en el 2006 muestra los parámetros que determinan la calidad del chile pimiento en los diferentes puntos de venta de Guatemala. Dicha información es la que se muestra en la imagen F2.

Los frutos deben ser de epidermis lisa, cerosa y de color uniforme. Al cortarlos deben presentar un pericarpio de paredes gruesas y crocantes, con semillas adheridas a la placenta central.

| Denominación en el Mercado | Sección Transversal (cm) | Sección Longitudinal (cm) | Peso (gr) | Vida de Anaquel (días) |
|---|---|---|---|---|
| **Primera** | 3 – 4 | 8 - 10 | 35 - 50 | INICIAL (Verde) 3 - 4 días; INTERMEDIA (Verde-Rojizo) 2 - 3 días; AVANZADA (Rojo-Amarillo) 1 - 2 días |
| **Segunda** | 2 – 3 | 6 - 8 | 30 - 35 | |
| **Tercera (Mezclado)** | --- | --- | --- | |
| Presentación en el Mercado | Mayorista | | | Caja 36 libras,Quintal |
| | Consumidor | | | Libra |

**Imagen F1.** Parámetros que determinan la calidad del chile pimiento.

## Cadena de producción del chile pimiento.

La cadena de producción de este fruto es como se describe en la siguiente imagen:

- **Mapa de la cadena agroproductiva del Chile Pimiento**

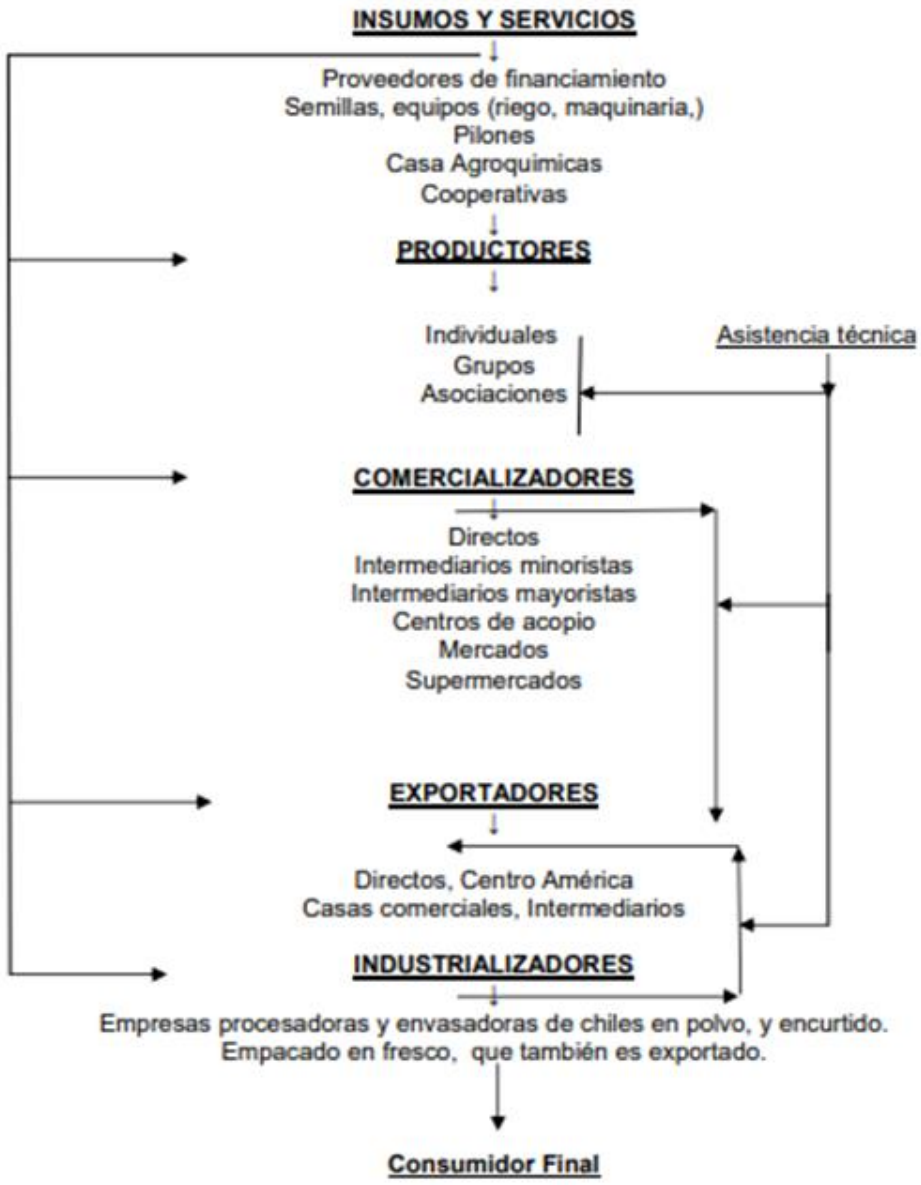



**Figura 5. Cadena Agroproductiva del chile pimiento**

**Proceso de investigación.**

Para recopilar información acerca de la calidad y el grado de madurez del fruto realizamos varias consultas algunas vendedoras del mercado “La Democracia” y las conclusiones fueron las siguientes:

