Universidad Autónoma de Bucaramanga

FACULTAD DE INGENIERÍAS

Desarrollo de un estudio para la implementación de cosecha selectiva de café arábica aplicando vibraciones de alta frecuencia

TESIS PARA OPTAR POR EL TÍTULO PROFESIONAL DE INGENIERO MECATRÓNICO

Autor: Jeison Ivan Yarce Herrera

Director: Carlos Arizmendi

Bucaramanga, 2022

COLOMBIA

DEDICATORIA

para mi familia y aquellos que incondicionalmente me han ayudado
a lo lardo de mis estudios de pregrado.

AGRADECIMIENTO

Gracias, padre y madre por todo el tiempo que me han dedicado y los valores que me han compartido, su ejemplo me ha enseñado a ser una persona ética y responsable. Gracias familia, sin su apoyo incondicional no sería la persona que soy ahora. Un agradecimiento grande a mis compañeros, profesores y a la Universidad autónoma de Bucaramanga. Adicionalmente, estoy muy agradecido con el Ing. Carlos Arizmendi. por la motivación constante para culminar con esta tesis de grado.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

1 Introducción

"Algunas personas denominan a esta tecnología inteligencia artificial, cuando en realidad lo que va a permitir es que aumente la nuestra propia"

Gin Rometty

1.1 Motivación

El café colombiano es reconocido especialmente por su aroma y sabor; esto se debe en gran parte a las condiciones favorables del clima y a la selectividad de frutos de café en un estado de maduración completa en la recolección. La recolección selectiva de frutos maduros garantiza la calidad superior en la taza de café; este proceso implica sólo la recolección de frutos maduros; es, por tanto, un proceso dispendioso, arduo y delicado; que se realiza de forma manual en Colombia dado que en una misma rama de arbustos de café pueden encontrarse bulbos que se convertirán en flores, flores formadas, y frutos en diferente estado de maduración. La recolección selectiva es una actividad que requiere mayor cantidad de trabajadores. Sin embargo, en los últimos años es cada vez más evidente la escasez de mano de obra para la recolección de las cosechas.

En el presente estudio se propone analizar un dispositivo que permita estimular los frutos maduros discriminando el movimiento de los frutos verdes en frecuencias muy específicas de movimiento. La tecnología del dispositivo se propone sobre la base de un dispositivo acústico, con una técnica que permita focalizar la energía mediante arreglos de ondas armónica que deben ser controladas para excitar solo frutos maduros. Por lo tanto, una oportunidad es ampliamente

observada en el estudio de la subestructura fruto pedúnculo para determinar en los diferentes estados de maduración índices dinámicos que favorezcan el desprendimiento de frutos maduros.

Un procedimiento numérico ya fue establecido y para lograr el objetivo de este proyecto de excitar solo frutos maduros, un procedimiento experimental es necesario para la determinación de las funciones de respuesta en frecuencia (FRF) que están asociadas a cada estado de maduración con el objetivo de buscar intervalos de frecuencia que favorezcan el desprendimiento de los frutos maduros. La propuesta se desarrolla con el fin analizar diferencias significativas en las frecuencias naturales para establecer cuáles frecuencias y modos de vibración son ideales para el desprendimiento y así usar el dispositivo acústico propuesto para la realización de la cosecha selectiva.

Este estudio se ha venido trabajando y se encontró mediante un análisis por elementos finitos que las frecuencias naturales del sistema fruto pedúnculo en estado de maduración varían entre 100 y 200 Hz. Por ese motivo este proyecto parte de validar experimentalmente los hallazgos que se hicieron previamente. Por otro lado, se plantea el desarrollo de una aplicación móvil basado en técnicas de Deep Learning que emplea redes neuronales convolucionales, en la actualidad es un método que se usa para la detección o clasificación y presenta alto grado de precisión en los resultados obtenidos

El café colombiano es reconocido especialmente por su aroma y sabor; esto se debe en gran parte a las condiciones favorables del clima y a la selectividad de frutos de café en un estado de maduración completa en la recolección. La recolección selectiva de frutos maduros garantiza la calidad superior en la taza de café; este proceso implica sólo la recolección de frutos maduros; es, por tanto, un proceso dispendioso, arduo y delicado; que se realiza de forma manual en Colombia dado que en una misma rama de arbustos de café pueden encontrarse bulbos que se convertirán en flores, flores formadas, y frutos en diferente estado de maduración. La recolección selectiva es una actividad que requiere mayor cantidad de trabajadores. Sin embargo, en los últimos años es cada vez más evidente la escasez de mano de obra para la recolección de las cosechas.

En el presente estudio se propone analizar un dispositivo que permita estimular los frutos maduros discriminando el movimiento de los frutos verdes en frecuencias muy específicas de movimiento. La tecnología del dispositivo se propone sobre la base de un dispositivo acústico, con una técnica que permita focalizar la energía mediante arreglos de ondas armónica que deben ser controladas para excitar solo frutos maduros. Por lo tanto, una oportunidad es ampliamente observada en el estudio de la subestructura fruto pedúnculo para determinar en los diferentes estados de maduración índices dinámicos que favorezcan el desprendimiento de frutos maduros.

Un procedimiento numérico ya fue establecido y para lograr el objetivo de este proyecto de excitar solo frutos maduros, un procedimiento experimental es necesario para la determinación de las funciones de respuesta en frecuencia (FRF) que están asociadas a cada estado de maduración con el objetivo de buscar intervalos de frecuencia que favorezcan el desprendimiento de los frutos maduros. La propuesta se desarrolla con el fin analizar diferencias significativas en las frecuencias naturales para establecer cuáles frecuencias y modos de vibración son ideales para el desprendimiento y así usar el dispositivo acústico propuesto para la realización de la cosecha selectiva.

Este estudio se ha venido trabajando y se encontró mediante un análisis por elementos finitos que las frecuencias naturales del sistema fruto pedúnculo en estado de maduración varían entre 100 y 200 Hz. Por ese motivo este proyecto parte de validar experimentalmente los hallazgos que se hicieron previamente. Por otro lado, se plantea el desarrollo de una aplicación móvil basado en técnicas de Deep Learning que emplea redes neuronales convolucionales, en la actualidad es un método que se usa para la detección o clasificación y presenta alto grado de precisión en los resultados obtenidos

1.2 Objetivos

El objetivo final de este proyecto consiste en:

Desarrollar un estudio para implementar tecnología de cosecha selectiva para frutos de café arábica aplicando vibraciones de alta frecuencia.

Los objetivos específicos necesarios para poder cumplirlo:

- Desarrollar un sistema de aprendizaje profundo (Deep Learning) que permita clasificar los frutos de café en estado de maduración.
- Analizar el efecto que generan las vibraciones sobre sistema fruto pedúnculo cuando se varía la frecuencia y amplitud de la fuente acústica.
- Desarrollar una metodología optima basados en el control de frecuencia y amplitud que permita la recolección de frutos de café en un ambiente controlado.
- Diseñar un sistema de focalización acústica para el control de arreglos armónicos que generen vibraciones focalizadas de alta intensidad.

Las tareas abordadas durante el desarrollo del trabajo se clasifican dentro de los siguientes campos

- **Inteligencia artificial**, concretamente en la rama de aprendizaje profundo.
- **Acústica**, en lo relevante de medidas de potencia sonora y vibraciones.

Capítulo 2

2 Estado del arte

2.1 Introducción

Podemos tener una intuición equivocada de que la detección de objetos sea una tarea sencilla, pero realmente no lo es y es de muchas maneras un gran problema a resolver. Para poder llegar al punto de realizar detecciones se han logrado grandes progresos gracias a tecnologías, tales como la extracción de flujos ópticos y diferentes aproximaciones en el aprendizaje profundo. Sin embargo, debido al gran éxito de arquitectura de aprendizaje profundo en clasificación de imágenes, los avances de detección múltiple de objetos se han producido de una manera mucho más lenta, siendo ahora mismo un campo altamente abierto a la investigación.

Las principales razones por las cuales el avance ha sido lento en este campo son las siguientes:

- Coste computacional: se requieres un gran poder de cómputo en paralelo para entrenar diferentes arquitecturas de aprendizaje profundo, ya que pueden tener millones de parámetros.

- Se requieren miles de datos de entrada para entrenar el modelo tales como el número de ejemplos, la cantidad de clases y la exhaustiva tarea que conlleva etiquetar manualmente dichos datos.

- Comparado con la clasificación de imágenes, no solo se necesita información espacial sino también se debe tener en cuenta el contexto en el que se encuentran los objetos.

2.2 Estado del arte en detección de objetos

Dentro de las ramas de visión artificial, la detección de objetos en imágenes es una de las más primordiales. La posibilidad de detección de objetos supone un fuerte apoyo en tareas de estimación de actividades humanas, detección de frutos dispuestos para ser cosechados y diferentes estados de un objeto, todos estos aplicados en este trabajo.

A pesar de sus similitudes, la detección de objetos presenta notable diferencias con respecto a la clasificación de imágenes- Un algoritmo de detección de objetos en imágenes, pretende señalar por media de cuadros o cajas delimitadoras las zonas de una imagen en la que se ha detectado una clase determinada que pertenece a un objeto con un umbral de confianza.

Una de las limitaciones a la que se enfrenta, es que en una única imagen pueden aparecer numerosas detecciones, y no se puede determinar el número de antemano-Esto significa, que el tamaño de la capa de salida de la red es variable.

Las posibles formas de solucionar estos problemas consisten en dividir la imagen en diferentes zonas de interés, y aplicar sobre todas estas una red neuronal, de esta forma es posible determinar la presencia de un objeto en esa región especifica. No obstante, uno de los principales problemas que es evidente en esta aproximación, es que no todos los objetos tendrían que estar dentro de una de estas posibles regiones, debido a que podrían situarse en posiciones diferentes en la imagen y tener tamaños variables. Por lo tanto, el coste computacional de tener que aplicar una red neuronal a cada una de las regiones posibles dentro de una imagen, teniendo en mente lo anterior, seria enorme, de forma que no resulta realizable en la práctica.

En este orden para poder detectar y seleccionar las regiones de interés dentro de la imagen, sin someterse a esos costes computacionales limitantes, se han desarrollado algoritmos como los explicados a continuación

2.2.1 R-CNN

Para poder resolver el problema de seleccionar un gran número de regiones de una imagen, Ross Girshick et al. (24), proponen R-CNN, que se enfoca en la utilización de una técnica de detección basada en propuestas de región, con el propósito de detectar objetos en imágenes.

Mediante esta técnica, es posible generar un total de 2000 cajas delimitadoras o "bounding boxes", que son dirigidas posteriormente a una red neuronal convolucional, que extrae las características de estas cajas delimitadoras.

Después de eso, el sistema usa una máquina de vectores de soporte (SVM) (19), que es la especializada en este caso de clasificar si el objeto propuesto es en efecto un objeto o no. En caso de tratarse de una clase, la SVM también será la encargada de predecir de que objeto se trata. además, es aplicada una función de "supresión no máxima", si algún objeto es detectado más de una vez. Todas las detecciones de un mismo objeto no se tienen en cuenta, lo anterior es denominado supresión no máxima.

Éste es un modelo bastante complejo, donde cada uno de esos pasos debe ajustarse específicamente para que funcione correctamente, lo que resulta en una velocidad de rendimiento de cerca de 4000 milisegundos por fotograma, que es notablemente inferior a otros detectores de objetos, haciendo imposible su aplicación en tiempo real.

2.2.2 Fast R-CNN

En 2015, Ross Girshick presentó Fast R-CNN (23). A pesar de la gran acogida que tuvo R-CNN, tenía algunos problemas que requerían ser resueltos, entre ellos:

- Era mucho más lento como se indica previamente, ya que necesitaba realizar los cálculos para cada región indicada.
- Dificultad para realizar el entrenamiento, tener en cuenta que R-CNN debía entrenar tres partes de forma separada (CNN, SVM, y el regresor de las cajas delimitadoras)
- Gasto excesivo de memoria, dado que se requiere almacenar cada resultado de calcular el mapa de informacion de la región especificada.

La forma en que Girshick resolvió dichos problemas fue desarrollando esta nueva derivación simplificada del cambio que presupone Fast-RCNN respecto a R-CNN, es que en este se combinan las tres partes diferenciadas presentes en el sistema R-CNN (CNN, SVM, y el regresor de las cajas delimitadoras) en una sola arquitectura, figura 2.1.

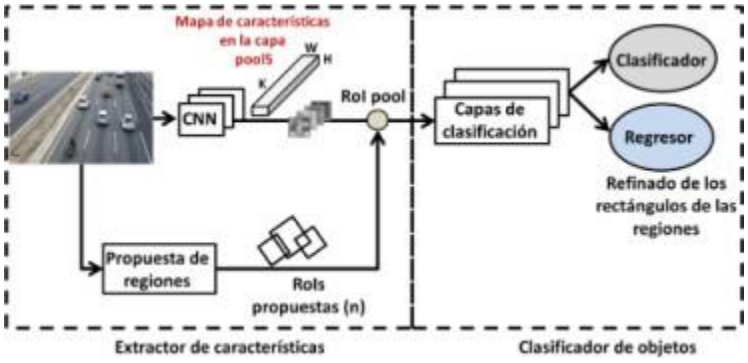


Figura 2.1: Arquitectura Fast-RCNN

El algoritmo Fast-RCNN se compone de 4 partes:

1. Procesamiento completo de toda la imagen mediante la CNN, obteniendo el mapa de información de la misma.
2. Para cada una de las regiones de interés, se extrae la parte correspondiente del mapa de información obtenido anteriormente. Posteriormente re-escala el mapa de información de las regiones propuestas, mediante la ayuda de una capa de agrupamiento.
3. El mapa de características de las regiones de interés es transformado en un vector del mismo tamaño que contiene la informacion
4. Se hace uso del vector obtenido, como entrada para la última capa. Debe pasar por una capa softmax predice los objetos, y al final pasa al regresor de cajas delimitadoras, que da como salida la posición de los objetos que se han encontrado.

.

2.2.3 Faster R-CNN

Los algoritmos anteriores (RCNN y Fast-RCNN), se basan en un tipo de búsqueda selectiva a la hora de encontrar propuestas de regiones, tomando una gran cantidad de tiempo. No obstante, Faster-RCNN, propuesta por Shaoqing Ren et al. (49), elimina este sistema de búsqueda de regiones, logrando que sea la misma red neuronal la que aprenda las propuestas de regiones.

Al igual que Fast-RCNN, la imagen se proporciona como una matriz de entrada a una red convolucional que genera un mapa de información como resultado. Sin embargo, para localizar las regiones propuestas, en lugar de utilizar el algoritmo de búsqueda selectiva en el mapa de información obtenido anteriormente, se utiliza una red solo para predecir las regiones, la denominada Red de Propuesta de Región (RPN), figura 2.2. La finalidad de la RPN es generar un conjunto de regiones, que tienen una probabilidad de ser una determinada clase.

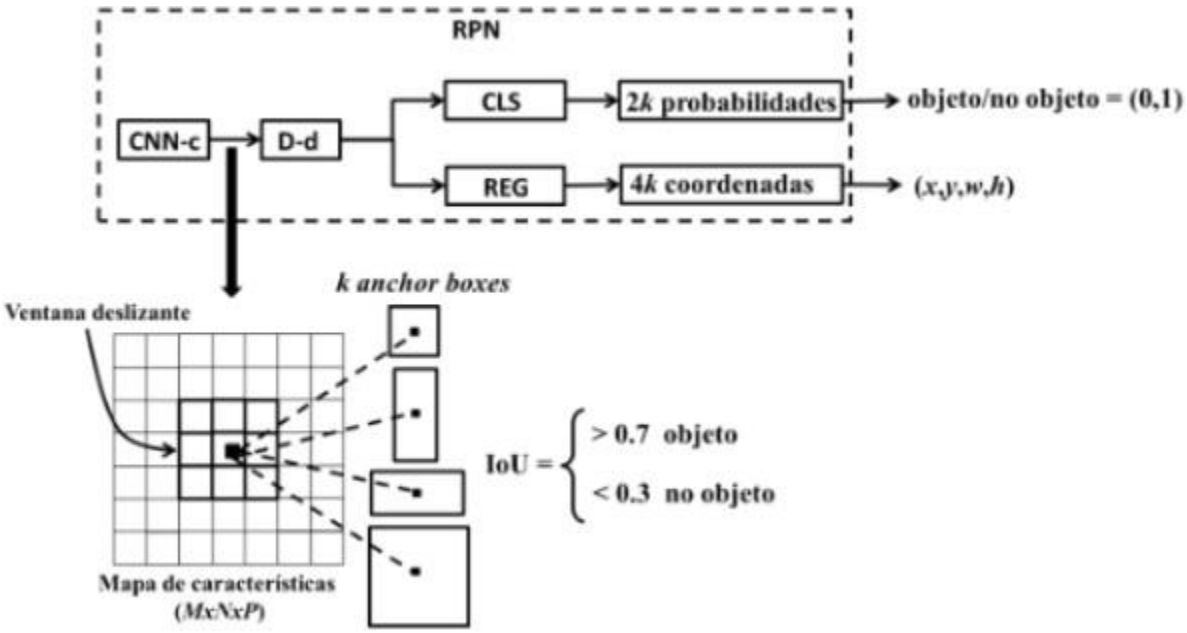


Figura 2.2: Red de regiones de interés

Las regiones de interés predichas por la RPN se re-escalan usando una capa de pooling denominada región de interés (ROI), que se utiliza para clasificar con ayuda de una R-CNN la imagen en una región de interés y de esta manera predecir los valores de desplazamiento para las cajas delimitadoras.

El modelo Faster-RCNN es mucho más rápido que sus predecesores, logrando en algunos casos ser aplicado en tiempo real. Es uno de los métodos más usados en la actualidad, siendo uno de los mejores algoritmos de detección de objetos disponibles.

2.2.4 YOLO

"You Only Look Once", o YOLO, es uno de los algoritmos de detección de objetos más eficientes y eficaces que existen, abarcando muchas de las ideas más innovadoras en los colectivos de investigación de visión artificial. Es una opción viable cuando se trata de detecciones en tiempo real, sin perder precisión en sus resultados.

YOLO es un algoritmo creado por Redmon et al. (46) en 2015, teniendo una gran acogida por los investigadores de visión artificial. La orientación de este algoritmo es diferente de los anteriores, siendo proyectado como una red neuronal capaz de detectar en tiempo real.

YOLO aplica una única red neuronal en la totalidad de la imagen, dividiendo posteriormente la imagen en regiones y predice cajas delimitadores y probabilidades para cada región. Estas cajas delimitadoras están ponderadas en base a las probabilidades predichas.

Como sus siglas lo indican, YOLO solo" mira una vez" la imagen, es decir, solo necesita un paso forward en la red neuronal para finalizar devolviendo las predicciones. Después de esto, aplica una función de supresión no máxima, para eliminar aquellas detecciones reiteradas de un mismo objeto.

Entre las características más primordiales de este algoritmo están:

- Su alta velocidad.
- Es posible codificar implícitamente información contextual sobre las clases y su apariencia.
- Aprende representaciones generalizadas de los objetos.

Un año después, en 2016, se publicó la versión posterior del algoritmo, YOLO9000 o YOLO v2. Fue resultado de las investigaciones de Redmon et al. (47), con el objetivo de potenciar la primera implementación. Entre las mejoras destaca la capacidad de detectar más de 9000 tipos diferentes de objetos. En 2018, los investigadores publicaron la versión más modernizada de YOLO, YOLO

v3 (48), capaz de ejecutarse significativamente más rápido que otros métodos de detección con un rendimiento similar. Además, dentro de las mejoras corrige uno de los problemas de las anteriores versiones de YOLO, como es la detección de objetos de tamaño pequeño. En el caso de YOLO v3, utiliza una red neuronal convolucional denominada DarkNet-53, de mayor cantidad de capas y aún más compleja que la usada en YOLO v2, la DarkNet-19. En la figura 6.2, se representa la arquitectura detallada de Yolov3.

2.3 Estado del arte en cosecha selectiva

Como antecedentes diferentes estudios enfocaron sus esfuerzos en el estudio de la cosecha mecanizada aplicando vibraciones mecánicas de diferentes formas y controladamente, entre los que podemos mencionar están [5-6]. En estos estudios, los dispositivos diseñados fueron probados en las diferentes partes del árbol; en el tronco [4] así como en las ramas [7-8]. Estos dispositivos utilizaron diferentes tipos de actuadores que promovían vibraciones mecánicas de forma directa en el árbol. Debido a la baja eficiencia de cosecha de frutos maduros revelada, la cosecha manual sigue siendo el método preferido de recolección en Colombia hasta el día de hoy. Pero en países como Brasil, la mecanización agrícola ha impulsado el aumento en la producción, la reducción en los costos entre el 30 al 40%, la rapidez en las operaciones del campo.

2.3.1 Vibraciones mecánicas

En el campo de las vibraciones mecánicas los componentes que determinan las resonancias de cualquier sistema mecánico son definidas principalmente por la rigidez (geometría y propiedades elásticas), amortiguamiento (disipación de energía) y la masa [9-10].

Cuando hablamos de resonancias nos referimos a frecuencias de movimiento de un valor especifico que promueven que toda la energía pueda ser focalizada en ciertos puntos de una estructura ocasionando en algunos casos que estas sean cambiadas o destruidas dependiendo de la cantidad de energía suministrada. Estos parámetros pueden ser medidos en estructuras diseñadas, pero en sistemas biológicos los retos son diferentes, ya que la caracterización de los árboles de café presenta una variación considerable en su distribución geométrica (formas, ramas, tallo) y las estimaciones de todas las propiedades debe ser hechas aplicando una combinación de experimentos y herramientas computacionales.

Algunas de las subestructuras de árbol, tales como la estructura fruto-pedúnculo se caracterizan por diferencias topológicas menos significativas. Esto puede ser tomado como una ventaja, ya que en términos de caracterización es posible construir geométricamente un fruto, evaluar las propiedades mecánicas de forma experimental y por lo tanto estimar las frecuencias de resonancia; aunque no es una tarea fácil, pero es realizable como lo han reportado [11-12]. Otras investigaciones consideran esta subestructura como un sistema fundamental del árbol para el desarrollo de dispositivos vibratorios [13].

2.3.2 Técnicas visuales

A lo largo de la historia, las estimaciones de los rendimientos de los cultivos se han basado en condiciones agronómicas del cultivo, datos históricos de rendimiento del cultivo y observaciones visuales del cultivo (Dorj et al., 2017) En los huertos de naranjas, las técnicas visuales a menudo se usan para estimar los rendimientos; sin embargo, los métodos son altamente subjetivos porque dependen del conocimiento técnico y experiencia. Un error entre el rendimiento real y el rendimiento. Se puede producir una predicción del 15-25% (Castro-García et al., 2019). Por lo tanto, Desarrollar un método altamente preciso para estimar el rendimiento del cultivo y la fruta. El tamaño antes de la cosecha es la clave para ayudar a los agricultores a tomar decisiones.

