Taller 2

Nombres:

Juan Diego Velasquez

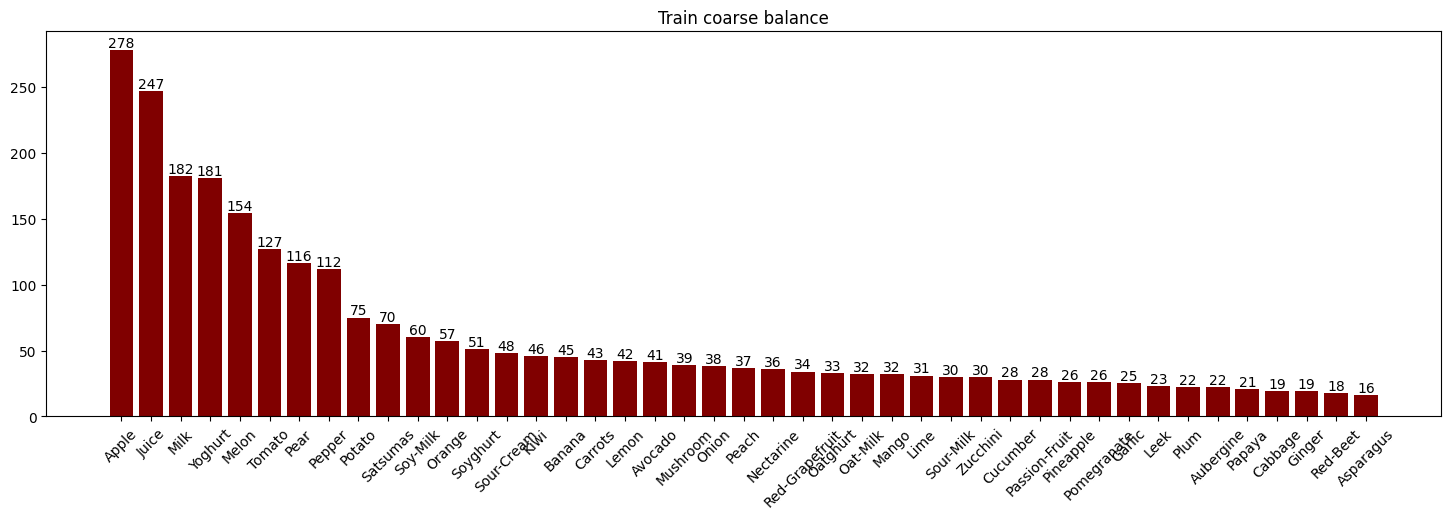
Carlos Niño Ramirez

1. Reporte EDA

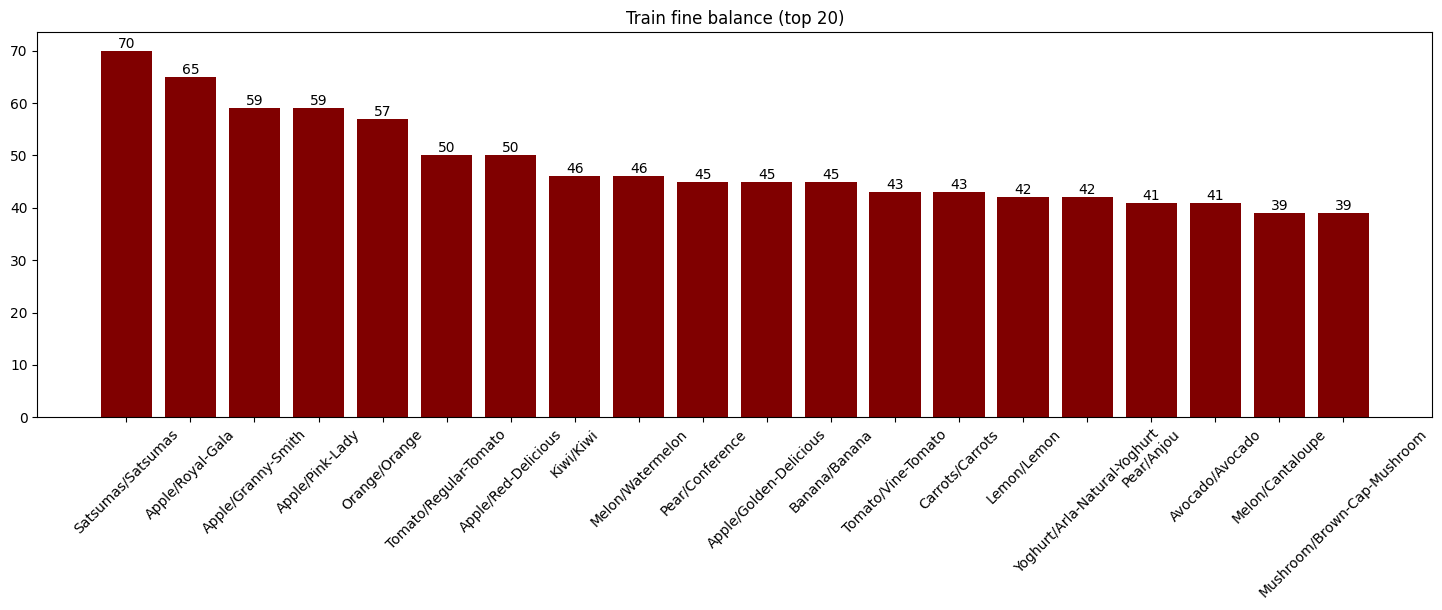
El dataset ya viene dividido en entrenamiento, validación y prueba, 2640 imágenes de en entrenamiento de 43 