- El tiempo de vida del fruto es de aproximadamente una semana. Esto desde que es verde (aún no maduro), pasando por el proceso de maduración hasta el proceso de descomposición. Durante este tiempo se pueden identificar 3 etapas importantes, distribuidos de la siguiente forma:
    - 1 a 3 días, Etapa de maduración. El color del fruto contiene tonalidades verdes. Conforme avanza hacia el proceso de maduración, disminuye el color verde y es reemplazado por colores rojos.
    - 4 a 5 días. Punto óptimo. Se caracteriza por ser completamente rojo. En esta etapa el fruto es apto para su consumo.
    - 6 días en adelante. Etapa de descomposición. El fruto tiende a tener una tonalidad amarilla o rosada, formándose en su superficie grietas de resequedad, y se forman algunos puntos de color negro. No es apto para el consumo pues puede ser dañino para la salud.
- Los atributos de calidad del fruto pueden ser determinados a través del contacto. Entre ellos, la firmeza, el diámetro, entre otros, por lo que no pueden ser determinados con solo verlos. En la mayoría de casos, se considera un fruto de buena calidad siempre que esté en su punto óptimo, posea un fuerte color rojo, superficie lisa y brillante.

# ESTRUCTURA DE LA RNA.

## Ambientes de la IA.

**Accesible:** Ya que toda la información está presente en la imagen.
**No determinista:** Ya que no le importa que es lo que sucede ha sucedido con anterioridad.
**Episódico:** Ya que se necesita delimitar el fruto de la imagen, esto quiere decir poder identificar el fruto a evaluar y no tomar en cuenta otras cosas que se encuentran a su alrededor. Y luego evaluar el aspecto del fruto por medio de los colores que posee.
**Estático:** Ya que no tendrá otro ente que afecte sus decisiones.
**Discreto:** Ya que la cantidad de salidas es limitada pues solamente identificamos 5 estados los cuales son: 1) no maduro = tonalidades verdes; 2) maduro = tonalidades rojos; 3) pasado de maduro = tonalidades cafés; 4) descompuesto = tonalidades negro, café, blanco; 5) no se identificó el fruto.

## Restricciones generales.

Dentro de la imagen a evaluar no debe haber objetos con colores similares a los que posee el fruto. Estos son: rojo, verde, naranja, amarillo, y algunas tonalidades rosadas y grises claras.

## Consideraciones generales.

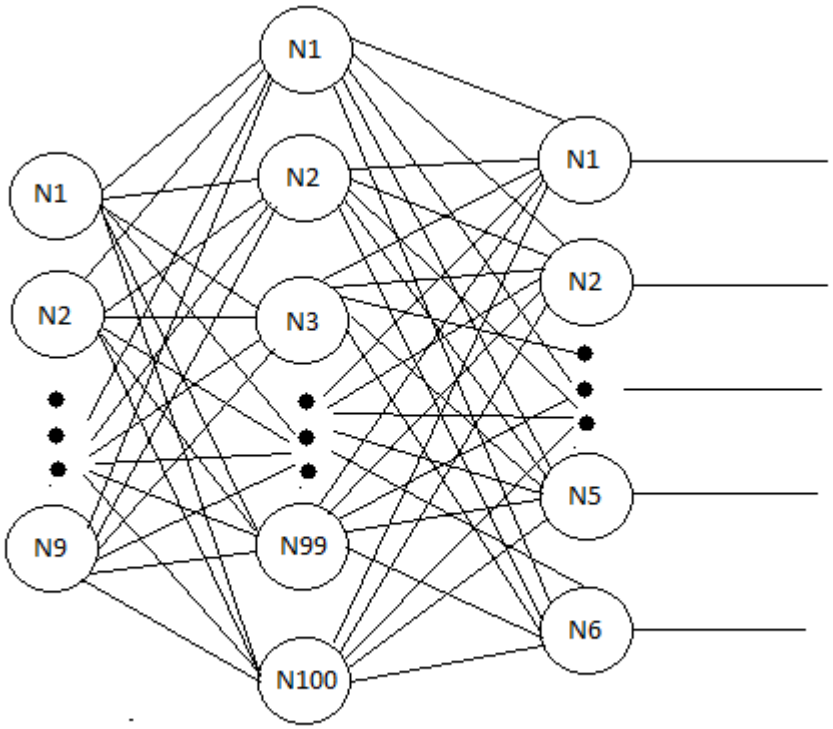
Para obtener mejores resultados se recomienda que no haya otros objetos cercanos al fruto a evaluar.
Tratar de que toda la imagen se enfoque en el fruto a evaluar.