En los últimos años, debido a los avances en computadoras, cámaras e imágenes técnicas de análisis, una amplia gama de metodologías basadas en conteo. Se han desarrollado números de frutas para estimar los rendimientos de los cultivos (Gongalet al., 2015). Por ejemplo, algunos estudios anteriores sobre la recuperación de frutas, la cognición se realizó en manzanas (Aggelopoulou et al., 2011 ; Gongalet al., 2018; Zhou et al., 2012), cítricos (Blasco et al., 2003; Gonget al., 2013; Kurtulmus et al., 2011 ; Lin et al., 2020, 2019a ; Okamotoy Lee, 2009), tomates (Yamamoto et al., 2014 ; Zhao et al., 2016)y guayaba (Lin et al., 2019b) Sin embargo, la mayoría de estas técnicas utilizan algoritmos basados en la respuesta espectral entre píxeles como un única función para detectar y contar frutas (Burnett y Blaschke, 2003).

Como los píxeles de la imagen son muy sensibles a los cambios de iluminación bajo condiciones de luz estructuradas, el uso de tales algoritmos puede reducir la precisión de detección de fruta (Lin et al., 2019a; Rosebrock, 2016)

2.3.3 Propiedades de la fruta del café

El estudio de las propiedades de la fruta de café no es una nueva tarea; por ejemplo, Chandrasekar y Viswanathan (1999) Determinado el tamaño, densidad, masa, pergaminos propiedades, y la resistencia al aplastamiento de arábica y robusta frutos de café utilizando semi-madura, madura, y las muestras demasiado maduras. Posteriormente, Ciro (2001) Determinaron algunas propiedades físicas, geométricas y mecánicas para estudiar la dinámica de la fruta-pedúnculo de café.

Carvajal-Herrera et al. (2012) demostró experimentalmente que la masa, la firmeza y la fuerza de desprendimiento son parámetros que dependen de estados de madurez en el café arábica. Hace poco, Coelho et al. (2015) También determina las propiedades elásticas (módulo de Young), geométricos (dimensiones) y mecánica (coeficiente de amortiguamiento) del sistema de pedúnculo de la fruta arábica por medio de procesamiento de imagen digital y ensayos mecánicos. Podemos mencionar que, con respecto a la mayoría de los estudios realizados con frutos de café, éstos aún no han establecido un modelo matemático que correlaciona las propiedades físicas con los parámetros dimensionales. Por lo tanto, este aspecto se considera como corriente.

Investigaciones acerca de la correlación matemática entre la geometría y las propiedades físicas han sido estudiados por diferentes especies de frutas (Spreer y Müller, 2011). Por ejemplo, Tabatabaeefar y Rajabipour (2005) Desarrollaron un modelo matemático para las manzanas que se correlaciona masa y el tamaño y permite estimar el volumen. Khoshnam et al. (2007) Propuesto una ecuación para determinar la masa de granadas como una función de un parámetro geométrico de la fruta. Otros estudios son mencionados por Spreer y Müller (2011). En el caso del café arábico de variedad colombiano (Tinoco,2017) desarrolla un modelo matemático que relaciona el volumen de la fruta con las dimensiones ortogonales además de que presenta una ecuación para el módulo de Young de la fruta.

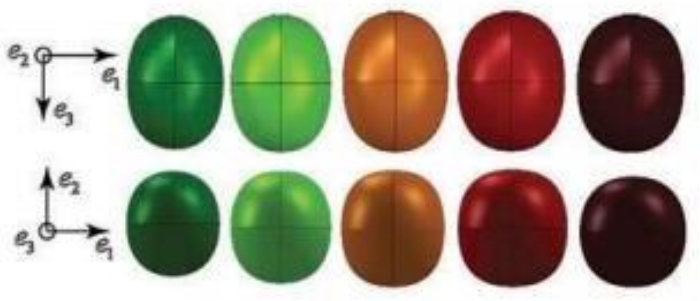


Figura 2.3: CAD para las diferentes etapas de maduración obtenidos por Tinoco.

Las ecuaciones paramétricas que describen la geometría en los tres principales planos de frutas (Coffea arabica L. var. Colombia) para todos En este estudio se determinaron las etapas de maduración. Modelos geométricos de la fruta fueron validados calculando el error volumétrico con datos experimentales de volumen; esos modelos mostraron un enfoque menos del 9%. Se propuso un proceso numérico-experimental para determinar el módulo de Young del fruto del café en diferentes etapas de maduración. Las propiedades elásticas mostraron que cuando el fruto aumenta sus días de maduración, el módulo de Young disminuye. Este significa que el fruto comienza a perder su capacidad elástica cuando es madurando. (Hector A. Tinoco et al., 2014)

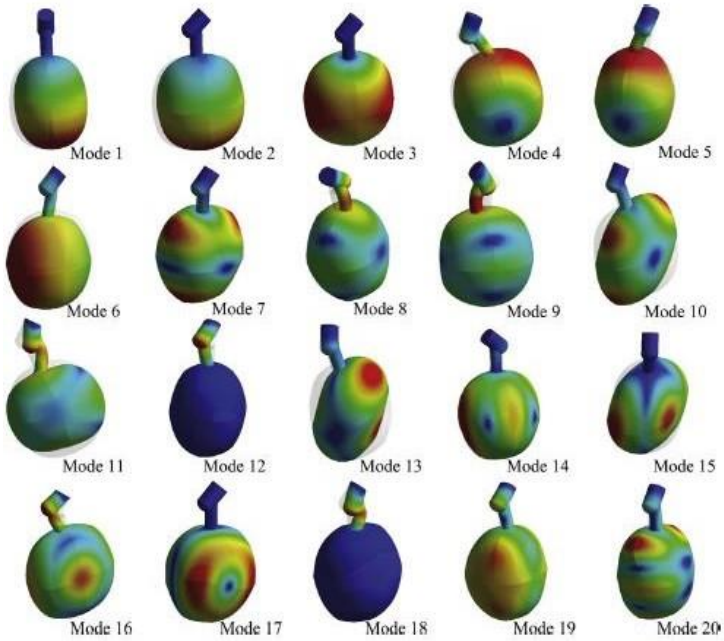


Figura 2.4: FEM modos de vibración de fruto pedúnculo

2.3.4 Análisis armónico

Se realizó un análisis armónico de estrés para evaluar la intensidad del estrés en las interfaces pedicelo-fruta y pedicelo pedúnculo de Coffea arabica L. var. Colombia. Se identificaron tres modos de vibración en el espectro de frecuencias de 0-400 Hz mediante un análisis de elementos finitos. Observamos que las frecuencias naturales disminuían cuando aumentaba la maduración del fruto. Con el fin de realizar una cosecha selectiva, la frecuencia natural 2 (128 Hz) demostró ser adecuada en su comportamiento dinámico, ya que el sistema de pedúnculos de la fruta-pedicura experimentó movimientos de flexión. Y se analizó que la excitación dinámica entre 120 y 150 Hz podía ayudar a separar selectivamente los frutos maduros según los resultados numéricos. Estos resultados muestran que la estimulación dinámica del cafeto debe completarse en frecuencias entre 100 y 200 Hz. (Hector A. Tinoco et al., 2017)

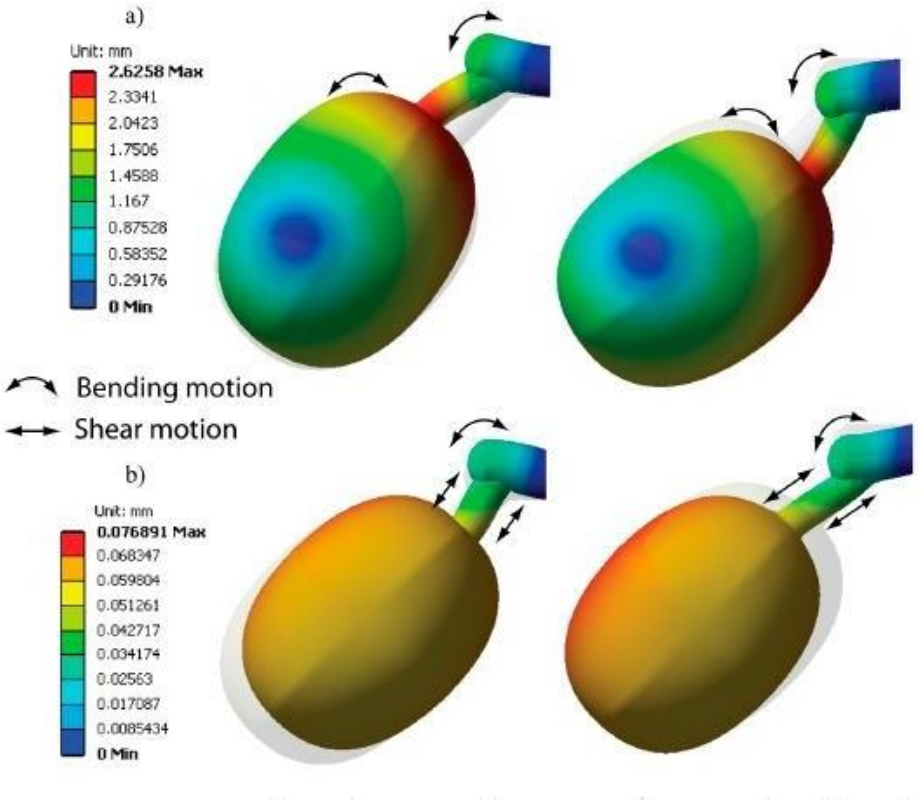


Figura 2.5: FEM modos de vibración de fruto pedúnculo maduro

2.4 Principales tecnologías utilizadas

2.4.1 Python

Python es un lenguaje de programación de alto nivel difundido por Guido van Rossum (52), cuenta con una gran versatilidad, permitiéndole ser multiplataforma y multiparadigma. Python enfatiza la legibilidad del código y su limpieza, lo que facilita su utilización.

Es un lenguaje interpretado que contiene un fuerte tipado dinámico y una administración de la memoria automática, siendo, además, compatible con variados paradigmas de programación, incluidos los imperativos, los orientados a objetos y en mucho menor medida a los funcionales.

Python contiene una librería estándar grande y muy completa, además destaca aún más por la cantidad de librerías y documentación que tiene gracias a grupos de terceros.

2.4.2 Pytorch

Pytorch (45) es una biblioteca de código abierto basada en la arquitectura Torch, es decir un paquete de computación científica basado en Python, con especial énfasis en el uso de unidades de procesamiento gráfico (GPU).

Desde que esta librería fue lanzada en 2016 por Facebook, muchos investigadores la han usado debido a que proporciona la posibilidad de realizar cálculos complejos de tensores con un gran soporte de aceleración por GPU y por su facilidad para crear redes neuronales profundas de enorme complejidad.

Al tratarse de una librería muy esencial crea una metodología ideal para encajar con un estilo de programación phytonico, lo que, gracias a su interfaz simple, y a sus gráficos computacionales. Además, permite una gran flexibilidad a la hora de ser usado en una amplia diversidad de algoritmos diferentes y por esta razón, ha sido utilizado en muchos sectores de la computación, como reconocimiento por voz, visión computacional, procesamiento del lenguaje y detección de objetos.

Aunque aún no ha sido adoptada al nivel de TensorFlow (12) debido a que aún está ampliándose y sigue en construcción, está generando una dura competencia. La comunidad de

Pytorch está creciendo paulatinamente y está consolidándose como uno de los paquetes principales de todo investigador, siendo usada por gigantes tecnológicos como Facebook, Twitter o Nvidia.

2.4.3 OpenCV

Es una librería de código abierto destinada a temas de visión computacional, desarrollada por Intel y lanzada en el año 2000 (3). Fue construida para suministrar un entorno de desarrollo sencillo y fácil de utilizar a la hora de crear una infraestructura común para aplicaciones de visión por computador.

La librería permite crear y trabajar con una cantidad considerable de visión artificial y de aprendizaje profundo, que pueden ser utilizados para reconocimiento de rostros, seguimiento y detección de objetos, edición de videos y de imágenes, entro muchos otros. importantes compañías hacen uso de esta poderosa herramienta a la hora de crear sistemas de detección por videovigilancia o software de identificación de objetos para robots. Proporciona interfaces en C++, Java, Matlab y Python, por esto, resulta de gran utilidad para el proyecto a la hora de procesar los vídeos.

Figura 2.6: frame de flujo óptico

2.4.4 Pandas

Pandas (5) es una librería de código abierto desarrollada por Wes McKinney, fue diseñada para funcionar como una extensión de Numpy (2) esto facilita el trabajo cuando intervienen grandes cantidades de datos

Esta extensión para Python proporciona estructuras de datos rápidas, flexibles y diseñadas para ser intuitivas y fáciles de emplear.

Hoy en día el equipo que tiene el soporte de esta librería tiene el objetivo de convertirla en la herramienta de análisis, tratamiento y gestión de datos más potente de código abierto, cada día este objetivo está más cerca de ser cumplido.

Algunas de sus funcionalidades más destacables son:

- Mutabilidad de tamaño: se pueden insertar y eliminar las columnas del DataFrame.
- Reestructuración y segmentación de conjuntos de datos.
- Conjuntos de unión de datos.
- Herramientas de entrada/salidas robustas que pueden cargar datos desde archivos planos (CSV), archivos de Excel, bases de datos, etc.
- Alineación automática de datos.

```
1  # Import cars data
2  import pandas as pd
3  cars = pd.read_csv('cars.csv', index_col = 0)
4
5  # Print out observation for Japan
6  print(cars.iloc[2])
7
8  # Print out observations for Australia and Egypt
9  print(cars.loc[['AUS', 'EG']])
```

Figura 2.7: Ejemplo de carga de dataframe a partir de un .csv

2.4.5 Numpy

Numpy (2) es una librería de código abierto para Python que permite mayor soporte a la hora de realizar cálculos con vectores y matrices.

Fue desarrollada por Travis Oliphant en 2005, basada en Numeric, una librería anterior creada por Jim Hugunin. Oliphant desarrolló un código fácil de mantener y lo suficientemente eficiente y flexible para implementar las características novedosas de manejo de vectores que implementa Numarray.

La principal importancia de Numpy ante todo es su capacidad de proporcionar estructuras para empaquetar diferentes tipos de datos como vectores o matrices, las cuales son mucho más rápidas y eficientes que las proporcionadas de forma normal por Python.

Capítulo 3

3 Redes neuronales artificiales

3.1 Introducción

Si describiéramos una red neuronal de forma concisa, podríamos verla como un canon de la programación con inspiración en la vida, más concretamente, inspiradas en las conexiones de las neuronas del cerebro, y que nos permite procesar datos a partir de los cuales posibilitar que un sistema aprenda.

Ahora mismo las redes neuronales son una de las vertientes con mayor potencial a la hora de conseguir detectar humanos en vídeos o imágenes, y más concretamente poder identificar las acciones que éstos realizan. Además, gracias a su gran versatilidad, han demostrado ser capaces de obtener muy buenos resultados en otros campos, como el procesamiento de lenguaje natural, problemas de predicción, etc.

Aunque los modelos matemáticos inspirados en esta rama de la biología se introdujeron en 1943, por McCulloh y Pitts (43), no fue hasta unos años después, en 1950, cuando Rossemblat creó el Perceptrón (51), unidad básica de la que nacerían las redes neuronales. El denominado Perceptrón es un algoritmo binario de clasificación, que produce una salida de 1 o 0. Fue aquí donde por primera vez se introdujo el concepto de "pesos", un número real que permite reflejar la consideración que se le da a una determinada entrada con respecto a la salida.

3.2 La neurona biológica

Desde el punto de vista biológico una red neuronal es en definición un conjunto de más de cien mil millones de neuronas conectadas entre sí, que transmiten información de una a otra mediante una serie de impulsos eléctricos. Si colocáramos una de estas neuronas bajo el

microscopio podríamos ser capaces de diferenciar las partes que la componen, y de forma sencilla comprender las similitudes de una neurona real con una neurona artificial. Aunque las neuronas biológicas pueden variar en tamaño y forma, se componen de tres partes esenciales, figura 3.1:

- El **cuerpo o soma** de la neurona, contiene el núcleo celular, que mantiene la estructura de la neurona y proporciona energía para impulsar las actividades.
- Las **dendritas** son raíces fibrosas que se ramifican desde el cuerpo celular. Son las encargadas de recibir y procesar las señales eléctricas provenientes del axón de otra neurona.
- El **axón** es una estructura alargada unida al núcleo de la neurona, por la cual se envían los impulsos eléctricos una vez han salido del núcleo, y solo si estos impulsos superan un cierto valor umbral mínimo.

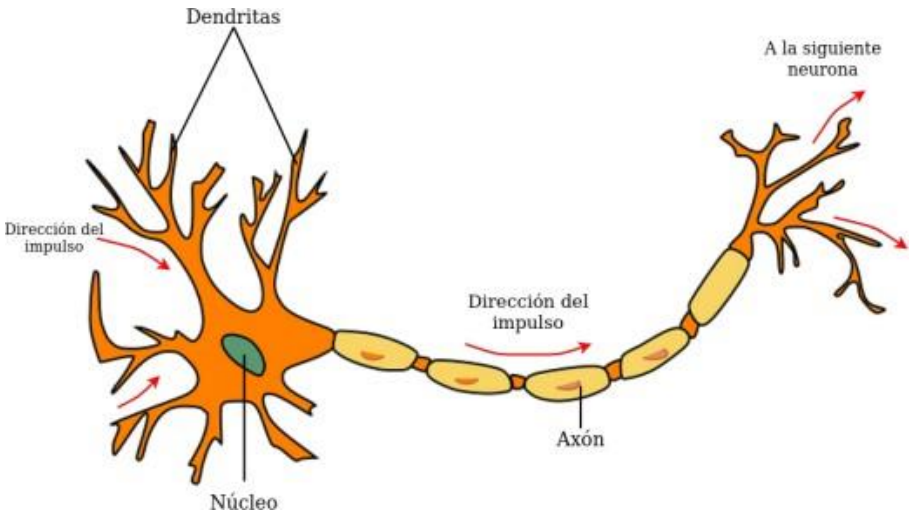


Figura 3.1: Neurona biológica

3.3 La neurona artificial

Como hemos visto la neurona artificial tiene cierta semejanza con la neurona biológica, no obstante, merece la pena mencionar todas las partes que la conforman. Una neurona artificial consta de las siguientes partes:

- Un número n de variables de entrada, siempre cumpliendo $n > 0$. Generalmente se representa en forma de vector de entradas, de forma $x = (x_1, x_2, x_3. x_n)$.

- Un **peso** asignado a cada variable de entrada, que define la importancia o "fuerza" de cada conexión. Estos pesos se representan en forma de vector de la forma $w = (w_1, w_2, w_3 . w_n)$.

- El **sesgo** o bias, b, es un parámetro que indica la viabilidad que tiene una neurona para activarse, de la misma forma que en las neuronas biológicas los impulsos eléctricos deben superar un determinado umbral para ser propagados.

- La función de propagación, es la aplicada en las variables de entrada, los pesos y en el sesgo tras la entrada, antes de la función de activación, $y = f\ (x_1 \ldots x_n, w_1 \ldots w_n, b)$. En su versión más básica, la perteneciente al perceptrón es de tipo lineal y se representa:

$$y = \sum_{i=0}^{n} w_i x_i + b \tag{3.1}$$

- La **función de activación**, será la responsable de determinar si se activa o no la salida de la neurona en función del valor resultante de la función de propagación, expresado de la forma $z = f_{act}(y)$. Dos funciones de activación muy utilizadas son la función sigmoide, que produce una salida entre 0 y 1, y la función de unidad lineal rectificada o RELU, ambas representadas respectivamente de la forma:

$$Sigmoide : f_{act}(y) = \frac{1}{1 - e^{-y}} \tag{3.2}$$

$$RELU : f_{act}(y) = \max(0, y) \tag{3.3}$$

La figura 3.2 muestra un esquema de una neurona artificial.

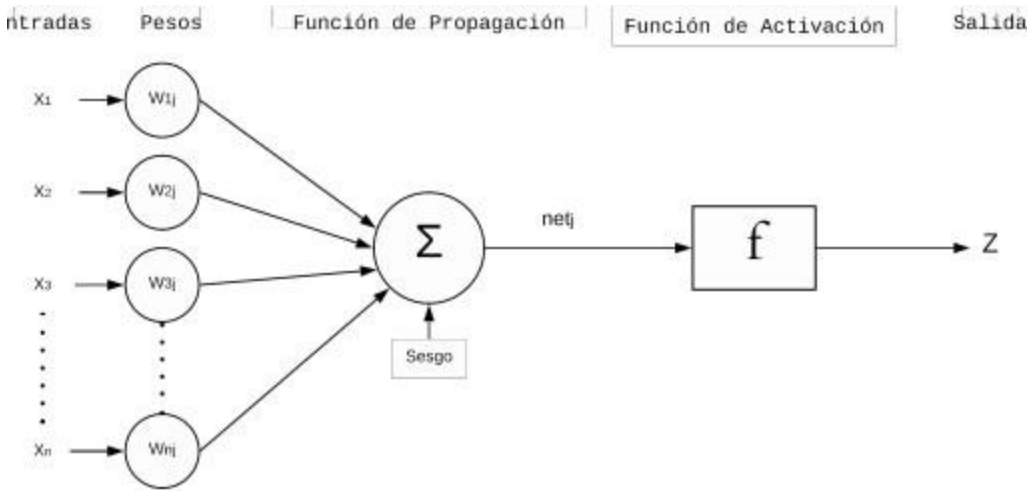


Figura 3.2: Esquema de la Neurona Artificial

3.4 Estructura de las redes neuronales

Si se quiere llevar a cabo cálculos verdaderamente complejos, con una sola neurona no es suficiente, se deberán utilizar muchas de ellas interconectadas entre sí. Por norma general, las neuronas se agrupan o constituyen conjuntos conformando las denominadas redes neuronales.

Entre las estructuras más utilizadas se tienen las redes recurrentes, y especialmente las redes *feed-forward*, que suponen el tipo de red más elemental. No obstante, otro tipo de red muy utilizada también en el ámbito de detección en imágenes y vídeos, bajo el paradigma de aprendizaje profundo, son las redes neuronales convolucionales en general y dentro de estas las redes residuales.

3.4.1 Redes de tipo feed-forward

Las redes feed-forward (hacia delante) reflejan la idea elemental que subyace en una red neuronal, las capas. En estas redes, encontramos agrupadas las neuronas en conjuntos denominados capas. Es en el momento en el que conectamos varias de estas capas, cuando hablamos de conformar una red neuronal. Entre estas capas distinguimos tres tipos: la capa de entrada, un número n de capas ocultas, y la capa de salida.

- **Capa de entrada**: esta capa no consta de operaciones matemáticas en su interior, sino que se corresponde con el conjunto de datos de entrada. Tiene tantas neuronas como elementos tiene el vector de entrada.
- **Capas ocultas**: contienen toda la lógica matemática intermedia de la red. Está conformada por neuronas ocultas, denominadas de esa manera porque las salidas de las neuronas de dichas capas se interconectan con otras neuronas, de forma que no pueden ser "vistas", desde fuera de la red.
- **Capa de salida**: es la capa final de la red, la que devuelve la predicción objetivo. Tiene tantas neuronas como número de clases posibles haya en el problema.

En este tipo de red, las neuronas se encuentran conectadas de capa en capa, hasta llegar a la capa de salida. Las neuronas de una capa, únicamente pueden estar conectadas a las neuronas de la capa inmediatamente siguiente.

En función del número de variables de entradas de la red, tendremos mayor o menor número de neuronas, siendo este número igual al de entradas. Por otro lado, el número de datos de salida dependerá de la cantidad de variables necesarias para representar la salida en cuestión. La figura 4.3 muestra un esquema de una red de tipo feed-fordward con sus correspondientes capas, de entrada y salida y dos capas ocultas.