clases gruesas. El de validación tiene 296 imágenes distribuidas en 37 clases gruesas y el de testing tiene 2485 imágenes en 43 clases gruesas.

**Balance**

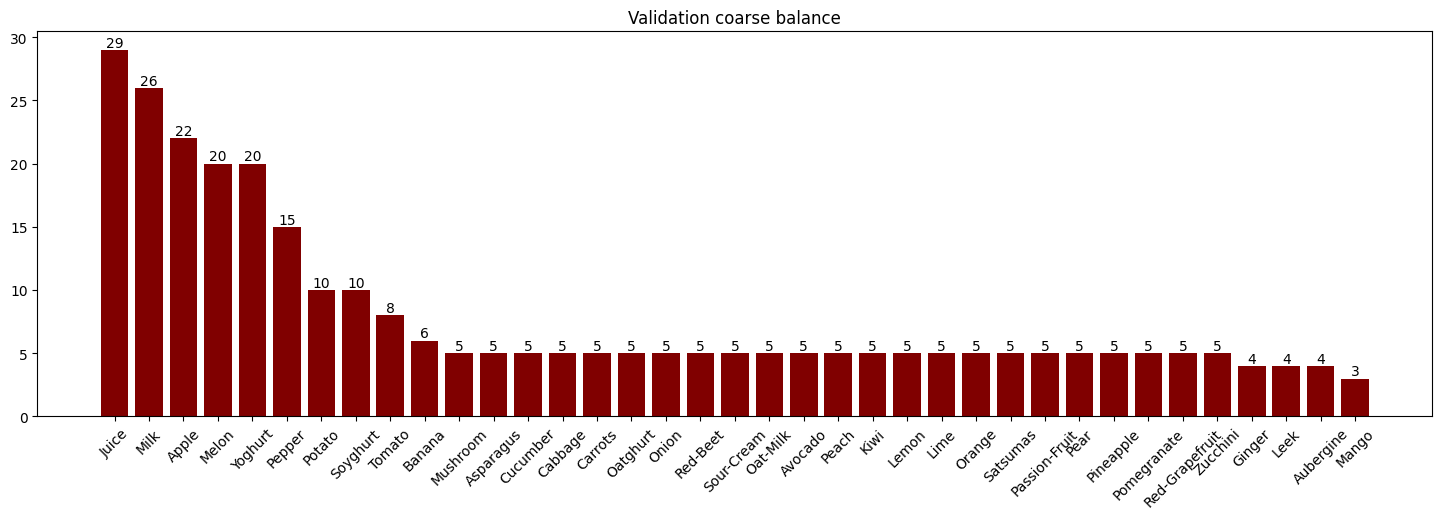
La distribución de las imágenes en las clases gruesas es la siguiente:



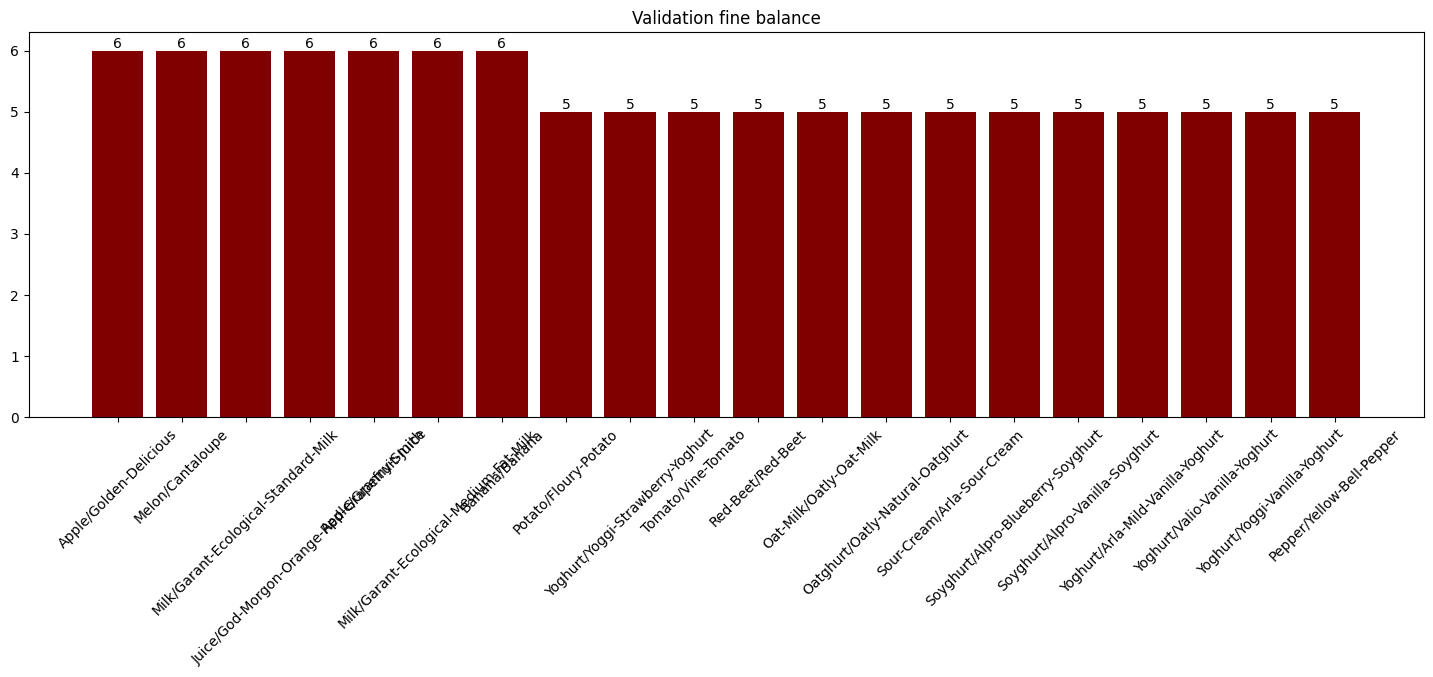
Para las clases finas el top 20 es de:



En validación la distribución es la siguiente para las clases gruesas:

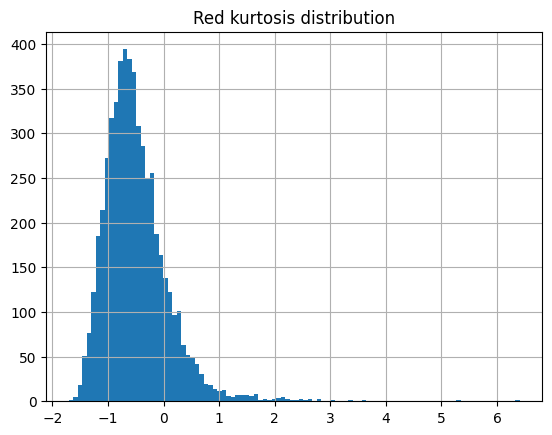
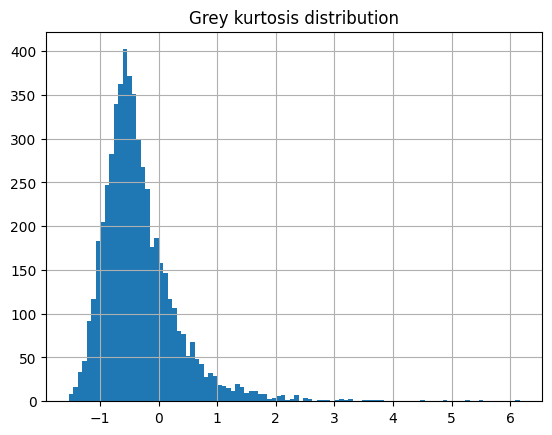


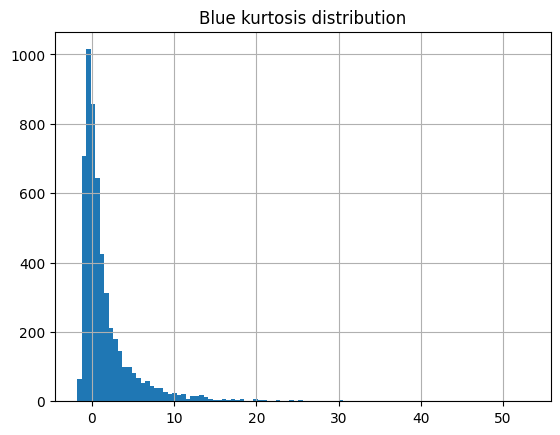
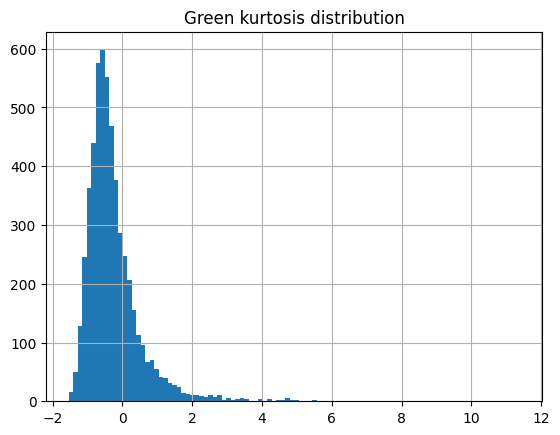
En validación la distribución es la siguiente para las clases finas:



**Curtosis**

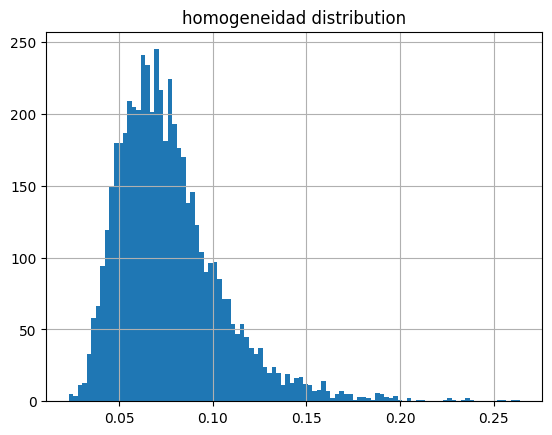
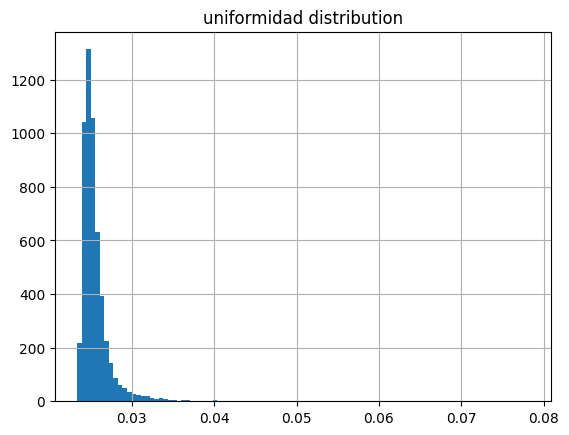
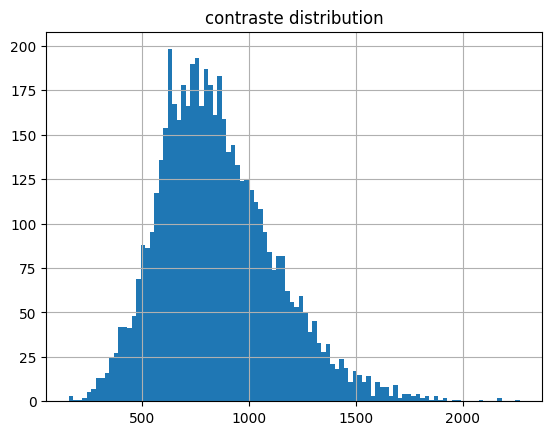
La curtosis para los componentes grises y de RGB es la siguiente:





La distribución de la curtosis en la mayoria de los filtros se ubica en 0 o por debajo, por lo que se suponer que la mayoria de las imagenes tuvieran una gran distribución de los colores rojo verde y azul dentro de su composición.

**Escala de grises.**



Se puede ver que los valores de uniformidad son bajos por lo que al parecer la mayoria de imagenes tienen varios patrones repetitivos, por ejemplo una foto de muchas manzanas o de un mismo elemento varias veces.

1. Entrenamiento del modelo

Se entrenaron dos modelos, un random forest y una red neuronal convolucional.