## Episodio 1: detección del fruto a evaluar.

**Entrada.**
Puede ser una imagen de cualquier tamaño.
En este episodio no se utiliza una red neuronal. En su lugar, se intenta determinar el fruto en cuestión en base a procesamiento previo de la imagen, para mostrar al usuario el fruto. En este episodio la aplicación hace el intento de ubicar el fruto a evaluar en caso de que no esté bien identificado en la imagen de entrada. Puede que no se haga identificación de manera adecuada, por lo que se le da al usuario la opción de identificar el fruto de forma manual.

**Salida.**
La imagen con el fruto a evaluar (enfocado lo mayor posible). Esta nueva imagen puede variar de tamaño, pero por estándar el lado mayor es de 300 pixeles. Esta imagen ya puede ser evaluada por la red neuronal del siguiente episodio.

**Episodio 2: detección de madurez.**
**Diseño**

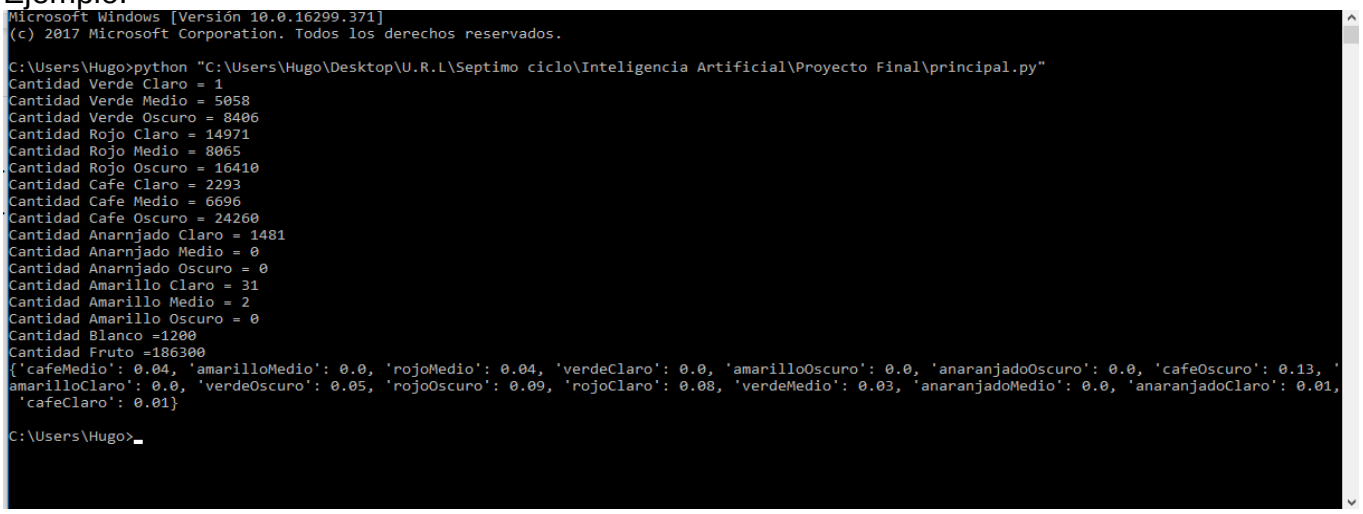



**Entrada.**
La entrada de la RNA es la entrada de la imagen proveniente del episodio anterior (que tiene un tamaño estándar con el lado mayor de 300 pixeles).
La RNA tiene **9 entradas**, las cuales representan un porcentaje de un rango de colores que se tiene dentro de cada pixel de la imagen.
Cada pixel cuenta con un código RGB el cual nos describe qué color contiene, y cada combinación de colores tiene un nombre en específico, tomaremos los nombres como ventaja y estableceremos los rangos de colores a partir de ellos. Los rangos estarán establecidos de la siguiente forma:

```
rojoClaro = ['indianred', 'lightcoral', 'ligthsalmon', 'salmon',
   'tomato', 'coral', 'crimson']
rojoMedio = ['red', 'orangered']
rojoOscuro = ['darkred', 'maroon', 'firebrick', 'brown']
verdeClaro = ['greenyellow', 'lawngreen', 'limegreen', 'lime']
verdeMedio = ['olive', 'olivedrab', 'yellowgreen']
verdeOscuro = ['darkgreen', 'green', 'forestgreen', 'darkolivegreen',
   'darkseagreen', 'mediumseagreen', 'lightgreen', 'seagreen',
   'palegreen', 'sapgreen']
anaranjado = ['darkorange', 'orange', 'sandybrown', 'cadmiumorange',
   'cadmiumyellow', 'carrot']
amarillo = ['yellow', 'gold', 'goldenrod', 'banana']
descompuesto = ['rosybrown', 'lightpink', 'pink', 'plum', 'mistyrose']
```