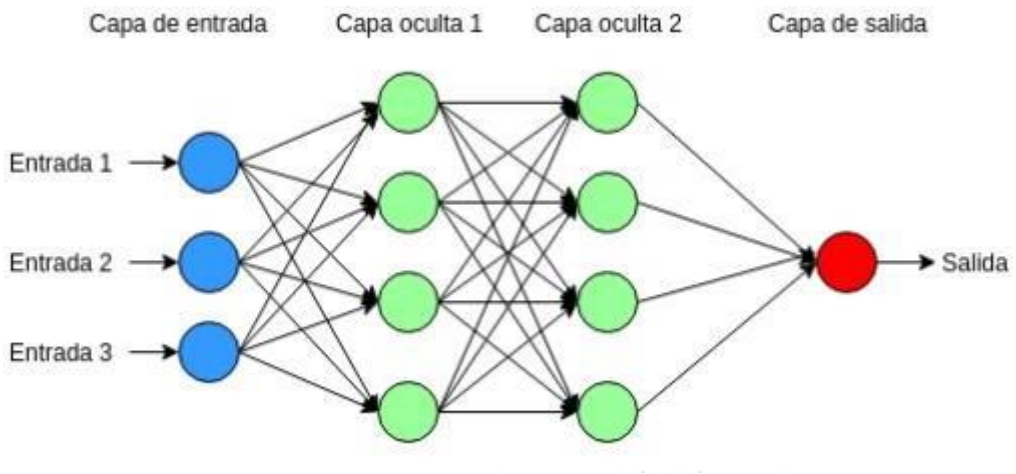


Figura 3.3: Esquema de una red feed-forward

podemos definir la salida de una capa n de red feed-forward mediante la siguiente fórmula matemática:

$$z_i^n = f_{activacion} \sum_{j=1}^{j} z_j^{n-1} w_{ij}^n + b_i^n \quad (3.4)$$

donde z_j^{n-1} es la salida de la neurona j de la capa $n-1$ de la red, la cual tiene un total de J neuronas. Por otra parte, tenemos w_{ij}^n, que es el peso de conexión de la neurona i de la capa n, con la neurona j de la capa $n-1$, y el sesgo b_i^n, de la neurona i de la capa n. Finalmente se define z^n, vector de salida equivalente a todas las posibles z_i^n de la capa n.

Una vez quedan claros estos conceptos principales, definiremos las redes profundas como aquellas redes con un gran número de capas ocultas, siendo éstas las utilizadas dentro del conocido paradigma del aprendizaje profundo.

3.4.2 Redes de tipo recurrente

Uno de los elementos que más caracterizan el razonamiento humano, es la existencia de una memoria, gracias al conocimiento que en ella almacenamos, podemos tomar unas u otras decisiones en el futuro. Por ejemplo, cuando leemos un libro somos capaces de comprender los significados de ciertas expresiones gracias al contexto que forman las palabras anteriores que hemos memorizado. Basándose en esa idea surgieron las redes neuronales recurrentes o RNN (Recurrent Neural Networks) (28), que son una de las arquitecturas más cercanas al razonamiento humano.

Las redes RNN, al igual que las demás redes, aprenden durante el proceso de entrenamiento, pero, además, son capaces de tener en cuenta lo aprendido de entradas anteriores, alterando así la salida. Esta idea es la que otorga a la red el concepto de temporalidad, permitiendo a las redes tener memoria. Precisamente este concepto temporal es el que permite establecer la conexión entre las secuencias de vídeo, que llevan implícita la vertiente temporal, y este tipo de redes.

Esta información adicional que recibe la red proviene del denominado vector de estado oculto o S, que es el que representa ese contexto previo. Una vez generada la salida en función de la entrada y el estado oculto, el vector de estado oculto es actualizado en base a dicha entrada, quedando preparado para usarse en la siguiente entrada, figura 3.4.

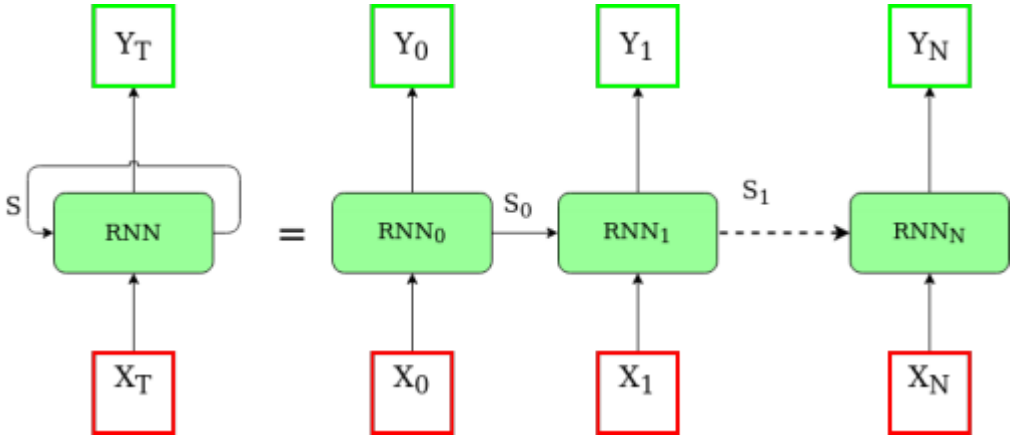


Figura 4.4: Arquitectura básica RNN

3.4.3 Redes de tipo residual

Este tipo de estructura de red neuronal surgieron en 2016, cuando He et al.(27) se plantearon la creciente dificultad para entrenar redes profundas. Es precisamente en la búsqueda de una mayor facilidad de entrenamiento o mayor facilidad en el proceso de optimización, donde surge el nuevo paradigma de las redes residuales.

En ciertos casos puede ocurrir que cuando la profundidad de una red neuronal aumenta, suele tender a saturarse la precisión, sin necesariamente ser por culpa del sobreajuste. La idea subyacente detrás de las redes residuales o ResNet, tiene como aporte principal, la introducción de un" cortocircuito", que permiten realizar conexiones que saltan una o más capas de la red.

En lugar de esperar que cada cierto número de capas apiladas, éstas encajen o se acoplen directamente hacia una proyección subyacente deseada, se deja explícitamente que estas capas se ajusten a una proyección residual. Si denominamos a la proyección subyacente deseada de la forma $H(x)$, se permite que las capas no lineales apiladas se ajusten a una proyección $F(x) = H(x) - x$. De esta manera, la proyección original se planteará de la forma $F(x) + x$, bajo la hipótesis de

que, si una proyección de identidad es óptima, resultaría más fácil llevar el residuo $F(x)$ a cero, que ajustar una proyección de identidad mediante una pila de capas no lineales. Es decir, es más fácil encontrar una solución tal que $F(x) = 0$, en lugar de $F(x) = x$, usando una pila de capas no lineales como función.

La función $F(x)$ es la que denominamos residuo. Bajo esta formulación se argumenta que las capas apiladas no deberían empeorar el rendimiento de la red, porque simplemente se pueden apilar asignaciones de identidad, es decir, una capa que no hace nada. En cualquier caso, conviene dejar claro que la red posee tanto las capas apiladas no lineales, que producen la salida $F(x)$, como el cortocircuito que produce la identidad x, figura 4.5.

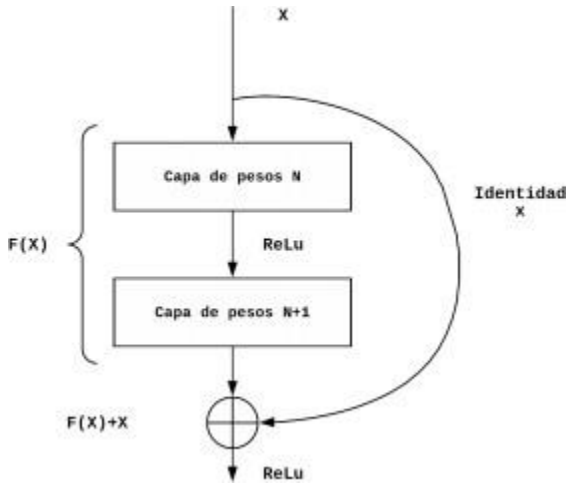


Figura 4.5: Esquema de un bloque de red residual

3.5 Tipos de aprendizaje en las redes neuronales

De igual manera que pueden distinguirse diferentes variantes de red neuronales en base a su estructura, otra forma de clasificarlas es respecto al tipo de aprendizaje que utilicen.

Existen un total de 4 vertientes de aprendizaje con las que un algoritmo de aprendizaje automático puede aprender: aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semi-supervisado o híbrido y aprendizaje por refuerzo. De entre los anteriores se pueden definir

dos tipos claramente diferentes, aquellos en los que se necesita conocimiento de los datos con antelación y aquel que precisamente se caracterizada por su ausencia.

3.5.1 Aprendizaje supervisado

En las redes neuronales que hacen uso de este tipo de aprendizaje, se parte de un conocimiento previo que debe ser entregado a la red. El objetivo es que mediante un conjunto de datos de entrada y la salida que debería devolver la red, se pueda inferir una función capaz de establecer una proyección de las entradas con las salidas de la manera más precisa posible. Estos datos de entrenamiento, son a menudo denominados tuplas (X, Y), siendo X los datos de entrada e Y los datos de salida o etiquetas. La figura 4.6 muestra un esquema simplificado del aprendizaje supervisado, desde la entrada a la salida pasando por el proceso de entrenamiento e incluyendo los datos de test para validación del modelo de red.

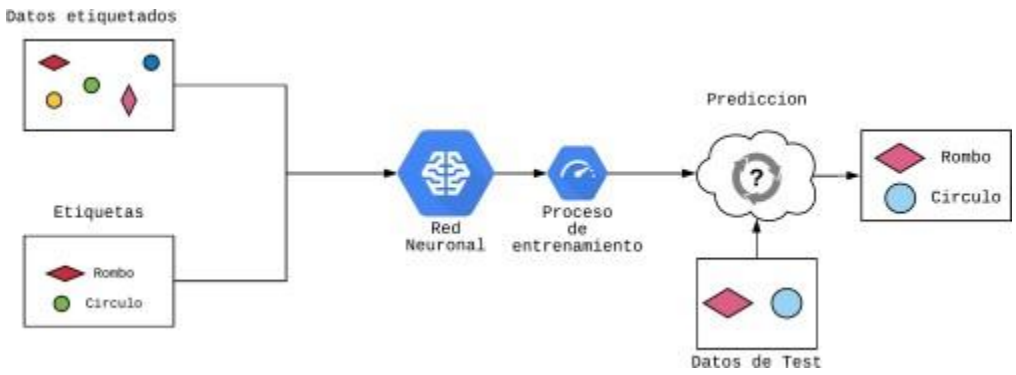


Figura 4.6: Esquema simplificado del aprendizaje supervisado

Durante el proceso de entrenamiento, la red neuronal ajustará sus pesos internos para que la salida de la red sea cada vez más similar a la salida deseada. Llevando a cabo una serie de iteraciones, hará uso de métodos capaces de calcular el grado de error o pérdida que tiene la red en cada momento, con el objetivo de disminuir dicho error mediante el citado ajuste de los pesos

3.5.2 Aprendizaje no supervisado

A diferencia del aprendizaje supervisado, en este tipo de aprendizaje se presenta una serie de datos de entrada, pero no los correspondientes datos de salida, es decir, no existe conocimiento previo. No recibe como entradas tuplas (X, Y), sino solo X.

El objetivo del aprendizaje supervisado es el de modelizar y distribuir los datos de manera que permita extraer conclusiones sobre ellos. Permite descubrir estructuras interesantes en los datos y estimar funciones de densidad de probabilidad de aparición de los patrones en el conjunto de datos.

De forma general, los algoritmos que utilizan este tipo de aprendizaje pueden ser agrupados en algoritmos de clustering o agrupación y algoritmos de asociación.

3.5.3 Aprendizaje semi-supervisado o hibrido

El aprendizaje de este tipo se encuentra en un punto intermedio entre el aprendizaje supervisado y el no supervisado. Se tienen, por un lado, tantos datos etiquetados de la forma de tupla (X, Y) y por otros datos sin etiquetar de la forma X. Generalmente utiliza una pequeña cantidad de datos etiquetados y un gran número de datos no etiquetados.

La razón por la que se lleva a cabo este tipo de aprendizaje es porque ha demostrado en algunos casos mejorar de manera notable la precisión del aprendizaje. No obstante, otro de los puntos a favor de este tipo de aprendizaje es que permite usar pocos ejemplos etiquetados para entrenar. Conseguir una gran cantidad de datos etiquetados para resolver un cierto tipo de tarea suele requerir en ciertos casos de la presencia de un elemento humano que clasifique manualmente los datos. Muchas veces, un proceso de etiquetado de este tipo lleva acarreado un gran coste, llegando a hacer que conseguir ciertos conjuntos de datos de entrenamiento sea inviable, mientras que conseguir datos no etiquetados es relativamente fácil. En el tipo de situaciones en las que se presente un caso así, el aprendizaje semi supervisado podría resultar de gran utilidad.

3.5.4 Aprendizaje por refuerzo

En el caso del aprendizaje por refuerzo, se presenta un escenario, que al igual que el híbrido, está situado en un punto intermedio entre el aprendizaje supervisado y el no supervisado. Sin embargo, en este tipo de aprendizaje no se proporciona a la red la respuesta deseada, sino que se le indica el error que comete de forma global.

Este tipo de aprendizaje está basado en el conductismo, una rama de la psicología dedicada al estudio de la conducta en base a estímulos y recompensas. Si abstraemos este concepto al aprendizaje automático, descubrimos que lo que se busca definir es el comportamiento que debe tener nuestra red neuronal para maximizar algún tipo de recompensa. La figura 4.7 muestra un esquema de funcionamiento del aprendizaje por refuerzo, donde se muestra cómo la red es recompensada cuando la acción realizada se asume como correcta.

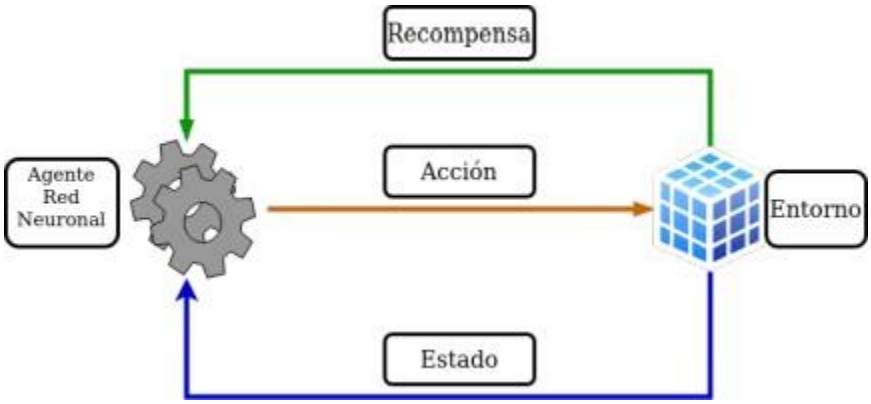


Figura 3.7: Esquema simplificado del aprendizaje reforzado

3.6 Métodos de aprendizaje

La teoría a subyacente en las redes neuronales tiene una gran extensión y repercusión, de forma que en este apartado se describen brevemente los métodos de aprendizaje más relevantes utilizados en las redes neuronales y cuyas ideas resultan de gran importancia a la hora de entender el funcionamiento interno de las redes neuronales durante el proceso de entrenamiento. El

siguiente paso después de definir la estructura de la red neuronal, es encontrar el valor de los diferentes parámetros que la componen, buscando en todo momento alcanzar las mayores precisiones del modelo de red en cuestión.

3.6.1 Función de coste

Esta función, también conocida como función de pérdida, tiene como objetivo medir cómo de equivocados están siendo los resultados de un modelo de aprendizaje automático. Simplificando su funcionamiento, puede definirse como la comprobación de la diferencia existente entre las respuestas dadas por el modelo, con respecto a las respuestas correctas que debería haber generado. De forma que permite realizar una medición sobre cómo de bien están ajustados los parámetros de dicho modelo con respecto al conjunto de datos de entrenamiento.

A la hora de abordar un problema, resulta prioritario elegir una función de coste adecuada, ya que los parámetros del modelo de redes neuronales serán ajustados con el objetivo de minimizar dicha función.

De entre las funciones de coste más utilizadas destaca *Mean Squared Error* (MSE) o error cuadrático medio. Simplificada de la forma:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(x_i - y_i)^2 \qquad (3.5)$$

Siendo xi el valor de salida esperado para una determinada entrada, y_i el valor dado por la red para esa misma entrada y M el número de entradas.

Otra función de coste a destacar es la función Cross-Entropy (CE) o entropía-cruzada, usada generalmente en problemas de clasificación. Esta función es capaz de medir el desempeño de un modelo de aprendizaje automático entre 0 y 1, siendo el 0 la pérdida ideal de un modelo perfecto. Se definen dos variantes de la función de entropía-cruzada, por un lado, la binaria, de la forma:

$$BCE = -(ylog(p) + (1 - y)\log(1 - p) \quad (3.6)$$

donde p es la probabilidad dada para la predicción, e y es el indicador binario. Por otro lado, la categórica, escrita de la forma:

$$CCE = \sum_{C=1}^{M} y_{o,c} \log(p_{o,c}) \quad (3.7)$$

siendo $y_{o,c} = 1$ si la clase c es la predicción correcta para la entrada o; $p_{o,c}$ es la probabilidad que ha predicho el modelo para la clase c para la entrada o.

3.6.2 Descenso del gradiente

El objetivo de utilizar un algoritmo de descenso de gradiente es el de encontrar los pesos y los sesgos óptimos, de manera que minimicen el error calculado mediante la función de coste. Para lograrlo se desplaza iterativamente siguiendo la dirección de la pendiente más pronunciada, de acuerdo a lo definido por el negativo del gradiente. Al desplazarse repetidamente en esa dirección lleva a cabo el "descenso de gradiente", logrando en cada una de estas iteraciones disminuir el valor de la función de coste, llegando en algún momento a lo que de forma intuitiva denominaremos el mínimo.

La manera en que este método funciona puede dividirse en las siguientes etapas:

1. La primera etapa consiste en inicializar la función que se quiere optimizar. Para ello suele escogerse un punto aleatorio en el dominio, $x_0 = (x^0{}_1, \ldots, x^0{}_n)$.
2. El segundo paso es calcular el gradiente de la función de coste, expresada como f, a partir del punto calculado en la iteración previa, expresado como x_{t-1}:

$$\nabla(fx_{t-1}) = \frac{\partial f}{\partial x_1^{t-1}}, \frac{\partial f}{\partial x_2^{t-1}}, \frac{\partial f}{\partial x_n^{t-1}} \tag{3.8}$$

3. El tercer paso consiste en calcular el próximo punto en el que el valor de la función f sea menor. Para ello se hace uso de la siguiente ecuación:

$$x_t = x_{t-1} - n\nabla(f(x_{t-1})) \tag{3.9}$$

en la cual η representa la denominada tasa de aprendizaje, que refleja cuánto se avanza en cada iteración.

4. El último paso es repetir los pasos anteriores tantas veces como sea necesario, disminuyendo la función de coste hasta alcanzar un mínimo, o bien si se llega a un número máximo de iteraciones previamente estipulado.

En resumen, cuando aplicamos de forma iterativa el proceso anterior, lo que se consigue es "bajar una pendiente" y encontrar un punto en el cual la función de coste genere un valor lo suficientemente bueno.

Cuando se aplica este método con el objetivo de minimizar el valor de la función de coste de una red neuronal, representada de la forma C(W,b), siendo W los pesos y b los sesgos, cada una de las iteraciones del proceso es denominada época o *epoch*. De esta manera, la actualización de los parámetros de la red puede expresarse de la siguiente forma:

$$w_{k+1} = w_{k_t} - n\frac{\partial C}{\partial w_{k_t}} \tag{3.10}$$

$$b_{k+1} = b_{k_t} - n\frac{\partial C}{\partial b_{k_t}} \tag{3.11}$$

3.6.3 Descenso estocástico de gradiente

Este método es una variante simplificada del algoritmo de descenso de gradiente. Destaca que, en él, se escogen de forma aleatoria un pequeño conjunto de elementos para llevar a cabo la actualización de los pesos, en vez de usar todos los datos.

El método se estructura según los siguientes pasos:

1. Dividir de forma aleatoria el conjunto de datos en un numero M de subconjuntos de igual tamaño. Estos subconjuntos reciben la denominación de mini-batches.
2. Realizar una aproximación de la función de coste para cada mini-batch, en lugar de calcular el coste total, de la siguiente forma:

$$C(W,b) = \frac{1}{M}\sum_{j=1}^{M} C_j(W,b) \tag{3.12}$$

   en la cual $C_j(W,b)$ representa el coste de un mini-batch
3. Al utilizar la ecuación aproximada anterior, se calcula el gradiente y se actualizan los parámetros de la red de la siguiente manera:

$$w_{k_t+1} = w_{k_t} - n\frac{1}{M}\sum_{j=1}^{M}\frac{\partial C}{\partial w_{k_t}} \tag{3.13}$$

$$b_{k_t+1} = b_{k_t} - n\frac{1}{M}\sum_{j=1}^{M}\frac{\partial C}{\partial b_{k_t}} \tag{3.14}$$

4. Repetir los pasos anteriores tantas veces como sea necesario, disminuyendo la función de coste hasta alcanzar un mínimo, o bien si se llega a un número máximo de iteraciones previamente establecido.

La principal mejora que supone este método frente al descenso de gradiente estándar, es que el aprendizaje se ve acelerado debido a la mayor tasa de actualización de los parámetros. Al trabajar con subconjuntos aleatorios es más complicado converger en un mínimo local, facilitando llegar al mínimo global.

3.6.4 Propagación hacia atrás

El algoritmo de propagación hacia atrás o backpropagation en inglés, es uno de los métodos más eficientes y potentes a la hora de realizar el cálculo de los gradientes de una función de coste, en relación a los parámetros que tiene la red. Aunque la idea detrás de este algoritmo existía desde los años 70, fue presentado formalmente en 1985 por Rumelhart et al.(53).

En el algoritmo de retropropagación se tiene una función de coste (p.ej. MSE o BCE) que se busca minimizar, de forma que el algoritmo de backpropagation se encarga de modificar y actualizar los pesos y los sesgos de la red en cada iteración o epoch.

Dentro del proceso de propagación de la información, se distinguen dos fases muy importantes, la fase forward, en la que las señales de información fluyen de forma natural desde las entradas a las salidas y la fase backward, en la que una vez obtenidas las salidas se procede a realizar la modificación de los pesos y sesgos, pero esta vez en sentido contrario.