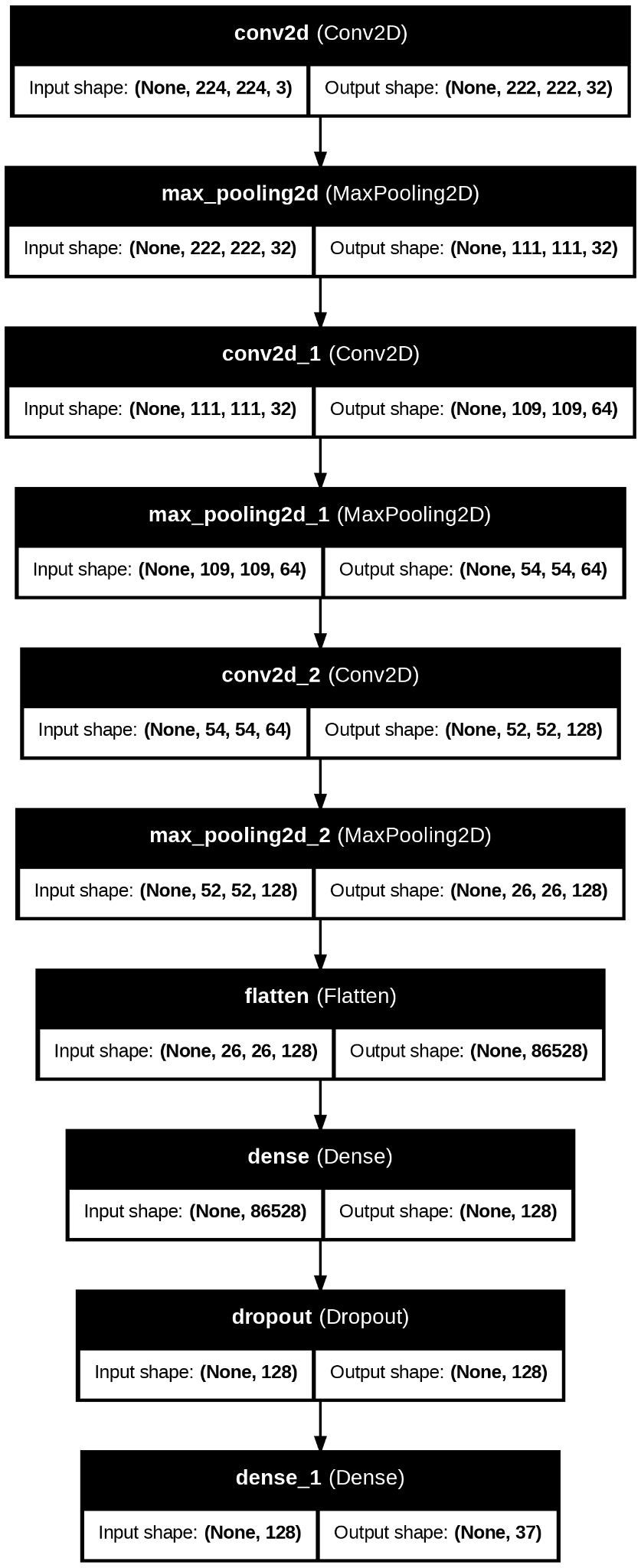
* Random Forest: Se obtiene los valores en escala de grises y de RGB de la matriz de cada imagen, esta posteriormente se aplana para conseguir en un vector de los valores de los pixeles. Con estos vectores se obtiene la curtosis de cada color y usando la matriz de grises se obtiene el contraste, uniformidad y homogeneidad, que se usaron como base para entrenar el modelo.

Adicionalmente se buscaron los mejores parametros para este contexto y se obtuvieron:

class\_weight: None

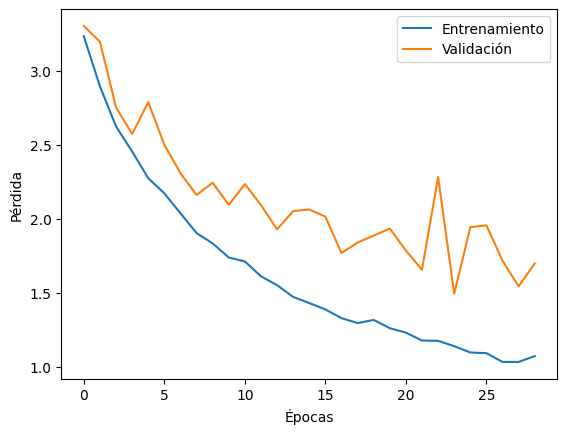
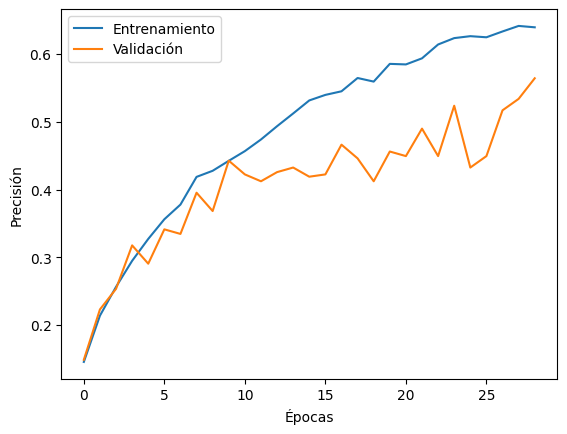
max\_depth: 6

n\_estimators: 150

* CNN: Para entrenar la red neuronal se utilizó el conjunto de datos de entrenamiento y el de validación para elegir el mejor modelo. La topología utilizada fue la siguiente: 

