Aprendizaje Automático

Proyecto final

## Autores:

## Daniel Vidal Soroa

## Juan Diego Peña

# Introducción:

El objetivo de este trabajo será obtener un mecanismo que permita realizar la clasificación de mangos en tres distintas categorías según su calidad. Para ello, se buscará complementar el trabajo anterior utilizando técnicas de agrupamiento, reducción de dimensionalidad y detección de anomalías.

Los datos se seleccionaron de la plataforma kaggle [3] y cuentan con tres clases de mango y 200 imágenes por clase:

* ***Clase extra:*** Usados generalmente para la exportación, no contienen defectos o son lo suficientemente leves para no afectar su aspecto, calidad o conservación en general.
* ***Clase I:*** Destinados al consumo local, de buena calidad con algunos defectos como quemaduras por el sol, rozaduras, etc.
* ***Clase II:*** Reservados normalmente para el procesamiento industrial, cuentan con los mismos defectos de la Clase I a una escala mayor.

A picture containing fruit

Description automatically generated

Figura 1. Clases de mango disponible

## Clasificación no supervisada

El desempeño de un clasificador depende principalmente de la calidad del set de datos usado durante el entrenamiento. Para ello, el algoritmo requiere que las imágenes utilizadas se encuentren previamente etiquetadas. Sin embargo, esto no es siempre posible y muchas veces se requiere realizar el etiquetado de forma manual, lo cual puede ser engorroso.

Una de las mejoras propuestas será contar con una técnica de clasificación no supervisada. Esto, permitiría etiquetar un conjunto de imágenes vírgenes para su posterior utilización para el entrenamiento de un clasificador. En este caso, se compararán distintos algoritmos de agrupamiento con el objetivo de separar las imágenes en tres clases distintas, y comparando este resultado con la etiqueta real.

Para el agrupamiento, en principio, se utilizarán las 600 imágenes aplanadas. Sin embargo, debido a que estas imágenes son tan similares, se utilizarán además el extractor de características HOG y una red neuronal pre-entrenada. En ambos casos, lo que se busca es que estos algoritmos sean capaces transformar las imágenes originales de manera que pudiera ser más fácil de agrupar. Los resultados obtenidos pueden apreciarse en la Tabla 1.

Tabla : Resultados obtenidos durante la clasificación no supervisada

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Objetivo | Extracción de características | Algoritmo de agrupamiento | Exactitud | Tiempo | multiprocesos |
| Comparar redes pre-entrenadas | inception | KMeans | 0,38 | 145,44 | FALSE |
| vgg | KMeans | 0,6 | 234,82 | FALSE |
| xception | KMeans | 0,29 | 188,29 | FALSE |
| resnet | KMeans | 0,18 | 196,06 | FALSE |
| Comparar VGG con HOG y RGB | vgg | KMeans | 0,6 | 270,4 | FALSE |
| hog | KMeans | 0,36 | 12,33 | FALSE |
| rgb | KMeans | 0,27 | 16,22 | FALSE |
| Comparar distintos extractores con distintos algoritmos | hog | decision\_tree | 0,4 | 54,37 | FALSE |
| rgb | decision\_tree | 0,41 | 45,07 | FALSE |
| vgg | decision\_tree | 0,57 | 273,32 | FALSE |
| vgg | spectral | 0,3 | 202,47 | FALSE |
| hog | spectral | 0,33 | 56,63 | FALSE |
| rgb | spectral | 0,3 | 6,53 | FALSE |
|  | hog | spectral | 0,33 | 41,8 | TRUE |

Como se puede apreciar, en primer lugar, se utilizaron cuatro redes pre-entrenadas disponibles en TensorFlow en combinación con el algoritmo de agrupamiento K-means. En este caso, el mejor resultado se obtuvo con VGG para un 60% de acierto en la clasificación. Sabiendo que esta red era la de mejor desempeño, se pasó a comparar su resultado con el extractor de características de HOG y el conjunto de imágenes vírgenes en formato RGB.

Luego, se compararon otros algoritmos de agrupamiento, combinándolos con los extractores antes utilizados con el objetivo de encontrar la mejor combinación. Finalmente, el mejor desempeño apreciado fue utilizando VGG como extractor y K-means como clasificador (Figura 2)

Chart, treemap chart

Description automatically generated

Figura : Matriz de confusión VGG + K-means (random\_state = 42)

Sin embargo, los resultados obtenidos no son suficientemente buenos como para considerarlos satisfactorios. En este caso las clases son demasiado parecidas, y ninguno de los algoritmos de agrupamientos utilizados logra separarlas adecuadamente. El hecho de que K-means logre distinguir la clase I aparentemente bien, parece ser más una consecuencia casual que causal. De hecho, si se cambia el parámetro random\_state del algoritmo, el resultado obtenido es totalmente distinto (Figura 3).

Chart, treemap chart

Description automatically generated

Figura : Matriz de confusión VGG + K-means (random\_state = 0)

## Detección de anomalías

Para la detección de anomalías se ha decidido orientar el problema en el sentido de una planta de empaquetado de mangos de clase extra. En este caso, se asume que la mayoría de las frutas será de la mejor categoría y solo tendrán que discriminarse algunas que hayan escapado a la clasificación previa. Es decir, los mangos de clase extra se considerarán como la clase normal, y cualquier mango con golpes o manchas serán considerados anomalías.

Para la detección de anomalías se utilizará una red autoencóder, ya que esta permite generar un umbral de detección y visualizar el desempeño del algoritmo. La estructura de la red a utilizar será de tipo convolucional y tendrá la siguiente configuración:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Capa | Filtros | Kernel | Salida | Activación |
| Conv. 2D (entrada) | 64 | 3x3 | 64x64x64 | Relu |
| Max\_pooling2D |  | 2x2 | 32x32x64 |  |
| Conv. 2D | 32 | 3x3 | 32x32x32 | Relu |
| Max\_pooling2D |  | 2x2 | 16x16x32 |  |
| Conv. 2D | 16 | 3x3 | 16x16x16 | Relu |
| Max\_pooling2D |  | 2x2 | 8x8x16 |  |
| Flatten |  |  | 1024 |  |
| Dense |  |  | 512 | Linear |
| Dense |  |  | 1024 | Relu |
| Reshape |  |  | 8x8x16 |  |
| UpSampling2D |  | 2x2 | 16x16x16 |  |
| Conv2D | 32 | 3x3 | 16x16x32 | Relu |
| UpSampling2D |  | 2x2 | 32x32x32 |  |
| Conv2D | 64 | 3x3 | 32x32x64 | Relu |
| UpSampling2D |  | 2x2 | 64x64x64 |  |
| Conv2D | 3 | 3x3 | 64x64x3 | Linear |