Para crear las entradas de la RNA se evaluará cada pixel de la imagen, llevando un conteo de la cantidad de pixeles pertenecientes a cada rango mencionado anteriormente. A continuación, se suman todos estos conteos para obtener la cantidad total de pixeles que posee el fruto. Por último, se determina el porcentaje que representa cada rango de colores, de la cantidad total de pixeles que ocupa el fruto.

Estos valores estarán en un rango entre 0 y 1, con dos decimales.
Ejemplo:




**Salida:**
De la RNA se obtendrán **6 salidas**, las cuales tendrán el valor de 0 o 1. Cada salida representa una de las etapas en las que se encuentra el fruto. Los valores de salida y su interpretación son las siguientes:

1. Fruto inmaduro (característico por ser completamente verde).
2. Fruto en proceso de maduración.
3. Estado óptimo. Se considera el mejor momento para su consumo.
4. Estado leve de descomposición.
5. Fruto descompuesto. No se recomienda su consumo.
6. Indicador de que la imagen evaluada no corresponde a un chile pimiento.

Ejemplo:
Salida: `[0,0,0,1,0,0]` = El chile pimiento lleva aproximadamente 6 días, y empieza a descomponerse.

## ALGORITMO DE ENTRENAMIENTO.

**Algoritmo BFGS.**
Algoritmo Broyden-Fletcher-Goldfarb-Shanno. Método iterativo para resolver problemas no lineales de optimización no lineal. Utilizado en las técnicas de optimización de segundo orden.
Hace uso tanto del gradiente como de una aproximación a la inversa de la matriz Hessiana (matriz cuadrada de nxn de las segundas derivadas parciales de una función de n variables) de forma iterativa, para hacer una aproximación al cálculo de la segunda derivada, por lo que se dice que es un método quasi-newtoniano (alternativa al método de Newton para encontrar ceros o máximos y mínimos locales de funciones).
Para funciones de muchas variables, el problema de este método es el costo computacional ya que almacena una matriz cuadrada de datos tan grande como el cuadrado de la cantidad de variables; además de que el tiempo necesario para calcular toda la matriz de forma exacta puede ser prohibitivo, por lo que sólo busca una aproximación.

**L-BFGS** (desarrollado por Jorge Nocedal) es capaz de resolver funciones sin restricciones. La variante **L-BFGS-B** (desarrollado por Jorge Nocedal y Richard Byrd) puede resolver funciones con restricciones simples en sus parámetros.

**Consideraciones.**

- BFGS se basa en el historial completo de gradientes.
- L-BFGS se basa solo en los m gradientes más recientes (de ahí que se le asocia a memoria limitada), que generalmente está entre 10 a 20 valores anteriores. Es útil para problemas muy grandes.
- Es más recomendable utilizar BFGS siempre que se puedan cumplir los requisitos de memoria ya que L-BFGS no puede funcionar tan bien como BFGS.

Para el entrenamiento de la RNA iniciamos utilizando el algoritmo Back-Propagation simple. Sin embargo, éste algoritmo no lograba disminuir en lo posible el error obtenido y por consecuente no se obtenía un entrenamiento adecuado.

Es por ello que se utilizó el algoritmo BFGS ya que este proporcionaba un mejor entrenamiento. El único inconveniente es que este algoritmo requiere de un alto costo computacional. Este algoritmo está disponible para Python en la librería **`neurolab`**.

## PROCESO DE DESARROLLO.

A continuación se adjuntan los informes realizados durante la fase de experimentación del proyecto, las consideraciones iniciales, los resultados obtenidos de las pruebas realizas y los cambios realizados mencionados en los informes siguientes.

# INFORME (Parte 1).

Para el desarrollo del proyecto el producto a utilizar será **chile pimiento**.