El funcionamiento de este método puede expresarse en forma de tres pasos:

1. **Fase de inicio**: En el caso de partir sin datos iniciales sobre la red, los pesos y sesgos de la misma deben ser inicializados de forma aleatoria y seguidamente prepararse un patrón de entradas, generalmente seleccionado de forma aleatoria, correspondiente a una época.
2. **Fase de forward**: En esta fase se lleva a cabo el proceso de propagación hacia delante. Partiendo de una parte de los datos correspondientes a la época n, se prepara como entrada a la red el vector x(n). Se realiza entonces el proceso de forward, expresado en la ecuación:

$$v_j^{(L)}(n) = \sum_{i=0}^{m_0} w_{j,i}^{(L)}(n) y_i^{(L-1)}(n) \tag{3.15}$$

Donde $v_j^{(L)}$ (n) representa el campo local inducido, siendo *j* el número de neurona y *L* el número de la capa. La expresión $y_i^{(L-1)}$ simboliza el valor de salida de la neurona *i,* de la capa anterior a *L* durante la época n. Finalmente $w_{j,i}^{(L)}$ representa el peso existente

entre la conexión de la neurona j de la capa L y la neurona i de la capa anterior. Una vez realizado este paso se podrá deducir el error, por medio de una función de coste. Usando una forma simplificada de lo que es la función de coste, la operación quedaría de la forma:

Siendo$e_i(n)$ el i-ésimo termino esperado para la salida $e(n)$, y $pj\ (n)$ el resultado dado por la salida de la neurona j.

$$error_i(n) = e_i(n) - p_j(n) \tag{3.16}$$

3. **Fase de backward**: Tras calcularse los gradientes locales correspondientes mediante el uso del método de descenso de gradiente, los pesos deben ser ajustados mediante esta operación:

$$w_{j,i}^{(L)}(n+1) = w_{j,i}^{(L)}(n) + \alpha[w_{j,i}^{(L)}(n-1) + \eta\delta_j^L(n)y_i^{(L-1)} \tag{3.17}$$

Siendo η la tasa de aprendizaje o learning rate$\delta_j^L(n)$ el gradiente local previamente calculado. El *learning* rate está relacionado con la dirección que lleva el método de *backpropagation* mediante el descenso de gradiente. El símbolo α hace referencia a la constante de *momentum*, que ayuda a reducir las oscilaciones que causa el hecho de que los pesos varíen.

Los pasos de *forward* y *backward* se repetirán tantas veces como sea necesario hasta alcanzar un mínimo, o bien si se llega a un número máximo de iteraciones previamente estipulado. Dado que la convergencia en un mínimo rara vez ocurre, es importante estipular cuando debe parar, p.ej. alcanzar un determinado valor de coste medio.

3.7 Medidas de prevención de sobreajuste

Cuando se entrena una red neuronal se busca que sea capaz de realizar la labor para la que fue entrenada, a partir de lo aprendido gracias al conjunto de datos de entrenamiento. Sin embargo,

la red también debe tener la capacidad de enfrentarse a casos distintos a los que fue entrenada, es decir, deben poder generalizar.

Si una red neuronal se ajusta de forma excesiva a los datos con los que fue entrenada, aprenderá de forma muy detallada las características de éstos, perdiendo como consecuencia la capacidad de generalizar. Este fenómeno es el denominado *overfitting* o sobreajuste. Se trata de un problema bastante habitual, por lo que se han desarrollado muchas medidas que puedan evitar este comportamiento.

Una de las formas más habituales para comprobarlo es dividir el conjunto de datos de entrenamiento en dos partes, una para entrenamiento y otra para validación. La red aprende con los datos de entrenamiento a lo largo de cada *epoch*, modificando los parámetros internos de sesgos y pesos, buscando minimizar el valor de la función de coste. El conjunto de validación no altera los parámetros de la red, pero sí permite comprobar el error.

Uno de los métodos más utilizados para evitar el sobreajuste de un modelo de redes neuronales es el llamado método de *dropout*.

El método de *dropout* soluciona el problema del *overfitting* mediante la eliminación de neuronas de forma aleatoria dentro de una red profunda. Esto quiere decir, que el aporte de esas neuronas a la activación de las demás queda anulado temporalmente durante el paso de forward y que además estas neuronas no se vean afectadas por la actualización de parámetros durante el paso *backward*.

En el proceso de aprendizaje normal de una red neuronal profunda, el paso de backward genera relaciones de dependencia de una neurona con otra, beneficiando el resultado con los datos de entrenamiento, pero no con los datos distintos. El uso del método de dropout de forma aleatoria acaba con estas relaciones, haciendo que ciertas neuronas no sean tan importantes.

Se ha podido comprobar que el uso de este método ha mejorado el rendimiento de los modelos de redes neuronales en diversos campos de la inteligencia artificial, como, por ejemplo, la visión artificial, la clasificación de textos o los análisis de voz.

La figura 3.8 muestra un esquema gráfico de aplicación del método de dropout de forma que en la parte superior se muestra un esquema general con todas las neuronas conectadas, mientras que en la parte inferior algunas de ellas han sido desconectadas.

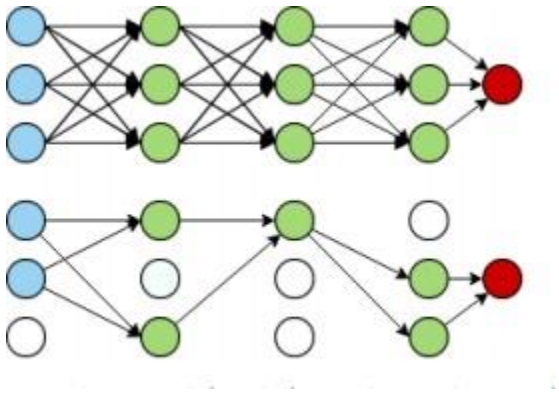


Figura 3.8: Red original y red con dropout

3.8 Redes neuronales convolucionales

Cuando se habla del término *deep learning* o aprendizaje profundo, el primer tipo de red que sale a relucir son las redes convolucionales. Este tipo de red neuronal, similar a las redes *feed-forward*, fundamentan su principal diferencia en la intervención de un tipo de capa denominada convolucional, que da nombre a la red. De esta forma puede decirse que las CNN (Convolutional Neural Networks) son un tipo de red neuronal que utiliza la operación de convolución, en vez de multiplicar matrices, en por lo menos una de las capas.

La figura 3.9 muestra un esquema general de un modelo CNN con las mencionadas capas, incluyendo además funciones de activación de tipo RELU, y capas de aplanamiento (Flatten) que pasan de volúmenes de datos a vectores o capas totalmente conectadas (Fully Connected) donde todas las neuronas están completamente conectadas entre capas.

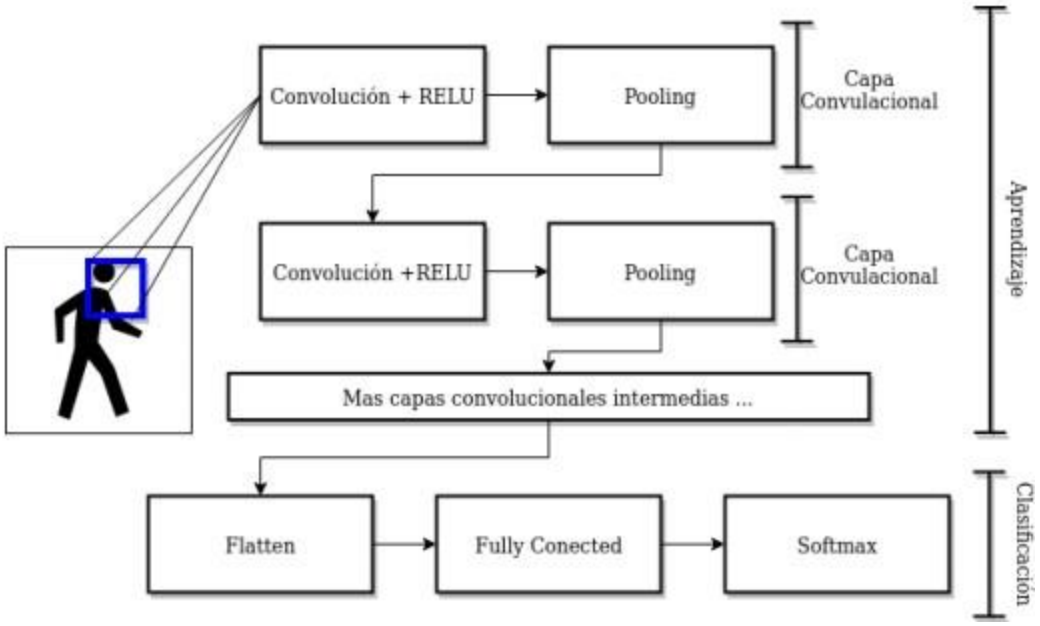


Figura 3.9 Arquitectura de una red CNN

Este tipo de red divide y procesa los datos en partes más pequeñas, para más tarde combinar esta información en algunas de las capas más profundas, permitiendo que las neuronas sean capaces de especializarse en ciertas regiones de los datos.

Han demostrado ser especialmente efectivas en problemas relacionados con el tratamiento de imágenes, la clasificación de textos o incluso el reconocimiento de voz.

Aunque la capa de convolución es la más representativa en una CNN, existen también otros tipos, destacando la llamada capa de *pooling* o agrupación, típicamente utilizada para realizar agrupaciones de píxeles sobre los resultados obtenidos tras la capa de convolución, y la capa de *softmax*, situada justo antes de la capa de salida donde se obtiene el resultado.

3.8.1 Capa de convolución

La operación de convolución parte de una serie de operaciones de sumas y productos entre la capa de entrada y el *kernel* (núcleo de convolución), permitiendo obtener un mapa de características de la imagen.

La operación de convolución puede expresarse de la forma:

$$s(t) = \int (x * w)(t) \tag{3.18}$$

El primer argumento, la x en la operación de convolución, es conocido como la entrada o *input*, y el segundo, la w, es el anteriormente nombrado *kernel*. Por otro lado, s, es el mapa de características extraído de la imagen, también llamado *feature map* (mapa de características).

A la hora de utilizar esta operación durante una tarea de aprendizaje automático, la entrada es un vector o una matriz multidimensional de información y el *kernel*, otro vector o matriz multidimensional, compuesto de un conjunto de parámetros capaces de ajustarse durante la fase de aprendizaje. En el caso de una imagen I, la estructura del input es una estructura 2-D, que es procesada por un *kernel* K, también de dos dimensiones. Quedando de la forma:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.19}$$

Estas capas aplican en paralelo varias operaciones de convolución en distintos píxeles (i, j). Para comprender cómo se realiza este paso, lo mejor es pensar en la aplicación del *kernel* de la misma forma que lo haría una ventana deslizante, desplazándose por toda la imagen y calculando el mapa de características correspondiente en cada movimiento, figura 3.10.

Aunque está bastante generalizado trabajar con convoluciones 2-D, también existen de tipo 3-D, de manera que el *kernel* tendrá en este caso la forma de un cuboide, este es un caso normal cuando se trabaja con imágenes RGB, en las cuales existe una tercera dimensión apilando los tres canales espectrales en el modelo de color RGB. Esto puede generalizarse a un número de dimensiones N, como es por ejemplo el caso de algunas de las redes convolucionales utilizadas en reconocimiento de vídeo, en las que existe una dimensión más para el tiempo

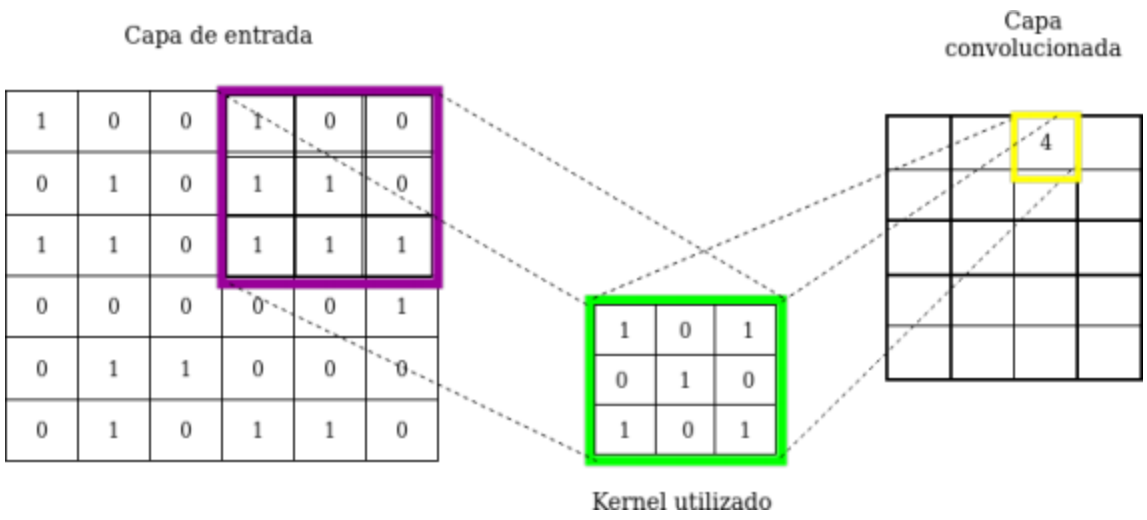


Figura 3.10: Ejemplo de un paso de convolución 2D sobre una imagen binaria usando kernel 3x3

3.8.2 Capa de pooling

Uno de las capas más utilizadas dentro del ámbito de las CNN, aparte de la capa convolucional, es la capa de *pooling* o agrupamiento. Generalmente esta capa se utiliza para reducir el tamaño de representación de los datos, permitiendo así aumentar la velocidad de cómputo. Para llevar a cabo una operación de *pooling*, primero debe definirse el tamaño del filtro que se aplicará. Posteriormente la imagen de entrada o matriz de datos, se divide en secciones del mismo tamaño que el filtro. Sobre cada una de estas secciones se lleva a cabo la operación de *pooling*, generalmente existen dos variantes: *max-pooling* y *average-pooling*. La variante de *max-pooling* comprueba los valores de los píxeles en cada sección y elige el máximo, en cambio, *average-pooling* devuelve la media numérica de los valores en la ventana o filtro, figura 3.11.

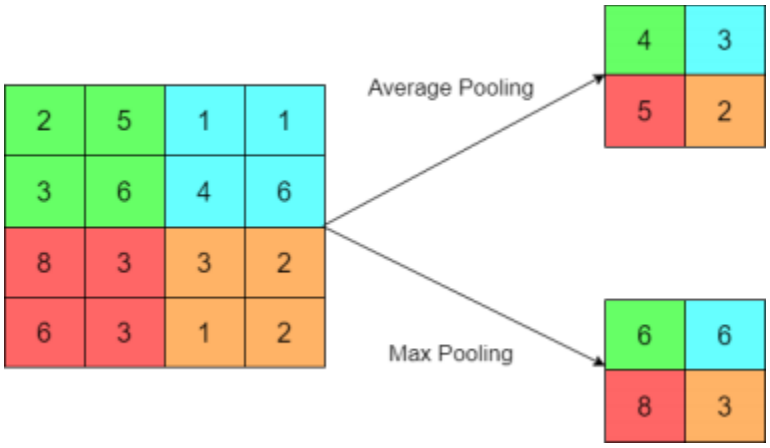


Figura 3.11: Ejemplo de *pooling* utilizando un filtro 2x2

3.8.3 Capa softmax

La capa *softmax* se introduce con el objetivo de aplicar sobre los datos la denominada función exponencial normalizada, capaz de "traducir" un vector M-dimensional, x, conformado por valores reales arbitrarios, en un vector M-dimensional, $S(x)$, formado por valores dentro del rango [0, 1], siendo S la función *softmax*.

$$S(j) = \frac{e^{z_j}}{\sum_{i=1}^{M} e^{z_i}} \tag{3.18}$$

$$z_j = w_j^T * x \tag{3.18}$$

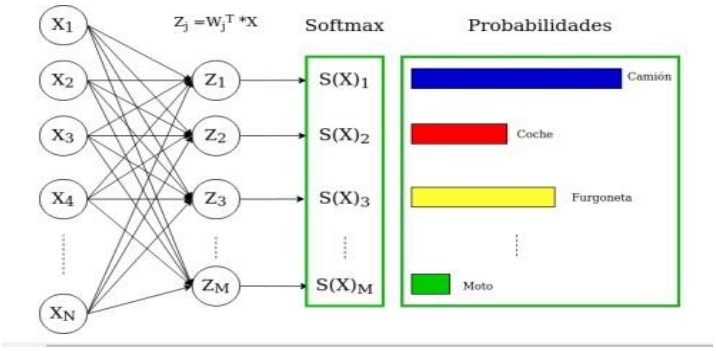


Figura 3.12: Ejemplo de capa *sofmax*

Capítulo 4

4 Detector de estados de maduración

4.1 Introducción

Como se explica en capítulos anteriores, uno de los objetivos finales de este proyecto es el diseño e implementación de un sistema que, mediante la aplicación de distintas técnicas de visión artificial y aprendizaje profundo, permita realizar labores de cosecha selectica, siendo capaz de detectar los diferentes estados de maduración del café, localizarlos y extraerlos.

La labor de un sistema con estas especificaciones, facilita las labores de cosecha que tradicionalmente se realizan de manera manual por recolectores, permitiendo maximizar la eficiencia de la cosecha.

Por tanto, según lo anterior, no hay duda de que la implementación se sistemas automáticos de cosecha selectiva en diversos escenarios facilita las tareas de cosecha según su finalidad. En las secciones que conforman este capítulo se expone el diseño del funcionamiento general del sistema y de los módulos que lo conforman.

4.1.1 Funcionamiento general del sistema

En el sistema desarrollado, la vertiente es el aprendizaje profundo en donde se abordan problemas de localización y clasificación, figura 4.1:

1. **Módulo de detección de frutos**: Detecta los frutos presentes en la imagen con base en una serie de umbrales de confianza, señalándolos mediante cajas delimitadoras.

2. **Módulo de clasificación del sistema:** clasifica el fruto en sus diferentes estados de maduración dependiendo de umbrales de precisión y de la información recibida del módulo anterior

Figura 4.1: Diagrama que representa el flujo de funcionamiento del sistema

4.2 Módulo de detección y clasificación

Llevando a cabo una división de los procesos presentes en este módulo, se pueden encontrar las siguientes etapas y subetapas

1. **Procesamiento de la imagen**
2. **Generación de detecciones y clasificación**

a. Inferencia sobre los modelos propuestos
b. Procesamiento de la salida de la red
c. Pre-filtrado de las detecciones encontradas

3. Filtrado por niveles de confianza

a. Almacenamiento de las detecciones

En primer lugar, el sistema accede a las imágenes de entrada, sobre el cual se quiere realizar la detección inteligente. En caso de que la imagen no cuente con el tamaño adecuado que recibe la red se re-escala la imagen, esto se realiza con la librería OpenCV.

Posteriormente cuando se haya asegurado que se cumplen los requisitos anteriores, éste es preprocesado y transformado en formato *blob*. Un blob es potencialmente una colección de imágenes de igual tamaño, mismas dimensiones espaciales y mismo número de canales de color, que deben ser procesadas de la misma forma.

En este caso, el preprocesado llevado a cabo sobre la imagen, se hace con el objetivo de generar una entrada válida para el modelo de red neuronal utilizado. En concreto sobre la imagen se han aplicado las siguientes operaciones de preprocesado:

- Redimensión de la imagen a 416 x 416, siendo éste el tamaño esperado como entrada por red neuronal
- Intercambio de los canales de color R y B, pasando de BGR a RGB.
- Escalado del espacio de la imagen en un rango particular, concretamente $\sigma = 255.0$

Una vez la imagen ha sido preprocesada, es proporcionada como entrada al modelo de detección de objetos.

De entre los métodos de detección de objetos, se consideran aspirantes a formar parte del sistema de cosecha selectiva, únicamente a:

- Yolo V3
- Yolo V5
- Mobilenet V2

Una vez procesado un paso sobre la red, devolverá una salida en forma de vector cuyos elementos presentas la siguiente forma

$$(cordenada_x, cordenada_y, ancho, alto, probabilidades) \qquad (4.11)$$

Donde los primeros parámetros hacen referencia a las coordenadas (x, y) de la caja delimitadora de un objeto, los siguientes son las dimensiones de la caja. El ultimo parámetro es la confianza.

Una vez que estas predicciones han sido obtenidas, son procesadas de acuerdo a los siguientes pasos:

1. Extracción del índice de mayor probabilidad de acuerdo al umbral de confianza previamente determinado.
2. Almacenamiento de la probabilidad obtenida en el paso anterior.
3. Generación de la caja delimitadora.

Cuando el proceso de detección ha finalizado y transformado a un formato de caja delimitador, el sistema puede proceder a aplicar una función de supresión no máxima.

Al aplicar la función NMS, se está llevando a cabo no solo un filtrado por capas de confianza de las predicciones que pasaron las etapas, sino que además elimina los cuadros delimitadores que se superponen significativamente, y de esta manera mantener los más precisos.

Finalmente, las cajas delimitadoras definitivas son filtradas, permitiendo únicamente aquellas clasificadas como verde, maduro, semi-maduro y sobre-maduro.

Figura 4.2: ejemplo de detección

Capítulo 5

5 Configuración sistema acústico

5.1 Introducción

Un sistema acústico que el objeto de estudio principal de este proyecto es un método no invasivo que busca solucionar los principales problemas actuales de los métodos invasivos que acortan la vida útil de un cultivo

uno de los objetivos finales de este proyecto es el estudio la influencia de un sistema acústico sobre el sistema fruto pedúnculo y de cómo la variación de frecuencias puede diferenciar los diferentes estados de maduración.

Por consiguiente, también se evaluará también el total de energía suficiente para alcanzar la frecuencia de resonancia de los frutos maduros y de la viabilidad de los sistemas acústicos actuales en cuanto a transmisión de energía

En este caso, se ha optado por realizar una serie de pruebas de funcionamiento con diferentes imágenes y escenarios. Comparando los datos obtenidos por el sistema con los que realmente están en las imágenes es posible comprobar la precisión del sistema. De esta manera las métricas utilizadas a la hora de comprobar la fortaleza del diseño son:

- mAP(mean average precision)
- precisión
- exhaustividad

Concretamente, la máquina donde se ha realizado el entrenamiento posee las siguientes especificaciones: máquina virtual de Google colab Tesla P100 16GB de memoria. Por tanto, estas características permiten establecer un límite inferior por debajo del cual las horas de entrenamiento del sistema tomarían varios días.

5.2 Dispositivos

5.2.1 TURBOSOUND IQ15 CABINA ACTIVA 15" TURBOSOUND

5.2.2 micrófono de medición Behringer ecm8000

5.2.3 Interfaz De Audio Usb Presonus Studio 24c

5.2.4 sensor piezo eléctrico

5.2.5 sonómetro uni-t ut353

5.3 Etapa de calibración

Para llevar a cabo la etapa de calibración y obtener un factor de calibración es necesario una referencia física la cual debe ser similar a la presión calculada.