Al entrenar el modelo notamos que la precisión en validación aumentaba considerablemente si se eliminaban las categorías que estaban ausentes en el conjunto de validación del entrenamiento. las categorías eliminadas fueron: 'Papaya', 'Sour-Milk', 'Nectarine', 'Soy-Milk', 'Plum', 'Garlic'.

Al final el mejor modelo que logramos obtener alcanzó una precisión del 52% en el conjunto de validación. El entrenamiento de la red tuvo el siguiente comportamiento:



1. Analisis de resultados

RandonForest:

A nivel de test se obtuvo un nivel de exactitud de 39% precisión de 51% y recall de 39%, estos valores varian bastante frente a los mismos valores de training y de validación, esto se puede dar por la diferencia en calidad en las imagenes de los diferentes grupos junto con el hecho de que los mismos grupos estaban desbalanceados ya que no se tenia la misma cantidad de categorias para validación.

CNN:

Tanto el tamaño de la imagen como la complejidad de la red(número y variedad de capas) determinan el tiempo de entrenamiento y el costo computacional que esto conlleva. Se pueden lograr mejores resultados con una red más compleja pero se vuelve indispensable el uso de una GPU.

En este caso obtuvimos una precisión del 52% un recall del 36% y una exactitud 32% de excluyendo las clases faltantes del dataset de validación del entrenamiento.

En general una oportunidad de mejora sustancial es mejorar el balance de las clases ya que es importante que todas las categorías tengan la misma o al menos una cantidad cercana de imágenes.

Por otro lado con el escaso recurso de cómputo que tuvimos a disposición se podía ver una mejora en el entrenamiento de las CNN por lo que se podría invertir en recursos de computo para obtener mejores modelos que puedan llegar a tener exactitudes del 80%.

1. Generar valor.

**Costo original:** suponiendo que el costo de un cajero es del salario minimo actual es decir aproximadamente 5.400 por hora. Suponiendo que el operario escanea 1000 productos por hora (aproximadamente 15 por minuto) se obtendria un costo de 5.4 pesos por escaner.

**Costo de predicción:** Suponiendo una predicción se demora 0.01 segundos y el costo de una GPU es de 0.35 dólares de GPU entonces el costo por segundo de la GPU es de $0.0001 dólares por lo que una predicción puede costar $0.000001 dólares.

**Ahorro esperado:** 5.4 - 0.0000045 = 5.3999955

**Precisión del modelo de CNN:** 52%.

**Ahorro por prediccion:** 2.8079955 pesos

Suponiendo que se venden 50.000 productos al mes es decir se realiza este número de predicciones el ahorro sería de 140.000 pesos, el salario de un datascientist es de 6 millones el punto de equilibrio seria de 42 meses, es decir, 3.6 años.

1. insights

* Es necesario obtener más imágenes de las clases desbalanceadas para tener una mayor exactitud teniendo en cuenta todos los productos de la tienda.
* Teniendo una mayor inversión en capacidad de cómputo el proyecto tendría un mucho mejor retorno ya que el ahorro por cada predicción sería mucho más considerable y se llegaría al punto de equilibrio más rápidamente
* Debido al bajo costo de la infraestructura en la nube, una vez desplegado el modelo el costo por predicción es muy bajo, por lo que vale la pena desarrollar un mejor modelo.