## Ambientes de la IA.

**Accesible:** Ya que toda la información está presente en la imagen.
**No determinista:** Ya que no le importa que es lo que sucede ha sucedido con anterioridad.
**Episódico:** Ya que se necesita delimitar el fruto de la imagen, esto quiere decir poder identificar el fruto a evaluar y no tomar en cuenta otras cosas que se encuentran a su alrededor. Y luego evaluar el aspecto del fruto por medio de los colores que posee.
**Estático:** Ya que no tendrá otro ente que afecte sus decisiones.
**Discreto:** Ya que la cantidad de salidas es limitada pues solamente identificamos 5 estados los cuales son: 1) no maduro = tonalidades verdes; 2) maduro = tonalidades rojos; 3) pasado de maduro = tonalidades cafés; 4) descompuesto = tonalidades negro, café, blanco; 5) no se identificó el fruto.

## Restricciones generales.

- Tamaño de la imagen = 1000*750 pixeles
- En la imagen puede haber cualquier cantidad de objetos sin embargo solamente un fruto que se desea identificar.

## Estructura de la red neuronal,

**Episodio: Identificar coordenadas del fruto**
**Entrada:**
Se llevará la imagen a evaluar a una escala de 1000*750 pixeles.
Por ahora no se realizará algún tratamiento a la imagen, por lo que cada pixel generará 3 entradas hacia la red neuronal. Esto hace un total de (1000px)*(750px)*(3 valores RGB) = 2250000 + 10 (indicador de pixeles por fila) = 2250010 entradas.

**Salida:**
Se utilizarán 2 coordenadas de un rectángulo que contendrá el fruto a evaluar.
La coordenada x se normalizará en el siguiente rango:
1 - 10px = 1
11 - 20px = 2
….
991 - 1000px = 100
La coordenada y se normalizará en el siguiente rango:
1 - 10px = 1
11 - 20px = 2
….
741 - 750px = 75
En total, la coordenada de un solo punto requerirá de 175 salidas. Para dos puntos se requerirá un total de 350 salidas. Agregando una salida extra del error.
En total, se tendrán 351 salidas.

**Pendiente.**
Convertir la imagen de un formato RGB a HSV. Con este nuevo formato utilizaremos solamente la matiz, ya que esta nos dice el color que se está utilizando. Pasar estas salidas HSV como entradas

de la red neuronal, tomando los tres valores. Esto hace un total de (1000px)*(750px)*(3 valores HSV) = 2250000 + 10 (indicador de pixeles por fila) = 2250010 entradas.
Formato HSV (Hue, Saturation, Value -> Matiz, Saturación, Valor)

- Matiz: valor entre 0 y 360 (en RGB son 3 valores).
- Saturación: que tan fuerte o débil es el color (100% o 0%, respectivamente).
- Valor: que tan claro u oscuro es un color (100% o 0%, respectivamente).

**Resultados.**

**Prueba 1.**

Entrenar con 3 patrones no da buenos resultados.
Utilizando 5 capas: aumente la constante de aprendizaje a 6 y no se obtuvo un error más pequeño.
Al hacer más pequeño epsilon el entrenamiento no llega a un error mínimo.

**Prueba 2.**

Se aumentó la constante de aprendizaje y así se aceleró el tiempo de entrenamiento y disminuyo un poco el error obtenido.

**Prueba 3.**

Se aumentó la constante de aprendizaje. Aceleró el tiempo de entrenamiento pero no disminuyó el error obtenido, más bien lo aumentó.

**Prueba 4.**

Aumentar la constante de aprendizaje y epsilon no nos da el error necesario para determinar en donde se encuentra el fruto.

**Prueba 5: Fallida.**

Se realiza el entrenamiento de la red neuronal con un total de 27 imágenes correspondientes al fruto en cuestión (chile pimiento). Para cada imagen se trató de enfocar el fruto en cuestión. En unas, el fruto aparece sólo sobre un fondo negro, pero en la mayoría aparece junto a otros frutos (siendo la diferencia el enfoque que se le dio a uno de los frutos). Para cada imagen se obtuvo las coordenadas del rectángulo que encierra al fruto y se aplicó la normalización mencionada anteriormente.
No se pudieron comprobar los resultados ya que se superó el límite (MemoryError) en el consumo de memoria RAM.

**Prueba 6.**

Se procedió a utilizar otra librería la cual lleva por nombre **`neurolab`**. No se ha podido entrenar la red neuronal debido a que la cantidad de entradas es muy grande y congela la máquina.

# INFORME (Parte 2).

Posterior a la primera entrega se realizó varias pruebas de entrenamiento.
En muchas ocasiones surgieron problemas de consumo de memoria RAM (MemoryError). Los motivos son varios, pero la mayoría de ellos afirma que este problema surge porque el programa consume más memoria RAM del que el sistema operativo lo permite. En base a ello se realizaron algunos ajustes:

- La imagen a evaluar tendrá un tamaño de 500x375 pixeles
- La librería a utilizar será **`neurolab`**.