Para llevar a cabo este algoritmo es necesario:

- Un tono de calibración grabado desde el micrófono que desea calibrar
- La frecuencia de muestreo utilizada por la tarjeta de sonido para la conversión de AD.
- El volumen conocido, generalmente determinado usando un medidor físico SPL
- La ponderación de frecuencia utilizada por el medidor SPL físico
- La presión atmosférica en el lugar de registro

Como se evidencia el diagrama 5.3.1 indica una configuración física y la ubicación de la informacion necesaria

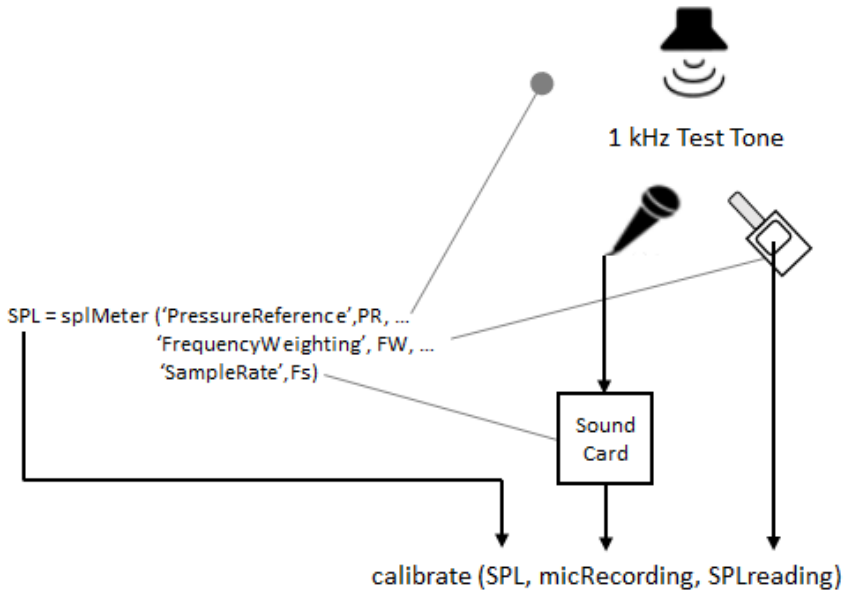


Figura 5.1: Esquema de calibración

Donde

$$factorCalibracion = \frac{10^{(\frac{SPLlectira-k}{20})}}{rms(x)}$$

5.4 Diseño de soporte

En esta etapa se detalla el diseño propuesto del soporte acoplado al montaje experimental para la identificación de los modos de vibrar y medición de voltaje de fruto de café arábica. El proceso consiste en proponer el diseño de un soporte donde se aloje el piezoeléctrico en el cual estará adherido un fruto de café, dicho soporte debe ser integrado a un excitador electrodinámico y el piezoeléctrico a una tarjeta de adquisición de datos

5.4.1 Diseño de soporte

Las especificaciones para el diseño del soporte cumplen las siguientes características alta rigidez, fácil montaje y desmontaje a un suporte principal. El diseño también contemplo el alojamiento de un piezoeléctrico

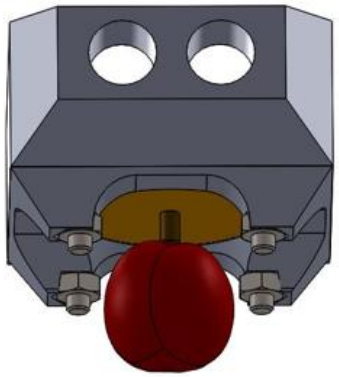


Figura 5.2: modelo 3D montaje

Además, se diseña un soporte inferior para el alojamiento de un piezoeléctrico el cual se acopla por medio de tornillos M3, este soporte inferior es desmontable, permitiendo una mejor configuración para realizar las medidas

5.4.2 Manufactura soporte

Posteriormente a su diseño esta pieza fue manufacturada en impresión 3D usando material PLA con una configuración definida de 80 % de relleno, altura de capa de 0.1 mm y matriz de relleno tipo triángulo. Los planos con sus respectivas dimensiones pueden ser consultados en el Anexo-D. En la Figura 1 se muestra el CAD del soporte de fijación para el alojamiento del piezoeléctrico, y el prototipo manufacturado en impresión 3D. Adicionalmente en la Figura 3 se muestra ensamble del soporte con el dispositivo excitador electrodinámico.

5.5 Montaje

Uno de los retos para adaptar este modelo de pruebas es encontrar un soporte para remplazar el método propuesto por la UNAM que supone un agitador.

Un soporte y un tornillo de 6 mm para sujetar el soporte del piezoeléctrico lo que facilita una configuración que causa interferencia con las mediciones realizadas

Figura 5.3: Vista superior montaje con piezoeléctrico

5.6 Simulaciones

Para calcular la optimización de la forma de los límites a una superficie asimétrica simple. Donde las consideraciones son las siguientes, el nivel de presión sonora de campo lejano se maximiza para la frecuencia en de máxima excitación en los frutos maduros y en una sola dirección. La atención se centra en el procedimiento de optimización, que implica la elección de la función objetivo y la configuración del solucionador de optimización. La deformación de la geometría se realiza mediante la funcionalidad de optimización de la forma de COMSOL Multiphysics.

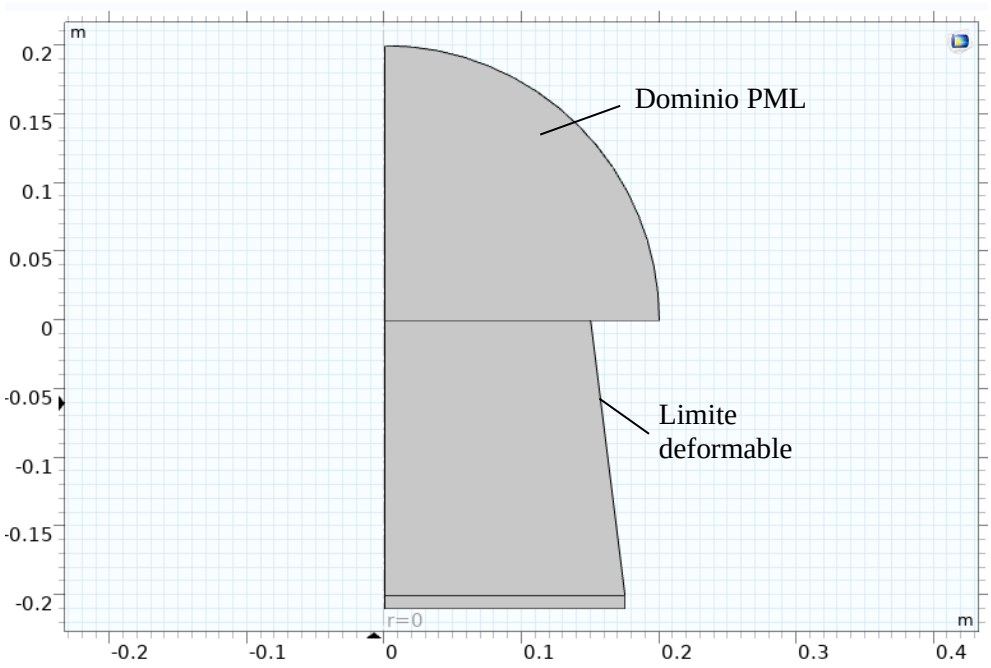


Figura 5.4: configuración inicial

5.6.1 Definición del modelo

Un modo de onda plana alimenta una bocina axisimétrica que irradia desde un bafle infinito hacia un semiespacio abierto; véase la figura 5.3. Se supone que el radio de la guía de ondas de alimentación es fijo, así como la profundidad de la bocina y el tamaño del orificio en el que se fija la bocina al bafle. Variando la curvatura de la superficie inicialmente cónica de la bocina, se puede modificar su directividad e impedancia.

La superficie se parametriza suponiendo que el radio y la posición z de la trompa se desvían del cono simple mediante un conjunto de polinomios de Bernstein de 8º orden. El desplazamiento máximo (en las dos direcciones) viene dado por el parámetro dmax.

El número de Las variables de optimización están determinadas por el orden del polinomio. El uso de un orden superior da más libertad y potencialmente un mejor valor final de la función objetivo, pero también hace que el proceso de optimización sea más sensible y puede generar una forma menos adecuada para la producción

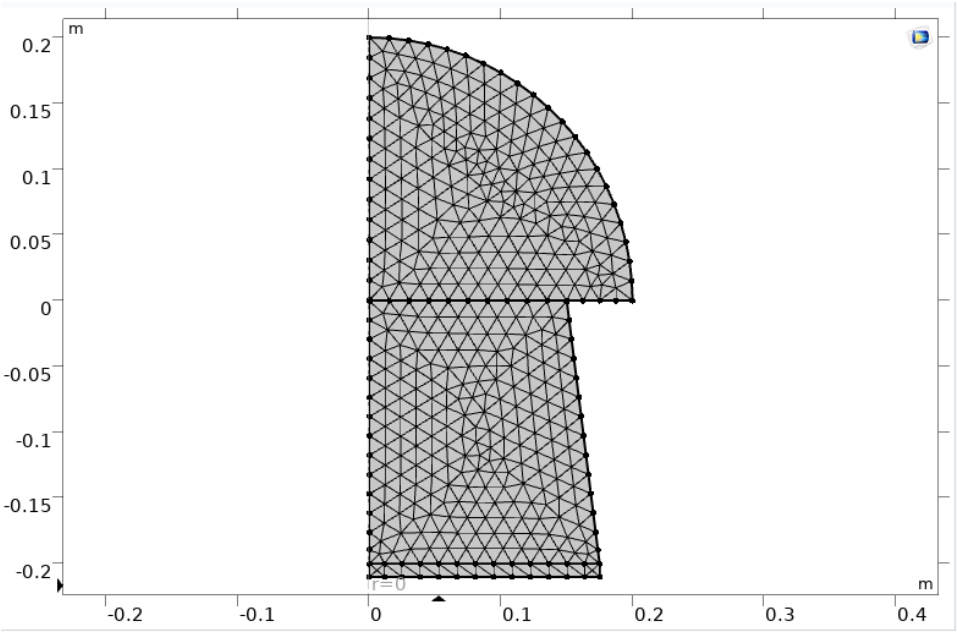


Figura 5.5: configuración de malla

Las geometrías de estudio que se van a optimizar son las diferentes configuraciones de cono invertido que se esperan aumenten la presión acústica en un área específica y posteriormente realizar la optimización de la forma geométrica.

Figura 5.6: Geometrías analizadas

Capítulo 6

6 Resultados

6.1 Introducción

Una vez ha finalizado el proceso de entrenamiento, diseño e implementación del sistema, llega el momento de probar su funcionamiento en determinadas situaciones, para, de esta manera, detectar sus puntos fuertes y sus debilidades, permitiendo en un futuro realizar mejoras del mismo.

Para ello, se analizan las utilidades del sistema propuesto y la importancia de las funcionalidades ofrecidas. Dada las características del sistema implementado, no existen mecanismos de *benchmark* o medición para este caso concreto; no obstante, caben diversas opciones a la hora de probar el correcto funcionamiento del sistema.

En este caso, se ha optado por realizar una serie de pruebas de funcionamiento con diferentes imágenes y escenarios. Comparando los datos obtenidos por el sistema con los que realmente están en la imagen es posible comprobar la precisión del sistema. De esta manera las métricas utilizadas a la hora de comprobar la fortaleza del diseño son:

- mAP(mean average precision)
- precisión
- exhaustividad

Concretamente, la máquina donde se ha realizado el entrenamiento posee las siguientes especificaciones: máquina virtual de Google colab Tesla P100 16GB de memoria. Por tanto, estas características permiten establecer un límite inferior por debajo del cual las horas de entrenamiento del sistema tomarían varios días.

6.2 Entrenamiento

El tiempo que toma para entrenar un modelo depende las características de la maquina como lo son la memoria o procesador, sin embargo, cada modelo presenta un numero de parámetros por cada capa que dependen del interés del equipo que desarrolla la red.

Como se evidencia en la tabla el modelo que presento un menor tiempo de entrenamiento esla red Yolov5, a esta red le toma menos de la mitad que las demás redes en alcanzar los 200 *epoch.* Es fundamental mencionar que estos resultados se deben que, a pesar de no haber sido desarrollado por el mismo grupo de YOLO, Yolov5 presenta optimizaciones en sus capas ocultas además de que aplica nuevas capas con funciones que no existían anteriormente

Tiempo de entrenamiento		
Mobilenet v2	*YOLO v3*	*YOLO v5*
90 min	120 min	30 min

Tabla 6.1: tiempos de entrenamiento de redes convolucionales

6.3 Evaluación

La precisión relaciona los verdaderos positivos con los falsos positivos, en este caso las detecciones acertadas de café sobre las detecciones que no son frutos de café y el sistema las reconoció erradamente.

Según la tabla 1 la red que presentó el porcentaje más alto de precisión es la red YoloV5 que con alrededor de 84% supero a las redes seleccionadas, las demás redes presentaron un porcentaje mas bajo y en el caso de mobilenet v2 presenta unas grandes fluctuaciones en cada *epoch*

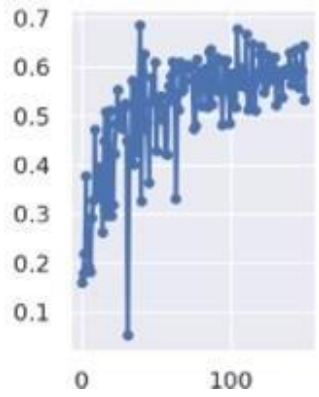
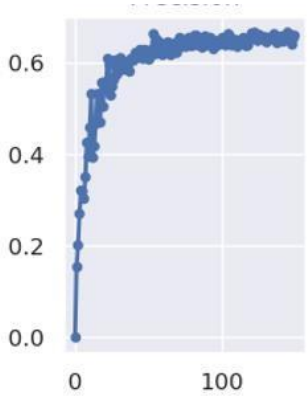
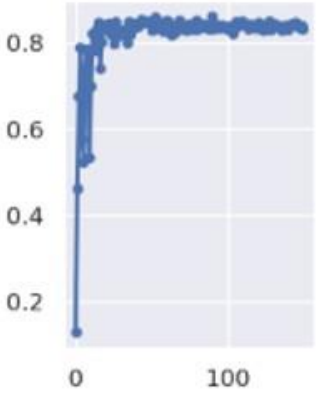
Precisión		
Mobilenet v2	*YOLO v3*	*YOLO v5*
		

Tabla 6.2: medida de precisión de las redes propuestas

La exhaustividad o *Recall* relaciona los verdaderos positivos con los falsos negativos, es decir las detecciones que no se tomaron en cuenta por el sistema, pero esta eran frutos de café.

Según la tabla 2 se evidencia que el porcentaje de Yolov5 y mobilenet v2 son los más óptimos y presentan resaltados alrededor de 80% de clasificación sin embargo YoloV5 presenta menores fluctuaciones y picos entre cada epoch, esto lo hace la mejor opción para aplicar.

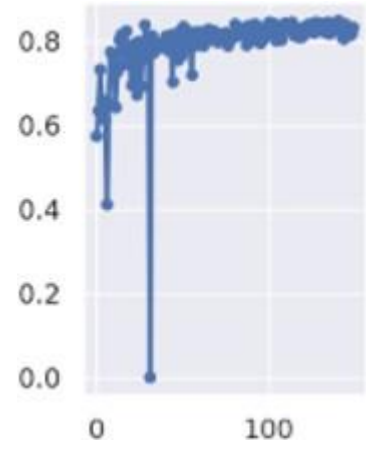
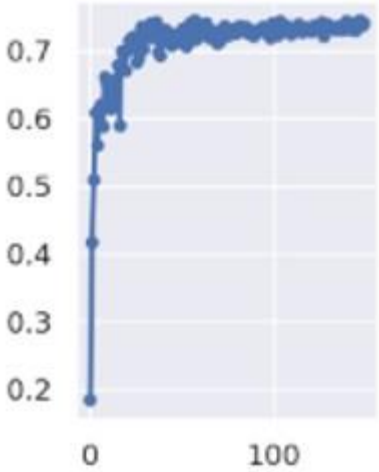
Exhaustividad		
Mobilenet v2	*YOLO v3*	*YOLO v5*
		

Tabla 6.3: medida de exhaustividad de las redes propuestas

El mAP es una medida muy usada en el campo de la detección de objetos ya que interviene el área de la región delimitadora predicha y el área de etiqueta. En muchos casos el mAP relaciona un conjunto de entre 0.5 y 0.95 es decir la media de cada valor nominal de mAP.

Según la taba 3 el mAP 0.5 de las tres redes presenta resultados similares que rondan el 80% de clasificación sin embargo Yolov3 y Yolov5 presentan menor fluctuación por cada epoch lo que hace que sean las mejores opciones en caso de definir el mAP como criterio de selección.

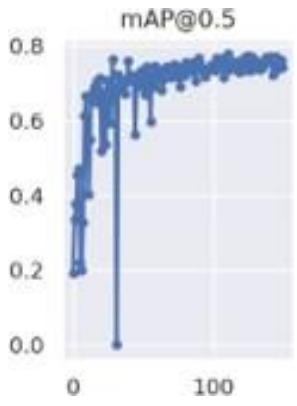
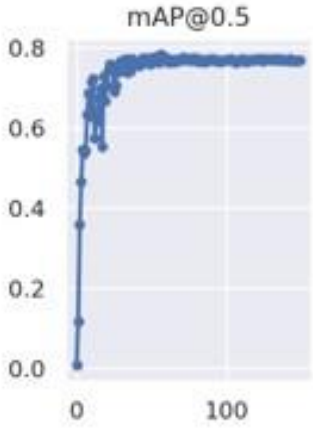
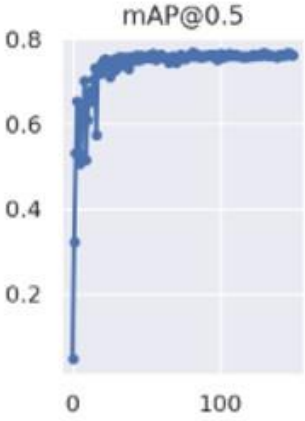
mAP 0.5		
Mobilenet v2	*YOLO v3*	*YOLO v5*
		

Tabla 6.4: medida de mAP de las redes propuestas

Otra de las medidas que el campo de la detección de objetos es muy popular y de gran acogida es el mAP general que va desde 0.5 hasta 0.95 esto lo hace ser uno de los mejores criterios de selección ya que evalúa que tan preciso es el sistema cuando hay una escala mucho mas estricta de coincidencia de áreas

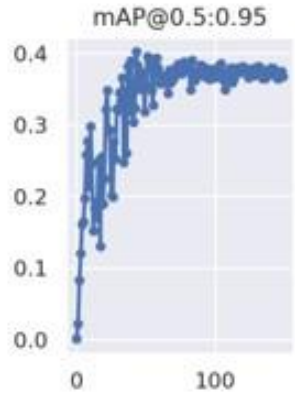
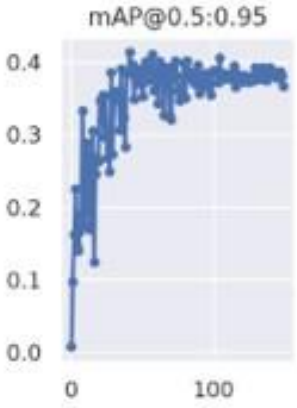
mAP 0.5:0.95		
Mobilenet v2	*YOLO v3*	*YOLO v5*
		

Tabla 6.5: medida de mAP intervalo de las redes propuestas

Realizando algunas detecciones como se evidencia en la tabla 5 se puede concluir que mobilenet v2 no detecta ciertos frutos que estas alejados del foco, a diferencia de Yolo v3 y Yolov5 que son capaces de detectar frutos a una escala de profundidad mucho mayor. Es fundamental agregar que, aunque Yolov5 presento un mAP mayor esta diferencia no es tan significativa cuando se evalúa de forma visual

Mobilenet v2	*YOLO v3*	*YOLO v5*
		

Tabla 6.6: resultados de detección

5.3.1 Matriz de confusión

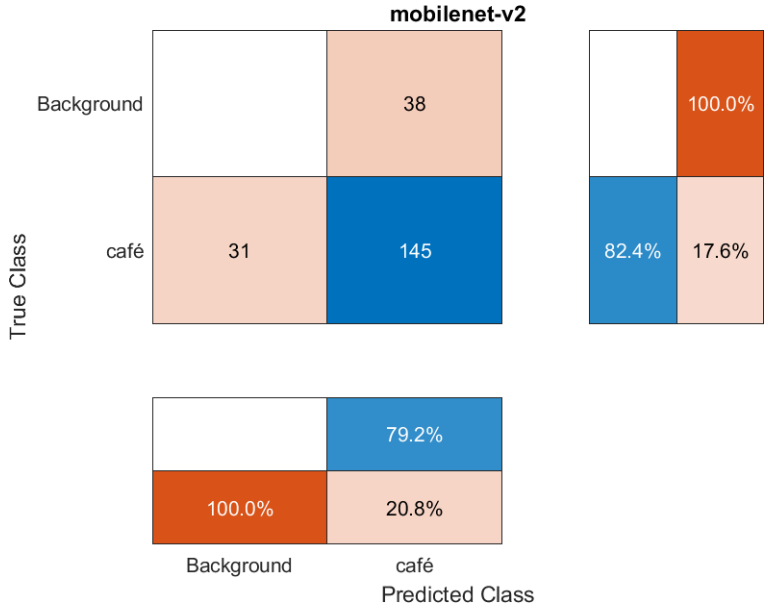


Figura 6.1: matriz de confusión mobilenet-v2

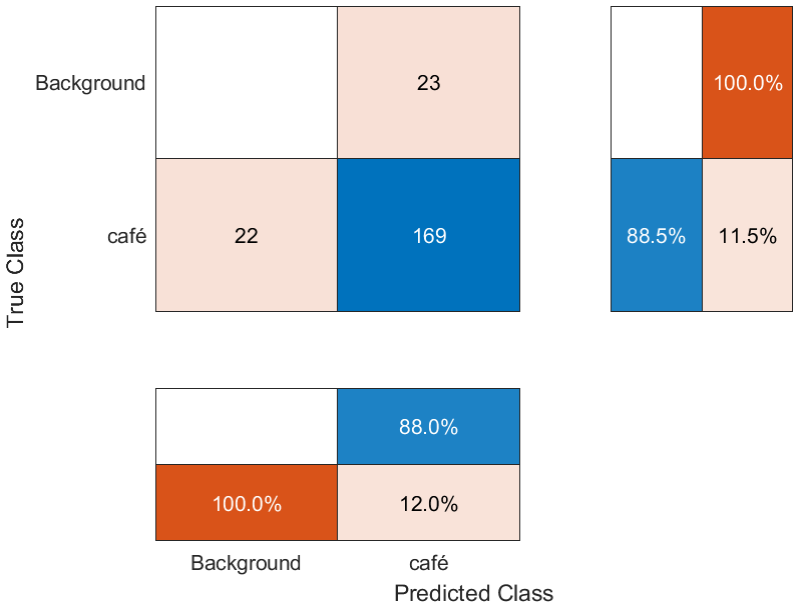


Figura 6.2: matriz de confusión yolo-v3

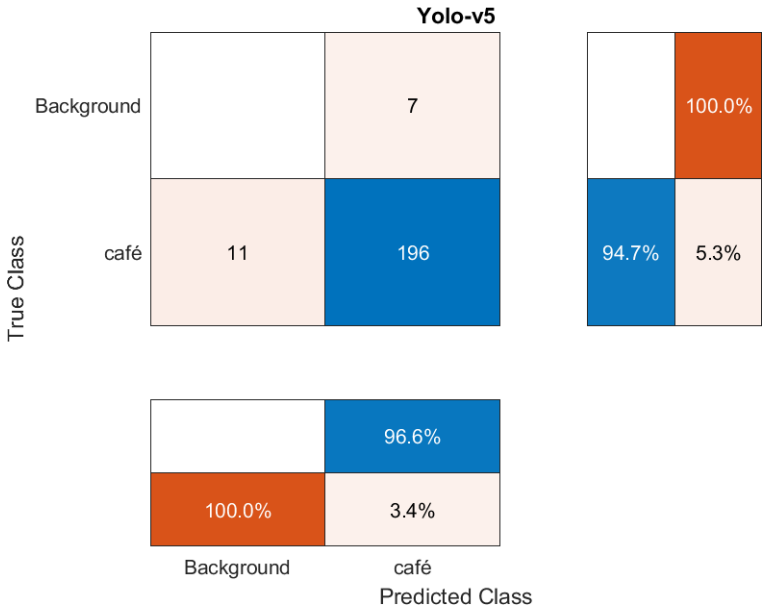


Figura 6.3: matriz de confusión yolo-v5

6.4 Clasificación de estados de maduración

Teniendo en cuanto los resultados de la sección anterior en donde solo se detectaba la presencia de granos de café y tomando como referencia la precisión y los tiempos de entrenamiento, los cuales son factores determinantes a la hora de elegir el modelo.

Yolov3 y Yolov5 son los métodos que se eligieron para formar parte del nuevo sistema de detección y clasificación de los estados de maduración.

5.4.1 Yolov3

Figura 6.4: clasificación yolov3

Figura 6.4: clasificación yolov3

5.4.2 Yolov5

Figura 6.5 clasificación yolov5

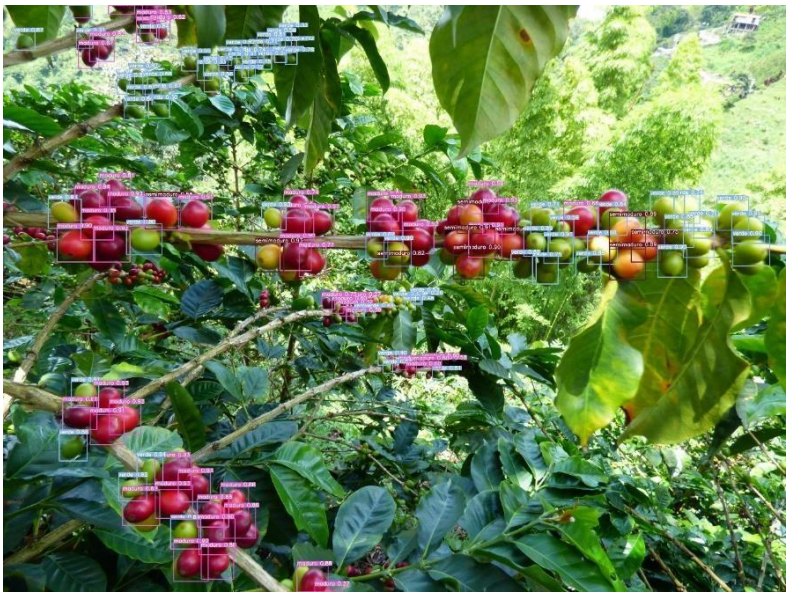


Figura 6.5 clasificación yolov5

Figura 6.6 clasificación yolov5

Mediante la figura 6.6 se puede evidenciar que el sistema es capaz de detectar en diferentes condiciones de iluminación lo además de que el sistema es capaz de detectar objetos a una distancia focal que nofue considerado en el conjunto de entrenamiento.

Se puede concluir que yolov3 y yolov5 presentar resultados bueno en la detección y clasificación de los diferentes estados de maduración sim embargo la gran diferencia radica en los tiempos de entrenamiento y en los tiempos de inferencia ya que yolov5 presenta tiempos de hasta cinco veces más cortos

Prueba 53 imágenes

Red	Tiempo de inferencia
Yolov3	30 s
Yolov5	25 s

6.5 Pruebas acústicas

Para caracterizar los elementos acústicos y la manera en cómo afectan las variables que se están analizando es importante evidenciar el comportamiento de cada una de estas variando las características como la frecuencia de la fuente acústica

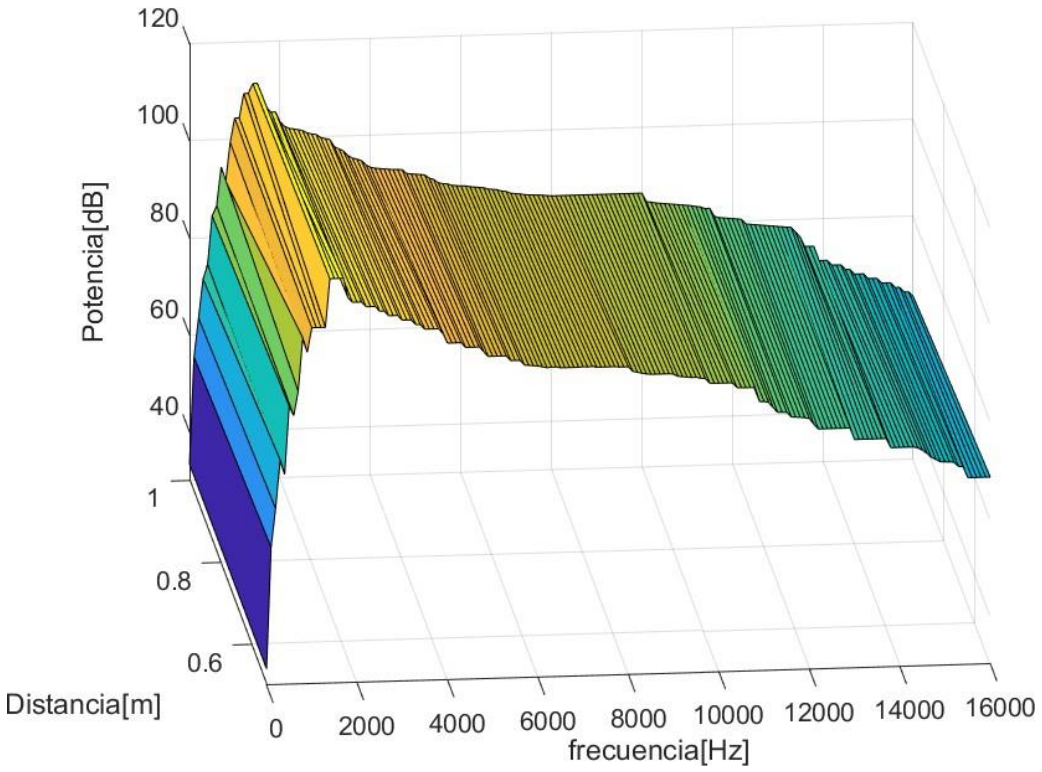


Figura 6.7: caracterización sistema acústico

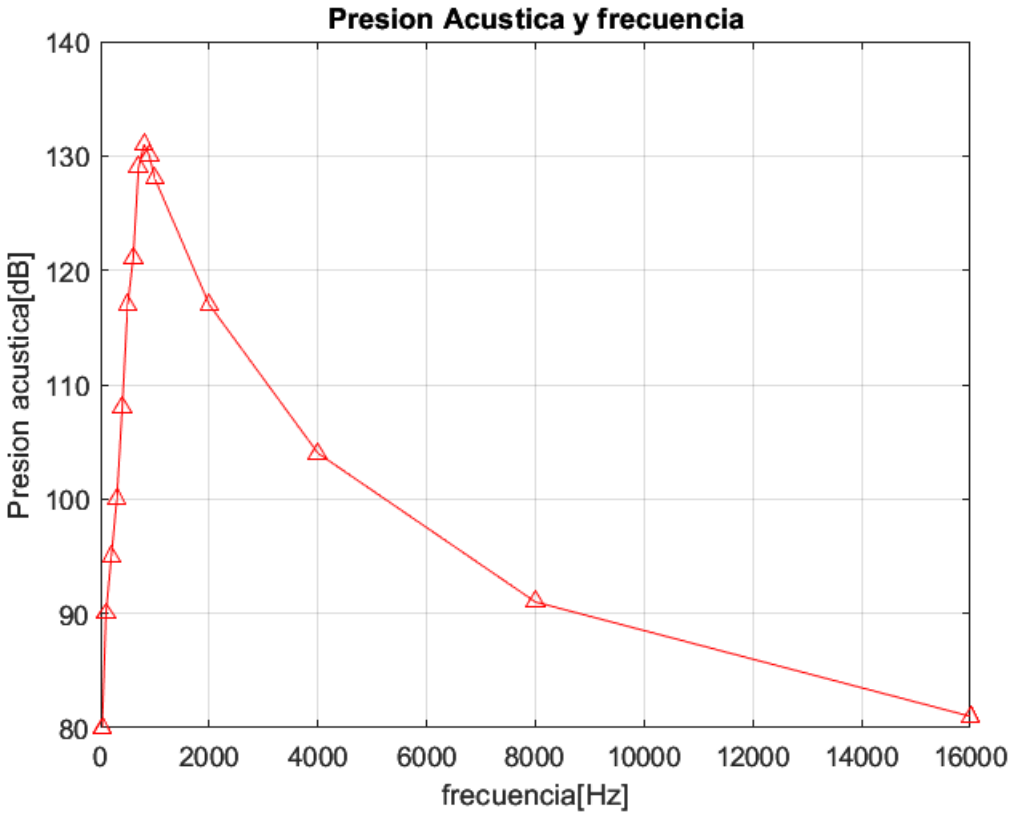


Figura 6.8: caracterización sistema acústico

Como se evidencia en la figura 5.7 se relaciona la frecuencia y la intensidad registrada. Se evidencia que para el arreglo armónico lo mayores valores registrados se encuentranentre 800 y 1000 Hz donde alcanza un máximo de 130 dB, dichas frecuencias serán tenidas en cuenta a la hora de registrar la vibración de los diferentes estados de maduración

6.6 Vibración estados de maduración

Evaluar los diferentes estados de maduración con incrementos de 10 Hz cada tres segundos en donde las frecuencias objetivas fueron entre 40 Hz y 1000Hz ya que causan la mayor presión acústica.

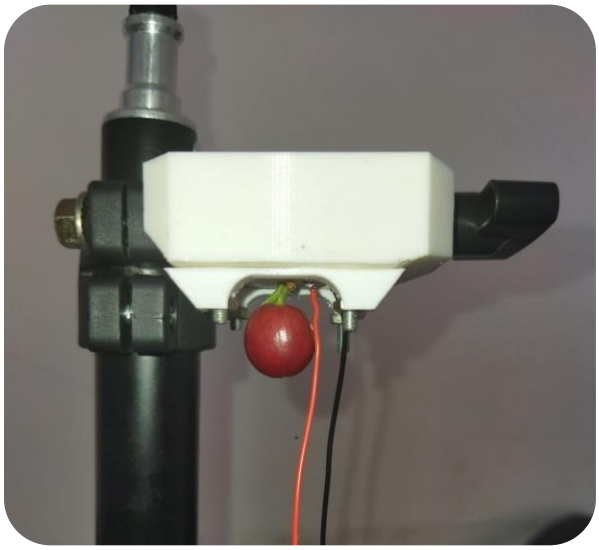


Figura 6.9a: Montaje fruto maduro

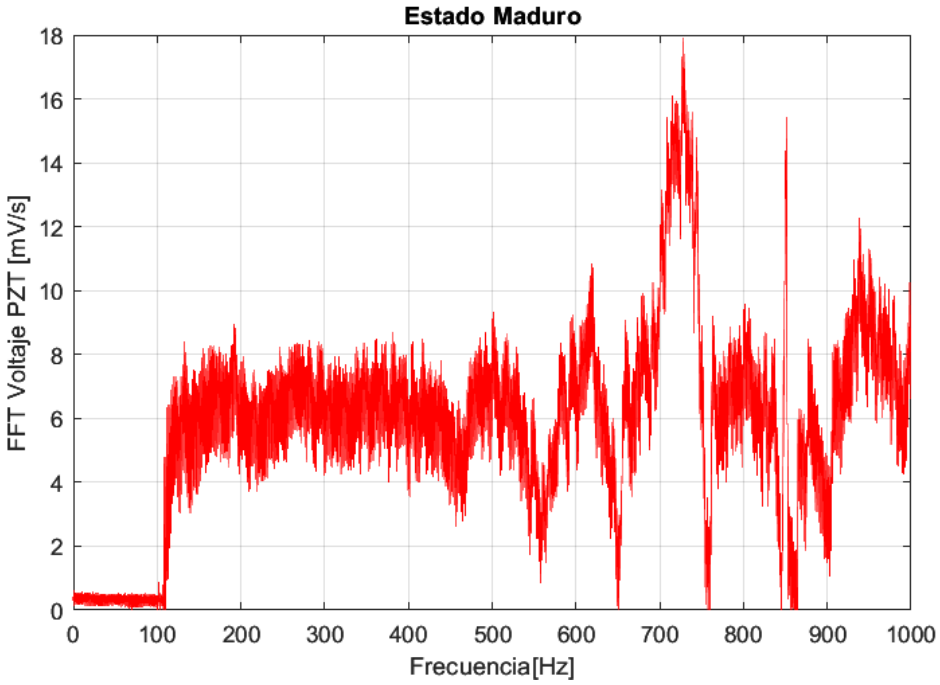


Figura 6.9b: respuesta fruto maduro

De la gráfica 6.9 es posible inferir que en los rangos entre 700-800 Hz los frutos maduros presentan el mayor grado de excitación lo cual será de interés a la hora de diseñar una geometría que permita focalizar y aumentar la presión acústica a la salida de sistema

También es posible evidenciar que a medida que va variando la frecuencia se van produciendo antinodos que oscilan en torno a un punto de equilibrio

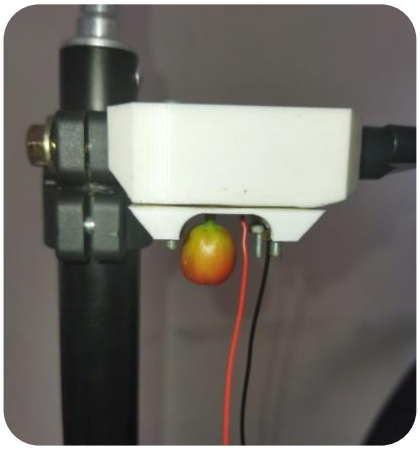


Figura 6.10a: Montaje fruto semimaduro

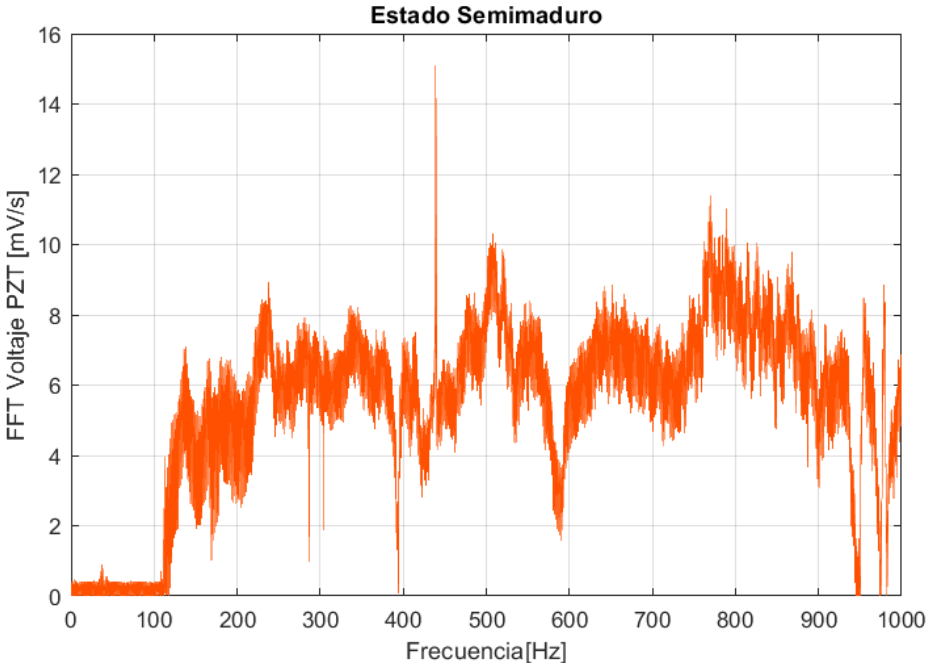


Figura 6.10b: respuesta fruto semimaduro

En la gráfica es posible inferir que en los rangos entre 700-800 Hz los frutos semi-maduros presentan el mayor grado de excitación y se alcanza el mayor grado de amplitud en el antinodo.

El comportamiento de los frutos semiduros es similar al de los maduros sin embargo estos requieren más energía para alcanzar los mismos niveles de excitación.

Figura 6.11a: Montaje fruto verde

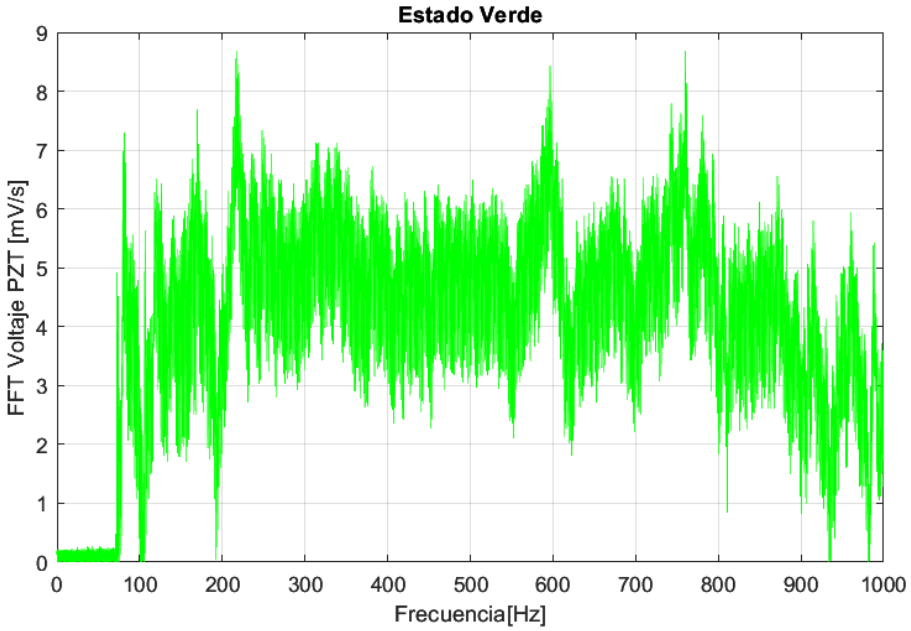


Figura 6.11b: respuesta fruto verde

En la gráfica 6.11 los frutos verdes presentan valores mucho menores que los registrados

en los demás estados de maduración además de que no se evidencia un claro rango de frecuencias en las cuales sea más grande la amplitud.

Prueba sin carga

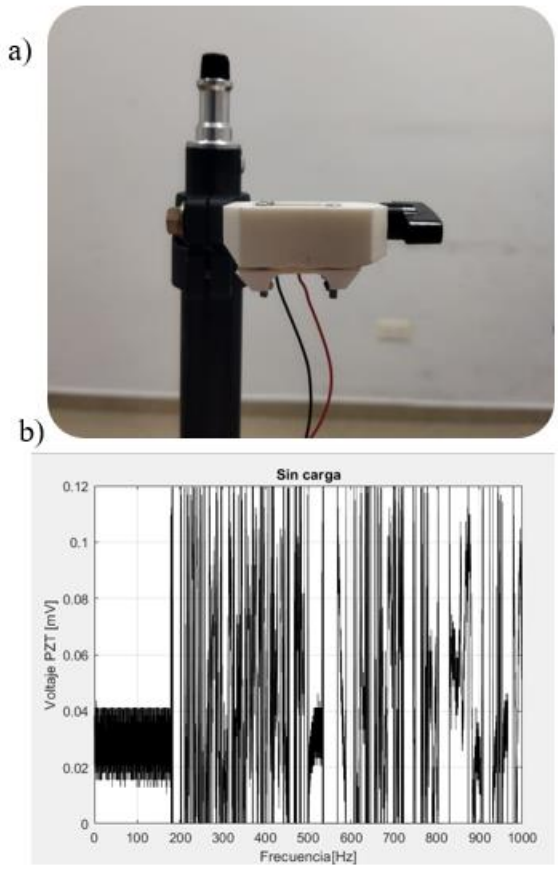


Figura 6.12: respuesta sin carga

Según los datos obtenido de la lectura en el piezo-eléctrico sin carga se puede inferir que las valores pico no son tan significativos para que afecte las mediciones registradas en cada uno de los estados de maduración.

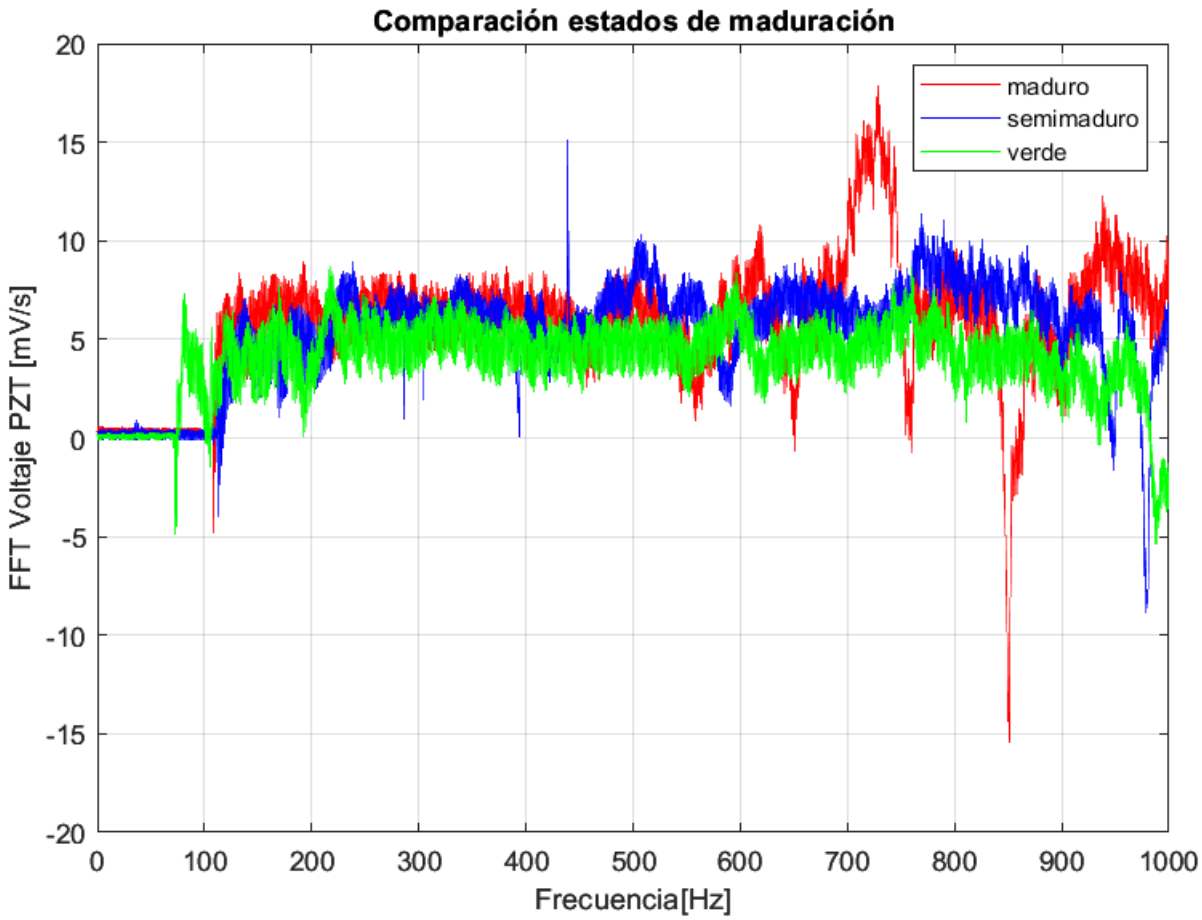


Figura 6.12: Comparación estados de maduración

6.7 Filtrado de señales

Los datos registrados a la hora de hacer mediaciones presentan un ruido de alta frecuencia por lo que es esencial realizar una fase de filtrado puede eliminar el ruido y de esta manera poder evidenciar de una mejor manera los datos registrados.