Este nuevo ajuste genera los siguientes cambios:

**Entrada:**
La imagen de entrada se convierte a escala de grises, por lo que cada pixel generará 1 entrada hacia la red neuronal. Esto hace un total de (500px)*(375px)*(1 valores en gris) = 652,500 + 9 (indicador de pixeles por fila convertido a binario) = **187,509 entradas**.

**Salida:**
Se utilizarán 2 coordenadas de un rectángulo que contendrá el fruto a evaluar.
La coordenada x se normalizará en el siguiente rango:
1 - 10px = 1
11 - 20px = 2
….
491 - 500px = 50
La coordenada y se normalizará en el siguiente rango:
1 - 10px = 1
11 - 20px = 2
….
371 – 375px = 38
En total, la coordenada de un solo punto requerirá de 88 salidas. Para dos puntos se requerirá un total de 176 salidas. Agregando una salida extra del error.
En total, se tendrán **177 salidas**.

En los días posteriores se realizaron pruebas con estos nuevos ajustes. Durante estas pruebas no obtuvimos resultados satisfactorios pues los valores devueltos como coordenadas eran muy variados a lo que se esperaban desde el inicio.
Después encontramos información de que entre las varias funciones de la librería OpenCV hay algunas que nos permite detectar los bordes de objetos en las imágenes y que también hay una función que permite encerrar en un rectángulo un objeto específico, que después de todo es lo que queremos hacer con el primer episodio.
Las pruebas iniciales devuelven el siguiente resultado:

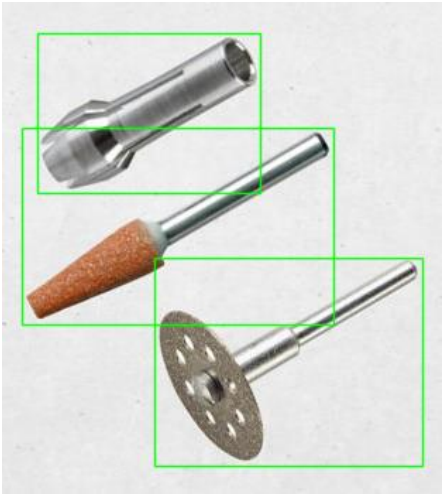


**Imagen N1**. Detección de bordes de una imagen probada con la funcionalidad de OpenCV.

Como se puede observar, se logra encerrar en un rectángulo los objetos encontrados en la imagen. Aplicándolo a nuestras imágenes de prueba se obtuvieron algunos resultados diferentes:

**Imagen N2**. Detección de bordes de una de las imágenes de entrenamiento.

**Imagen N3**. Detección de bordes de una de las imágenes de entrenamiento.

**Imagen N4**. Detección de bordes de una de las imágenes de entrenamiento.

Como se muestra en la imagen N3 e imagen N4 aún hay algunos problemas con obtener el rectángulo pues se obtienen varios que no encierran por completo al fruto en cuestión, a diferencia del resultado obtenido en la imagen N2. Esto se debe a que hay parámetros utilizados en la función que aún no comprendemos completamente.
Los resultados de la imagen N2 hacen que pensemos en restringir la imagen de entrada a que sólo tenga un fruto a evaluar. Con ello se realiza la siguiente prueba:

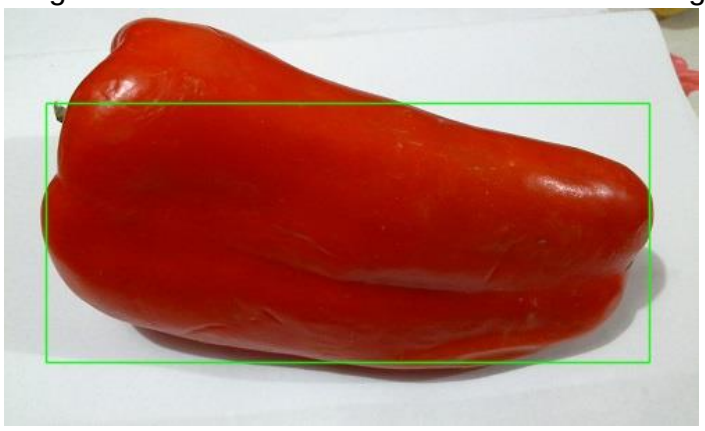


**Imagen N5**. Detección de bordes de una de las imágenes de entrenamiento, restringiendo a que sólo aparece el fruto en cuestión.