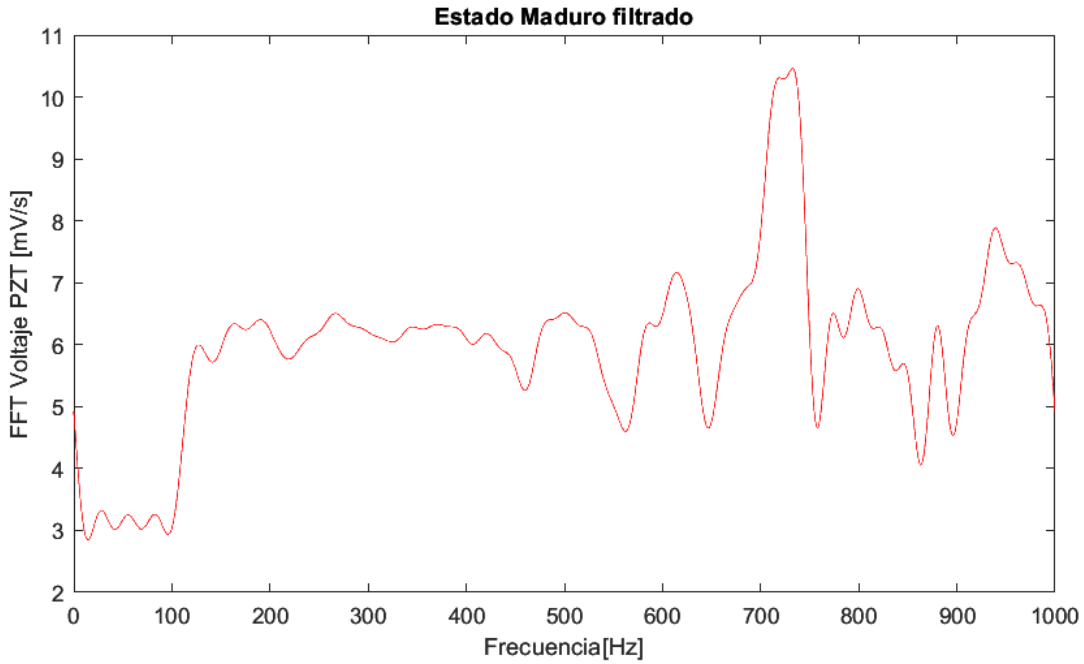


Figura 6.13: Filtrado señal fruto maduro

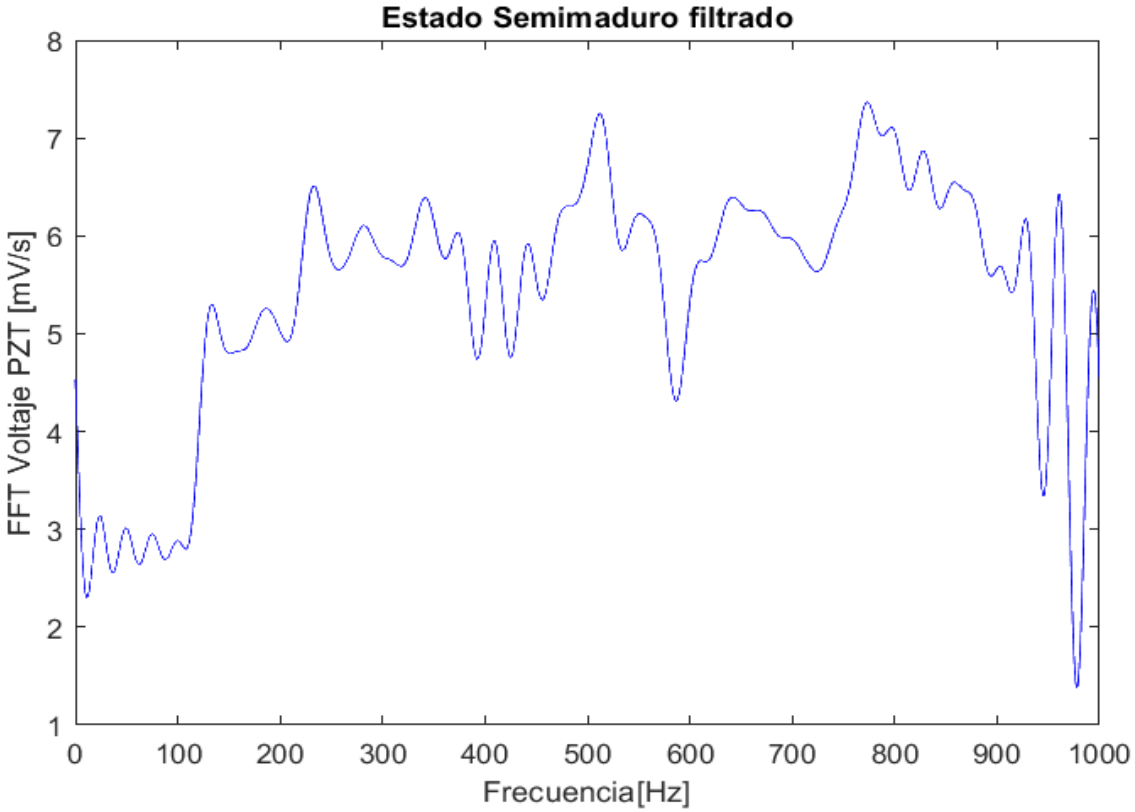


Figura 6.14: Filtrado señal fruto semi-maduro

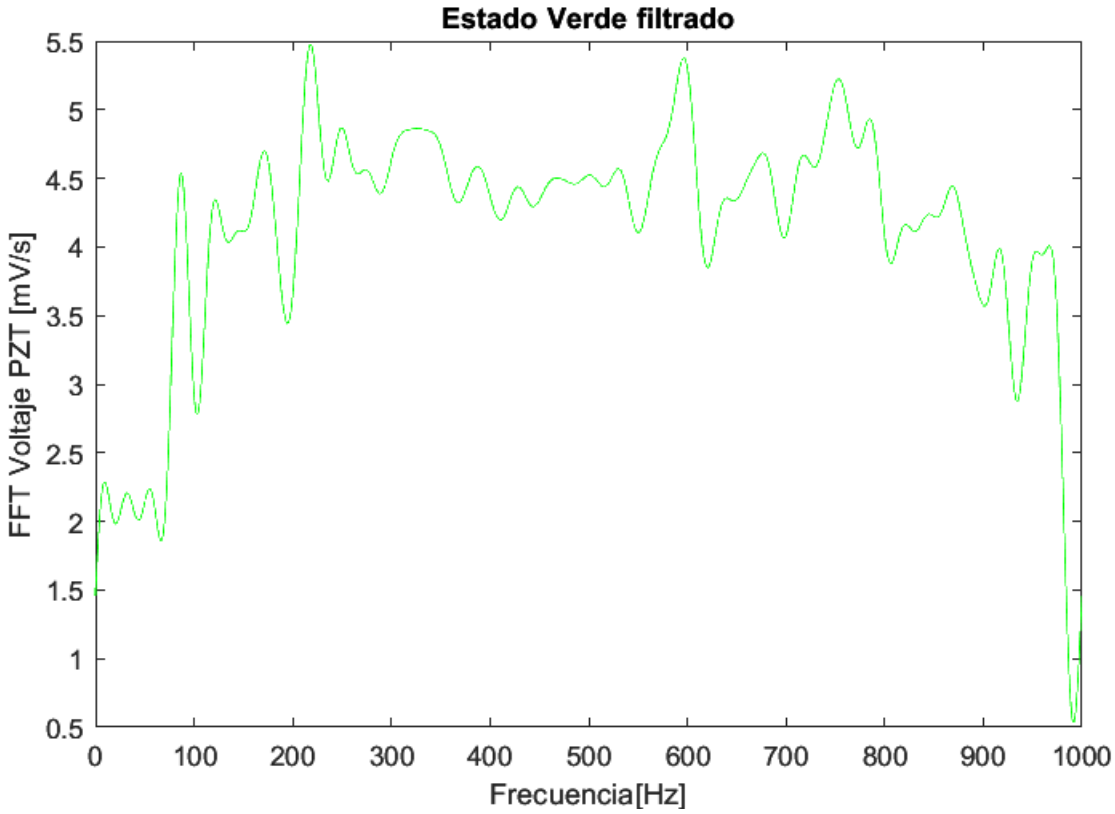


Figura 6.14: Filtrado señal fruto verde

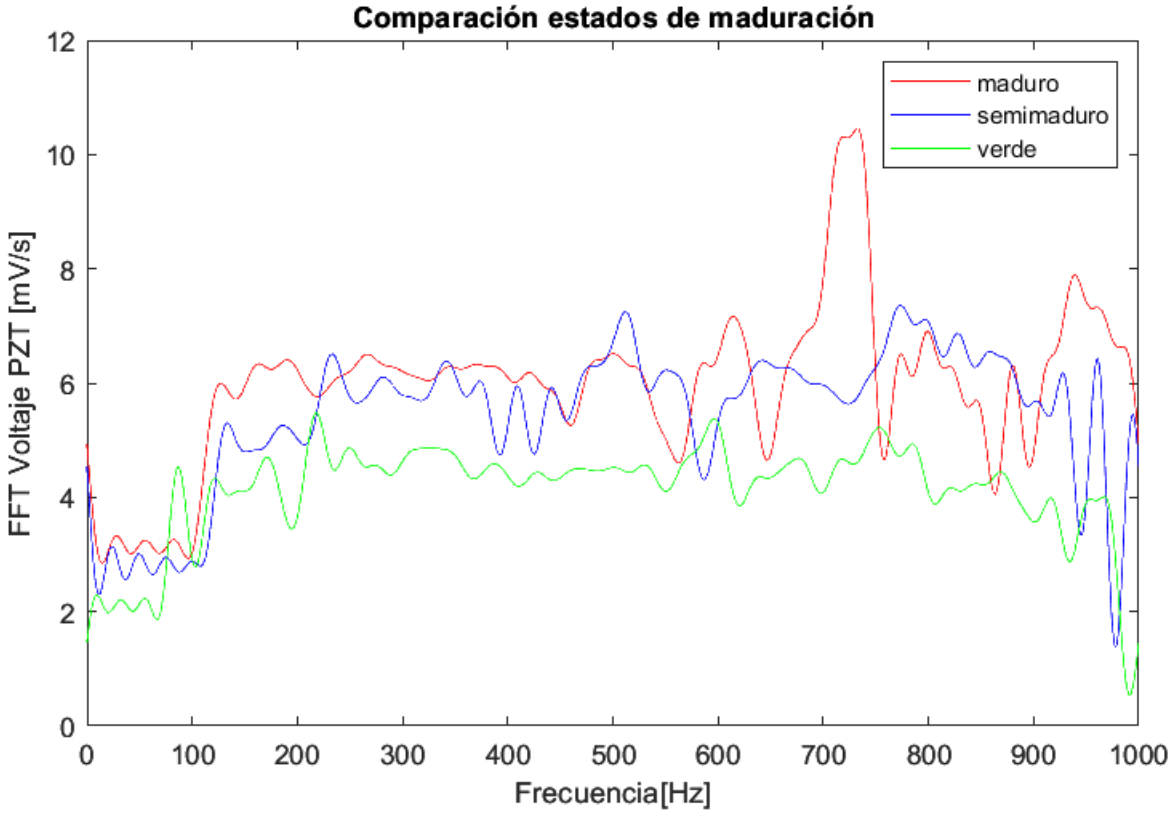


Figura 6.15: Filtrado y comparación estados maduración

En la figura 6.15 se puede establecer un rango de frecuencias especificas en las que se pueden distinguir los diferentes estados de maduración. Los frutos maduros resuenan entre 700 y 800 Hz alcanzo el máximo de voltaje registrados por el sensor.

En el caso de los frutos verdes no presentan valores muy significativos que permitan distinguir rangos de frecuencias. los semi-maduros presentan un comportamiento similar a los maduros sin embargo no presentan una amplitud tan alta registrada en el piezoeléctrico y se puede establecer en un rango de 750 y 900 Hz

6.8 Simulación focalización acústica

6.8.1 Configuración 1 cono recto

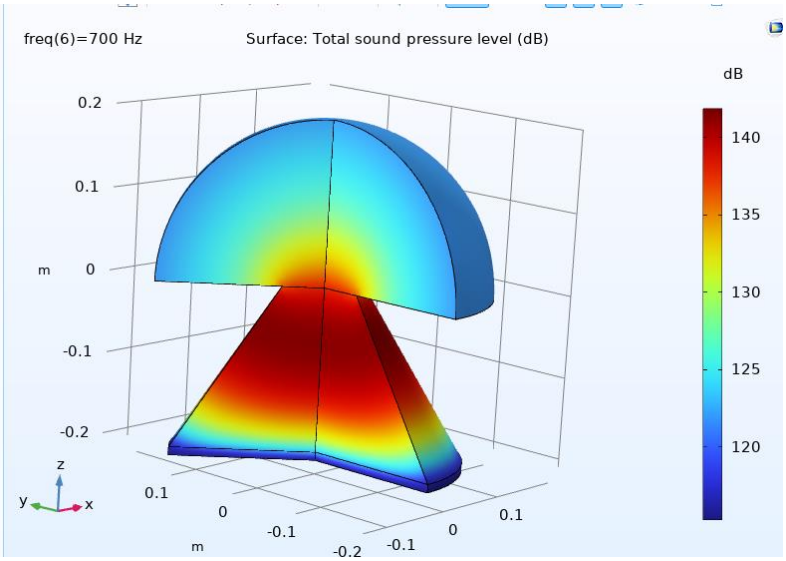


Figura 6.15: presión acústica configuración 1

De la configuración 1 se puede inferir que por las características geométricas se producen reflexiones internas de las ondas acústicas que minimizan la presión acústica a la salida del focalizador.

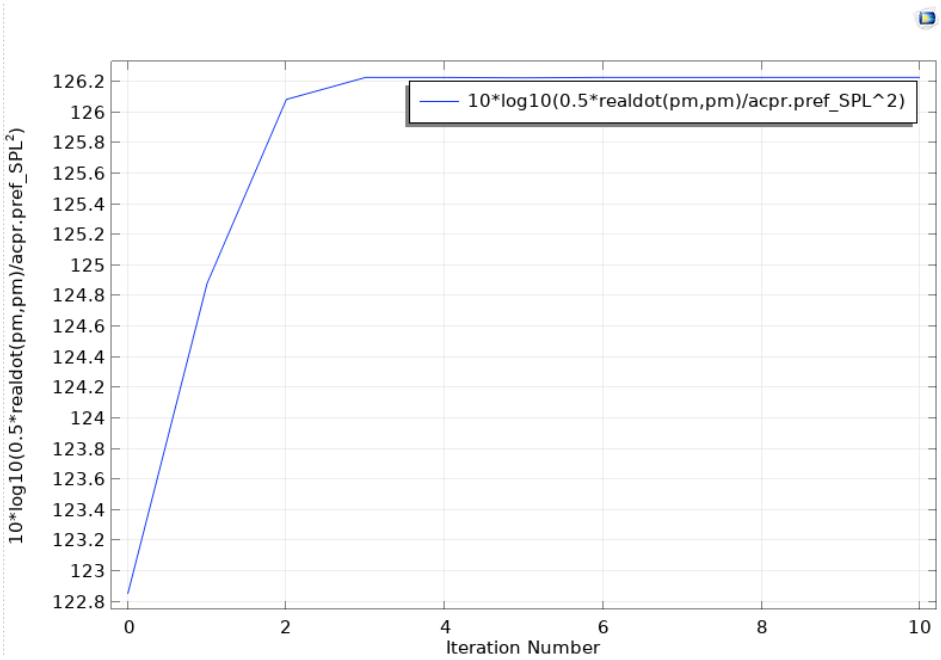


Figura 6.16: presión acústica región interés configuración 1

6.8.2 Configuración 2 cono recto optimizado

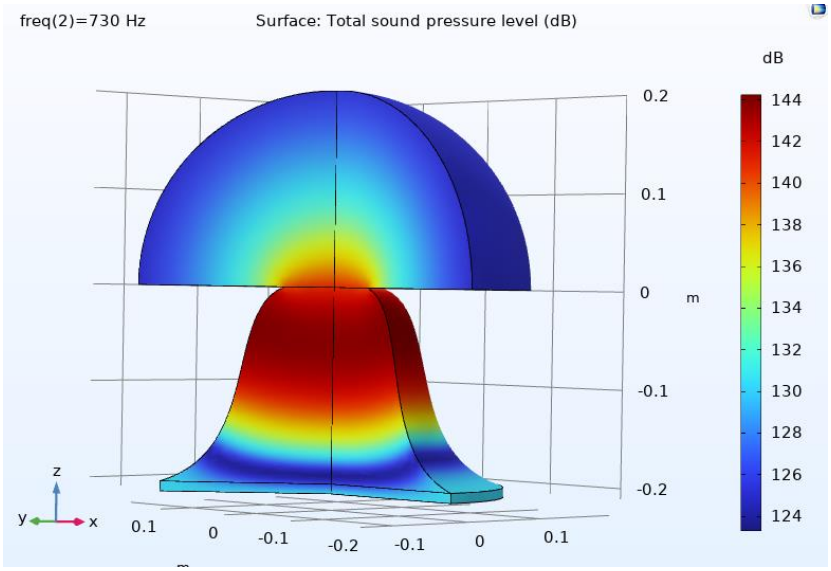


Figura 6.17: presión acústica región interés configuración 2

En la configuración 2 se realizó la optimización con un polinomio de orden 4 que tienen como resultado una geometría cóncava en el tramo final del focalizador.

6.8.3 Configuración 3 cono optimizado

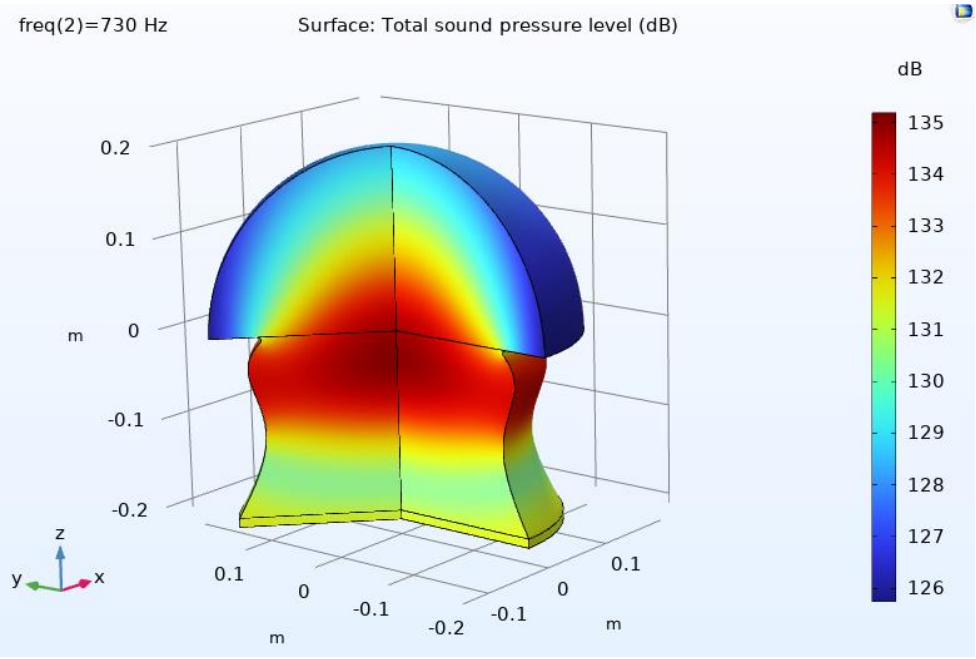


Figura 6.17: presión acústica región interés configuración 3

En la configuración 3 se presenta una geometría que incrementa la presión acústica para la frecuencia de mayor excitación en los frutos maduros.

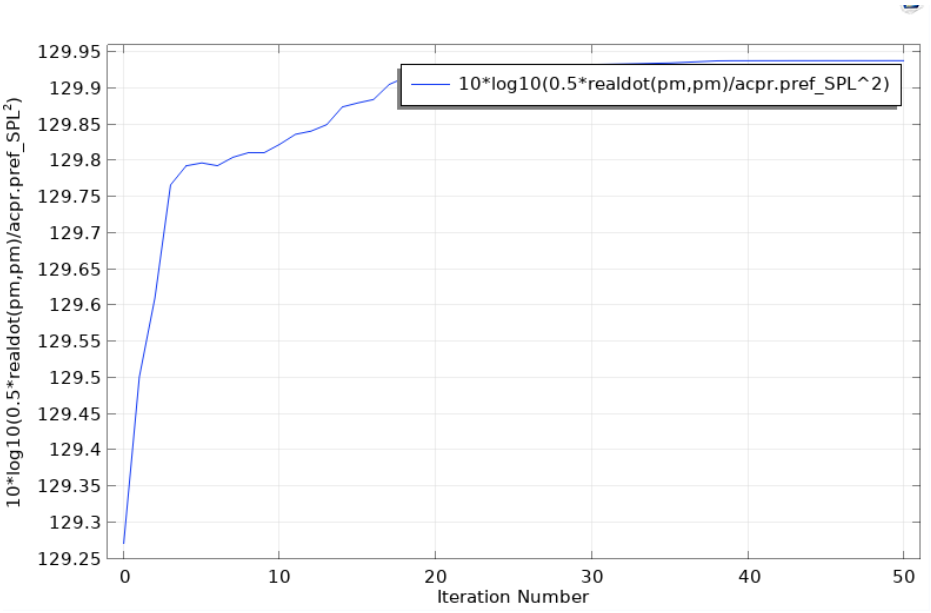


Figura 6.18: presión acústica región interés configuración 3

Un incremento de 3 dB supone un incremente del doble de la potencia de la fuente por lo que es considerable el aumento desde el punto de vista de la energía requerida para lograr el sistema fruto pedúnculo.

También es pertinente aclarar que este sistema de focalización es óptimo solo para las frecuencias de que comprenden los rangos entre 700 y 800 Hz que fueron las frecuencias de mayor excitación en los frutos maduros.

6.9 Simulación acústica estructural

6.9.1 Definición del modelo

Este modelo simula el comportamiento de un fruto de café en estado maduro en un dominio con una onda acústica incidente en el aire. Las paredes del objeto reciben el impacto de la presión acústica. El modelo calcula la respuesta en frecuencia del sólido y, a continuación, devuelve esta información al dominio acústico para que pueda analizar el patrón de ondas resultante.

Para configurar este modelo se utiliza la interfaz multifísica Acoustic-Solid Interaction, Frequency Domain. Esta interfaz incluye dos interfaces físicas: Mecánica de sólidos y presión acústica en el dominio de la frecuencia.