Esta restricción proporciona resultados más cercanos a los deseados, por lo que ahora lo tomaremos en cuenta.

**Nuevas consideraciones.**

- Reducir la imagen a un tamaño de 500x375 pixeles, y convertirla a escala de grises.
- La detección del rectángulo que encierra el fruto en cuestión se implementará con las funciones de OpenCV. Con esto asumimos por ahora de que podemos obtener las coordenadas del rectángulo. Y, estas coordenadas son las que utilizaremos en el Episodio 2. Como se observa en la imagen N5 existe la posibilidad de que no logremos tomar en cuenta todo el fruto, por lo que puede suceder que analicemos parte de la imagen del fruto.
- Con la consideración anterior empezaremos a entrenar la red neuronal para el episodio 2.

En nuestro grupo de trabajo dividiremos el trabajo en dos partes:
Uno de los dos se enfocará en mejorar la detección de la imagen y obtener las coordenadas del rectángulo para el episodio 2.
El otro se encargará de seguir intentando detectar mejor las coordenadas del rectángulo a través de la red neuronal del episodio 1.

Comenzamos a desarrollar el episodio 2 y para ello se definirán las entradas y salidas de la siguiente manera:

**Episodio: Identificar Madurez y Calidad del Fruto**
**Entrada:**
Se evaluará una imagen que contendrá solamente el fruto a evaluar, la imagen tendrá una escala de 300*225 pixeles.
Por ahora no se realizará algún tratamiento a la imagen, por lo que cada pixel generará 3 entradas hacia la red neuronal. Esto hace un total de (300px)*(225px)*(3 valores RGB) = 202500 + 9 (indicador de pixeles por fila) = 202509 entradas.

**Salida:**
Se obtendrá por el momento 21 salidas las cuales solo tendrán el valor de 0 o 1. Las primeras 20 nos darán un número del 1 al 20, que tendrán significan el porcentaje de madurez del fruto se interpretará del siguiente significado:
1 = 0-10%
2 = 11-20%
3 = 21-30%
4 = 31-40%
5 = 41-50%
6 = 51-60%
7 = 61-70%
8 = 71-80%
9 = 81-90%
10 = 91-100%
11 = 101-110%
12 = 111-120%
13 = 121-130%
14 = 131-140%
15 = 141-150%
16 = 151-160%
17 = 161-170%
18 = 171-180%
19 = 181-190%
20 = 191-200%
Del 0 a 80% se considera como un fruto en desarrollo, de 81 a 120 un fruto maduro y de 121 a 200 como un fruto pasado de maduro, y la última salida se interpreta como un error.
Ejemplo de salida:
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 = 2 = 11-20% = En desarrollo
0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 = 12 = 111-120% = Maduro
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 = Error

# INFORME (Parte 3).

Debido a que se ha cambiado la forma de normalizar las entradas de la red neuronal en el episodio 2 se harán los siguientes cambios en la fase de reconocimiento del fruto:

- Además de recortar la imagen al rectángulo encontrado (que se pretende encierre la mayor parte del fruto), se tratará de limpiar las partes de la imagen que no corresponden al fruto. Esto se debe a que, como se evaluarán los colores encontrados en la imagen del fruto y se obtendrá un porcentaje para cada color, hay colores que no corresponden al fruto y por lo tanto hay que omitirlos. Para ello, se pretende convertir dichos colores (porciones de imagen que son el fondo de la imagen del fruto) a colores blanco; esto con la finalidad de omitirlo al momento de hacer el conteo de toda la gama de colores posibles del fruto.

Alguno de los resultados son los siguientes:

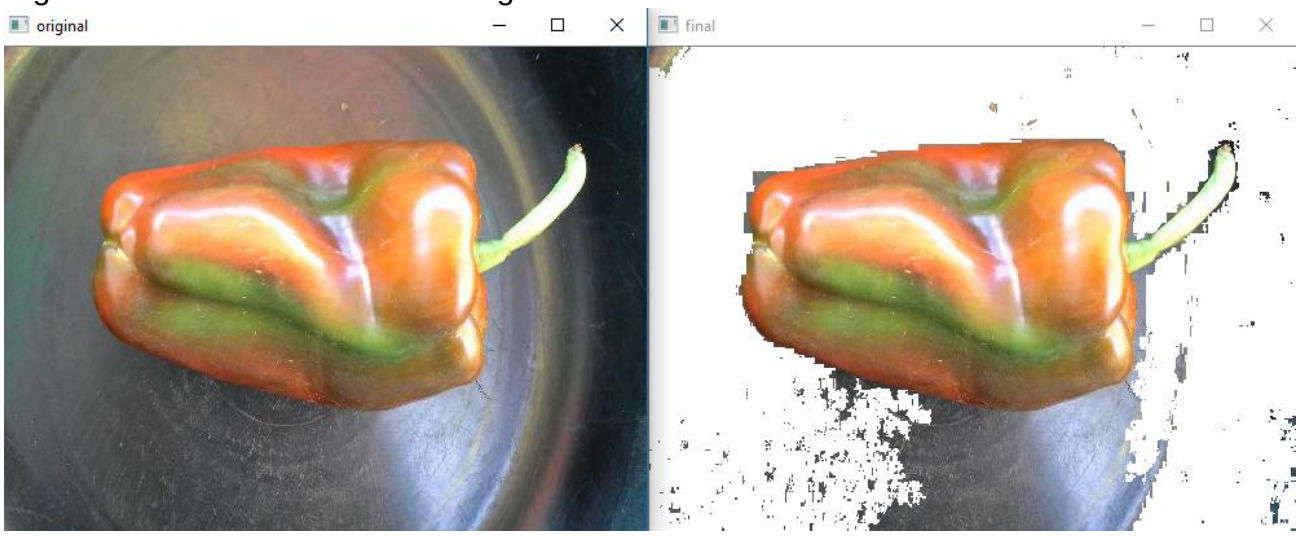



**Imagen N6**. Filtrado de imagen del fruto a evaluar.

En la imagen N6, a la izquierda se observa la imagen original y a la derecha se muestra la imagen procesada (eliminando partes de la imagen que no corresponden al fruto). El problema que se tiene es el contraste entre el fruto y su fondo, y se observa en mayor cantidad en la imagen N7

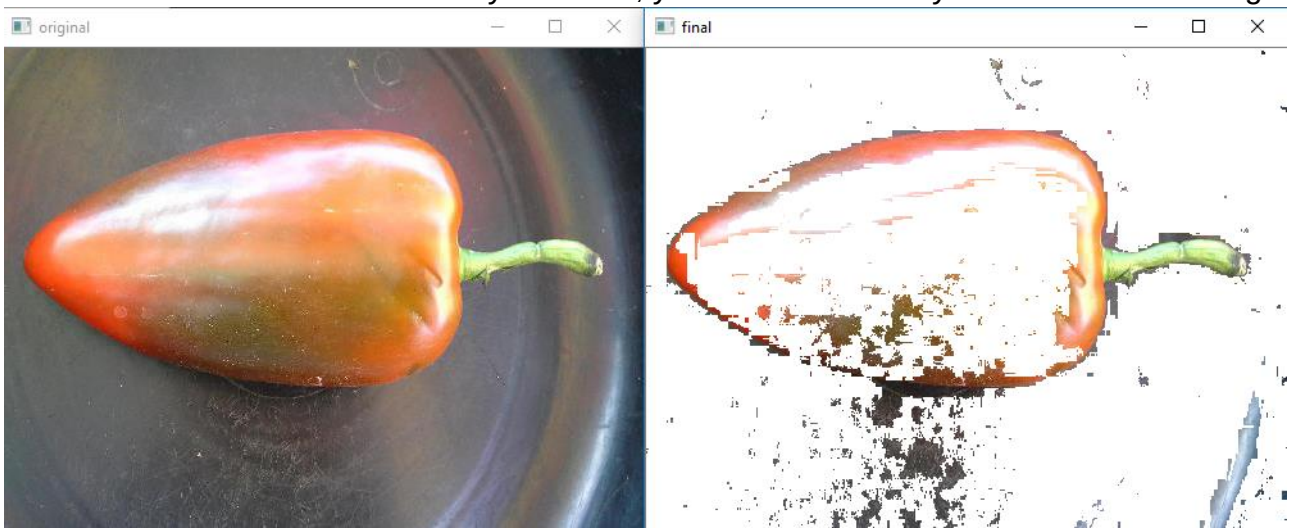



**Imagen N7**. Problema de filtrado de imagen del fruto a evaluar. Aún se está evaluando la causa del problema para intentar corregirlo.

Esto se logra utilizando la función **`floodFill`** de la librería OpenCV. Su funcionamiento se basa en expandir el relleno de pixeles blancos desde un punto inicial, omitiendo contornos cerrados dentro del área de barrido. El problema surge entonces cuando hay partes del fruto en los que no se detecta su borde (mismo contraste del fruto con su fondo) por lo que surge el problema de la imagen N7. Una alternativa para solucionar el problema es utilizar una máscara con la silueta binaria de la imagen original (ceros en donde está el fruto y unos en donde está el fondo) y hacer una intersección de dicha máscara con la imagen original para que con ello se logre conservar el fruto y eliminar el fondo.