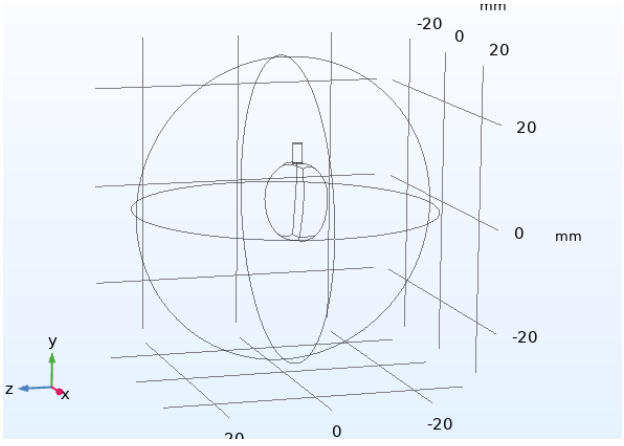


Figura 6.20: Geometría acústica estructural analizada

6.9.2 Ecuaciones de dominio

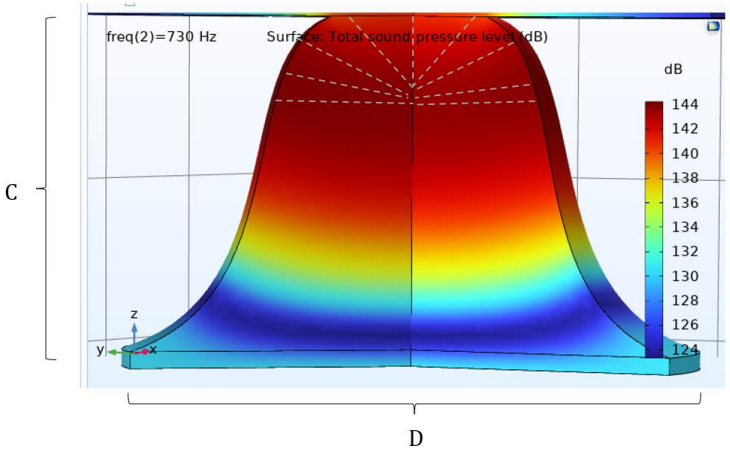
Se usa la ecuación de Helmholtz que modela las ondas sonoras armónicas en el dominio para la presión sonora.

$$\nabla \cdot \left(-\frac{1}{\rho_c}\nabla p\right) - \frac{\omega^2 p}{\rho_c c_c^2} = 0 \qquad (6.1)$$

Donde:
p presión $\frac{N}{m^2}$
ρ_c densidad $\frac{Kg}{m^3}$
ω frecuencia angular $\frac{rad}{s}$

6.9.2.1 Análisis foco acústico

El punto focal es un tema de interés de este estudio, ya que este punto se refiere al lugar en el que se produce la mayor intensidad acústica en un determinado espacio, hallar este punto puede asegurar que el fruto se excite con la mayor intensidad para una configuración especifica.



$$\lambda = \frac{2D}{16C} \qquad (6.2)$$

resolviendo la configuración para esta configuración se obtiene que la posición del foco en donde se ubicara el fruto para realizar los estudios de intensidad y desplazamiento

6.9.3 Convergencia de malla

El objetivo principal de convergencia de malla determina si los resultados de la simulación son los suficientemente precisos para el uso en aplicaciones del mundo real. Para esto, se llevan a cabo se llevan a

cabo simulaciones con diferentes resoluciones de malla y se comparan los resultados en términos de precisión y estabilidad.

Para este estudio se tomaron las siguientes configuraciones de malla

Estudio 1 Análisis asimétrico

- 475 elementos de dominio
- 86 elementos de frontera

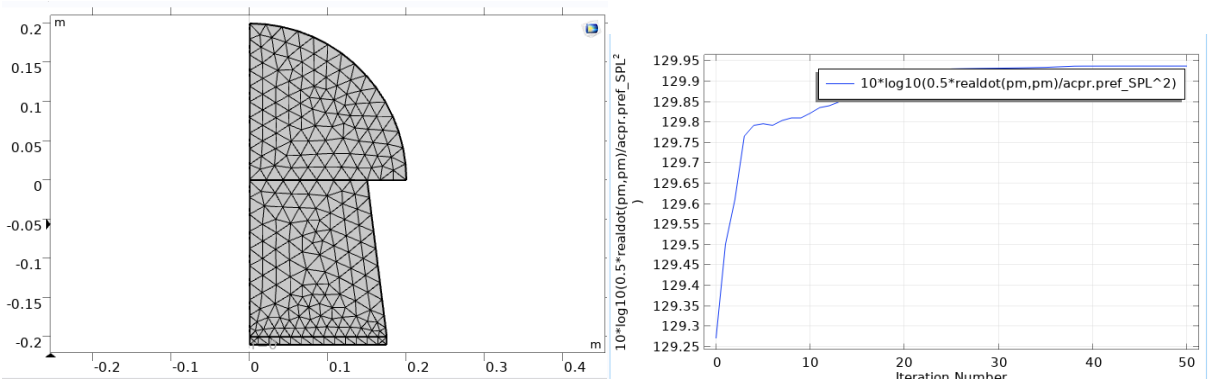


Figura 6.21: Configuración malla 1

- 843 elementos de dominio
- 107 elementos de frontera

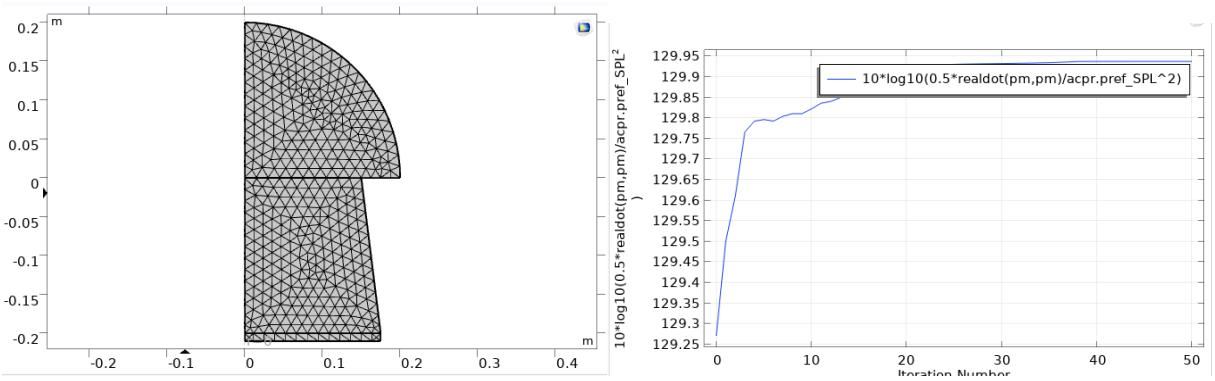


Figura 6.22: Configuración malla 2

- 2746 elementos de dominio
- 190 elementos de frontera

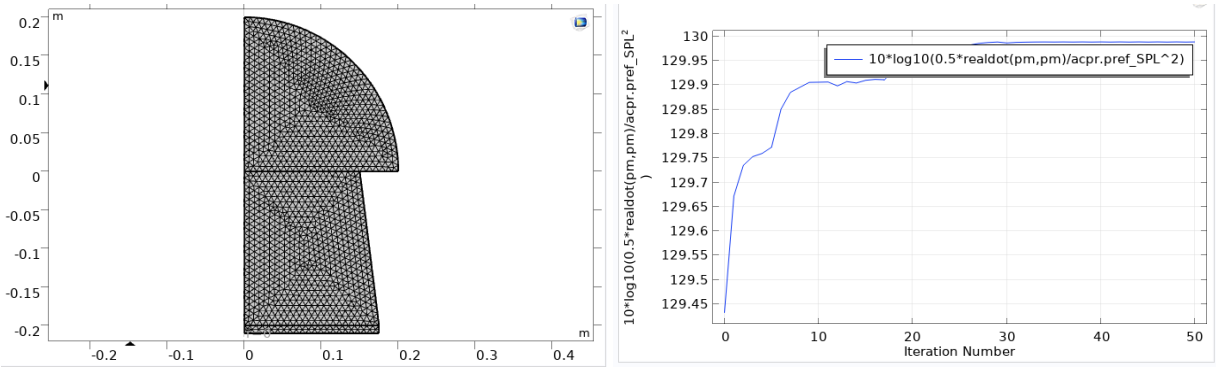


Figura 6.23: Configuración malla 3

El análisis de los resultados obtenidos indica que la precisión de los resultados aumento significativamente menor para las demás configuraciones de malla lo que indica que una configuración de entre 1000 y 2000 elementos de dominio se obtienen resultados precisos para el problema analizado

Estudio 2 Análisis 3D

- 96901 elementos de dominio
- 10726 elementos de frontera
- 819 elementos de borde

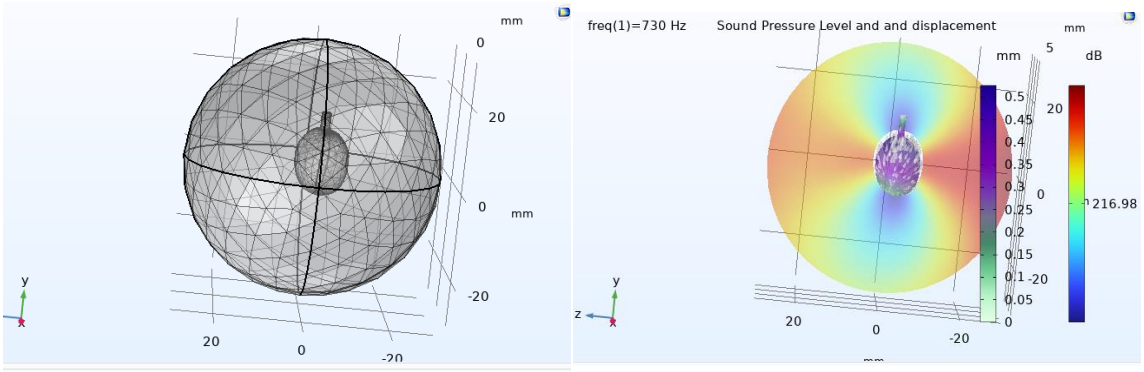


Figura 6.24: Configuración malla 1 estudio 3D

- 258346 elementos de dominio
- 23324 elementos de frontera
- 1649 elementos de borde

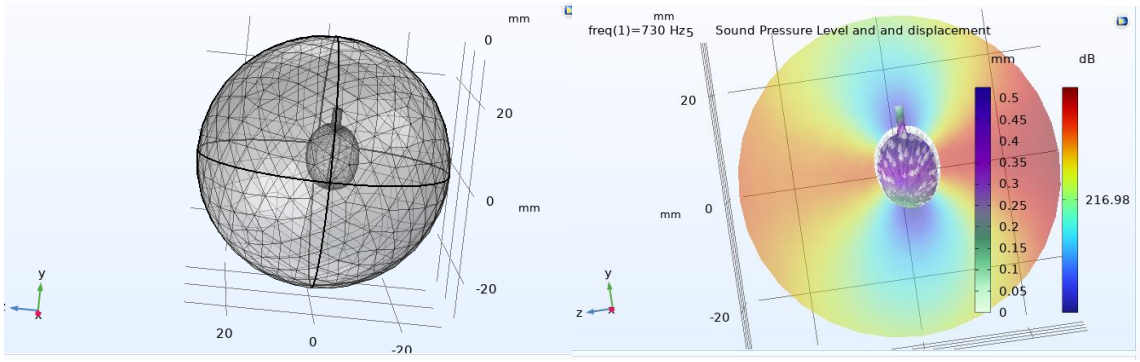


Figura 6.25: Configuración malla 2 estudio 3D

- 391948 elementos de dominio
- 31316 elementos de frontera
- 2077 elementos de borde

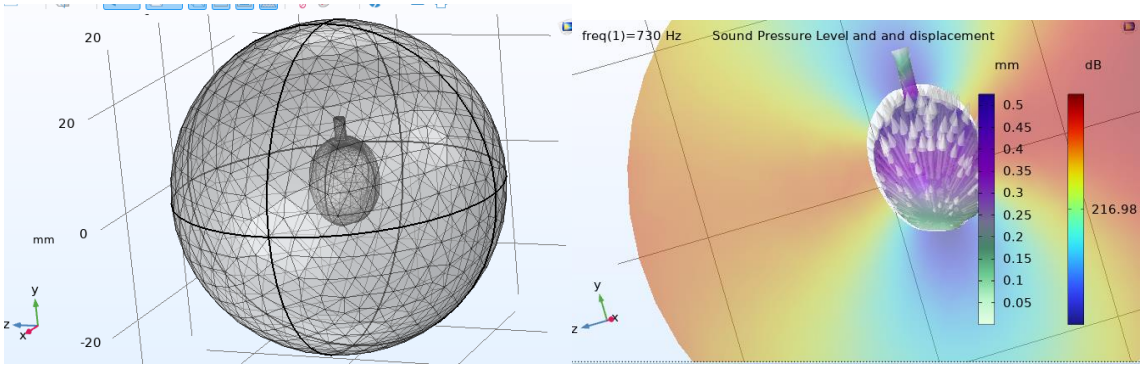


Figura 6.26: Configuración malla 3 estudio 3D

Según los resultados obtenidos variando la resolución de la malla se encontró que los resultandos no variaban significativamente en las diferentes configuraciones por lo que para este problema una malla que cuente con alrededor de 200.000 elementos de dominio se obtiene

resultados acertados para el problema abordado en este estudio.

Además, se evidencia que el tiempo de cómputo aumenta significativamente a medida que se incrementa el número de elementos de malla, lo que indica que la configuración optima de malla también depende de la capacidad de cómputo que se tenga disponible.

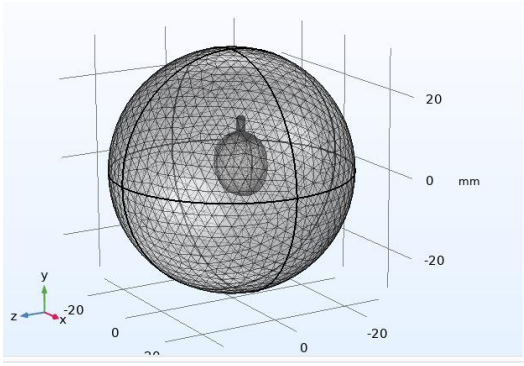


Figura 6.21: Malla simulación acústica estructural

6.9.4 Resultados y análisis

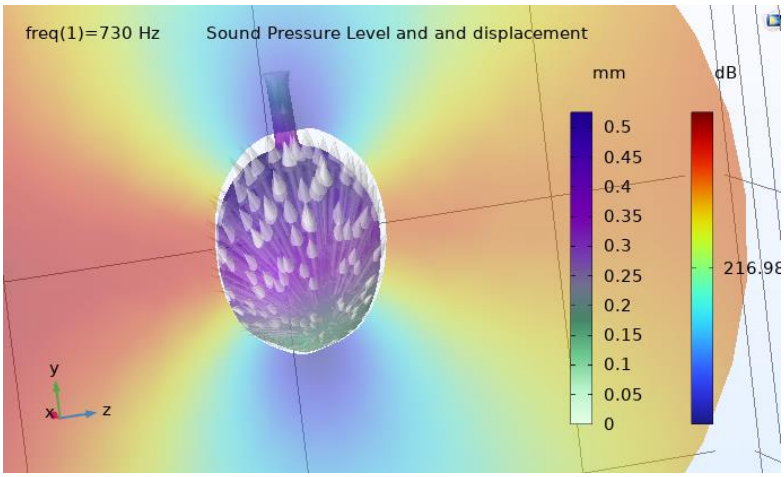


Figura 6.21: Resultados presión acústica y desplazamiento

De la anterior simulación se puede inferir para lograr desplazamientos significativos en el sistema fruto pedúnculo para los frutos maduros se requiere alrededor una onda incidente 2MPa

$$L_P = 20\log\left(p/p_o\right) \tag{6.2}$$

Resolviendo la ecuación 6.2 en términos de la potencia necesaria en un altavoz sería necesario un altavoz o una serie de altavoces que sumen una potencia de 33.5 MW

Esta potencia es 13400 veces mayor al altavoz del presente objeto de estudio

Capítulo 7

7 Conclusiones y trabajo futuro

7.1 Conclusiones

Después de la finalización del diseño del sistema y su posterior desarrollo, llega el momento de realizar una valoración. Con base en las pruebas realizadas, se puede afirmar que los objetivos impuestos durante el proyecto se han cumplido satisfactoriamente.

En primer lugar, el sistema implementado puedo detectar los frutos en imágenes, siendo capaz de clasificar los diferentes estados de maduración y realizando las cajas delimitadoras de manera precisa.

Gracias a la anterior funcionalidad el sistema tiene un potencial enorme, pudiéndose aplicar a drones para la cosecha automática ya que estos sistemas requieren un target con una confianza alta.

El estudio de la fuente acústica supone un estudio muy enriquecedor de todos los fenómenos acústicos y frecuencias específicas para los frutos maduros, sin embargo, con las características actuales no es posible alcanza la fuerza de desprendimiento que requiere el sistema fruto pedúnculo.

En resumen, este proyecto se ha investigado y puesto en común diferentes ramas de la visión artificial y aprendizaje profundo, culminando en la implementación de un sistema de detección y clasificación de los diferentes estados de maduración de café. No obstante, este no era el único objetivo, caracterizar un sistema acústico y evidenciar las frecuencias distintivas de cada estado de maduración también fue desarrollado como complemento del sistema de detección.

Además, median el análisis y simulación mediante elementos finitos el estudio de la fuente acústica supone un estudio muy enriquecedor de todos los fenómenos acústicos y frecuencias

específicas para los frutos maduros, sin embargo, con las características actuales no es posible alcanza la fuerza de desprendimiento que requiere el sistema fruto pedúnculo. Ademan también se evidencio que existen geometrías especificas en la cuales para una frecuencia especifica se logran aumentos significativos en la presión acústica.

7.2 Trabajo futuro

Si bien el sistema está funcionando correctamente en el entorno en el que se ha probado, existen algunas redes que podrían presentar mejores resultados, no obstante, requieren un coste computacional muy grande para la inferencia.

Una aplicación móvil es unos de los pasos importantes en el desarrollo de sistemas de visón artificial lo que hace sencillo y versátil la manera de implementar el modelo en un ambiente deseado y no solo restringido a un sistema de carga de fotos.

Otra vertiente del trabajo futuro seria la etiqueta del *dataset* automática lo que disminuiría exponencialmente el trabajo requerido ya que el proceso manual es minucioso y requiere una cantidad de tiempo considerable

El campo de la visión artificial se encuentra en constante cambio y crecimiento uno de los grandes puntos de trabajo futuro, sería la investigación de nuevos algoritmos que se vayan desarrollando, como por ejemplo el yolov5-lite que apenas está en versión de prueba y puede presentar resultados enormes.

Un aproximamiento optimo y que aun esta es desarrollo por la universidad autónoma de Manizales es uso de aire comprimido a determinada presión para lograr el desprendimiento de fruto de café arábica, ellos han desarrollado un sistema que canaliza el aire comprimido y lo concentra en una región especifica lo que supondría un aproximamiento más real a la hora de conseguir la fuerza de desprendimiento necesaria

Bibliografía

[1] L. Figueiredo, I. Jesus, J. A. T. Machado, J. R. Ferreira, and J. L. Martins de Carvalho, "Towards the development of intelligent transportation systems," in ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585), 2001, pp. 1206–1211.

[2] F. Torres, G. Barros, and M. J. Barros, "Computer vision classifier and platform for automatic counting: More than cars," 2017 IEEE 2nd Ecuador Tech. Chapters Meet. ETCM 2017, pp. 1–6, 2017.

[3] E. Soroush, A. Mirzaei, and S. Kamkar, "Near Real-Time Vehicle Detection and Tracking in Highways," no. March 2017, p. 11, 2016.

[4] A.-O. Fulop and L. Tamas, "Lessons learned from lightweight CNN based object recognition for mobile robots," 2018.

[5] P. Abrahamsson et al., "Affordable and energy-efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment," Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom, vol. 2, pp. 170–175, 2013.

[6] D. Pena, A. Forembski, X. Xu, and D. Moloney, "Benchmarking of CNNs for Low-Cost, Low-Power Robotics Applications," 2017.

[7] J. Huang et al., "Speed/accuracy trade-offs for modern convolutional object detectors," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 3296–3305, 2017.

[8] Ministerio del Trabajo, "Incremento del Salario Básico Unificado 2019.," 2018. [Online]. Available: http://www.trabajo.gob.ec/incremento-del-salario-basico-unificado-2019/. [Accessed: 10-Sep-2019].

[9] Z. Chen, T. Ellis, and S. A. Velastin, "Vehicle detection, tracking and classification in urban traffic," IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC, pp. 951–956, 2012.

[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-Decem, pp. 779–788, 2016.

[11] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017, vol. 2017-Janua, pp. 6517–6525, 2017.

[12] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018.

[13] W. Liu et al., "SSD: Single shot multibox detector," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9905 LNCS, pp. 21–37, 2016.

[14] R. Girshick, "Fast R-CNN," Proc. IEEE Int. Conf. Comput. Vis., vol. 2015 Inter, pp. 1440–1448, 2015.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, 2017.

[16] M. J. Shaifee, B. Chywl, F. Li, and A. Wong, "Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video," J. Comput. Vis. Imaging Syst., vol. 3, no. 1, 2017.

[17] L. Zhang, L. Lin, X. Liang, and K. He, "Is faster R-CNN doing well for pedestrian detection?," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9906 LNCS, pp. 443–457, 2016.

[18] Y. Jia et al., "Caffe: Convolutional architecture for fast feature embedding," MM 2014 - Proc. 2014 ACM Conf. Multimed., vol. abs/1506.0, pp. 675–678, 2014.

[19] P. B. Martín Abadi et al., "TensorFlow: A system for large-scale machine learning," Methods Enzymol., 2016.

[20] Y. LeCun et al., “Backpropagation Applied to Handwritten Zip Code Recognition,” Neural Comput., vol. 1, no. 4, pp. 541–551, 1989.
[21] Chuanqi305, “MobileNet-SSD,” 2019. [Online]. Available: https://github.com/chuanqi305/MobileNet-SSD. [Accessed: 10-Sep- 2019].
[22] A. Rosebrock, “Real-time object detection on the Raspberry Pi with the Movidius NCS,” 2018. [Online]. Available: https://www.pyimagesearch.com/2018/02/19/real-time-object-detection-on-the-raspberry-pi-with-the-movidius-ncs/. [Accessed: 10-Sep-2019].
[23] Taehoonlee, “High level network definitions with pre-trained weights in TensorFlow,” 2019. [Online]. Available: https://github.com/taehoonlee/tensornets. [Accessed: 10-Sep-2019].
[24] K. Hyodo, “YoloV3,” 2019. [Online]. Available: https://github.com/PINTO0309. [Accessed: 10-Sep-2019].
[25] J. Redmon and A. Farhadi, “Tiny YOLOv3,” 2018. [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed: 10-Sep- 2019].
[26] J. Hui, “SSD object detection: Single Shot MultiBox Detector for real-time processing,” 2018. [Online]. Available: https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing- 9bd8deac0e06. [Accessed: 10-Sep-2019].
[27] CyberAILab, “A Closer Look at YOLOv3,” 2018. [Online]. Available: https://www.cyberailab.com/home/a-closer-look-at-yolov3. [Accessed: 10-Sep-2019].
[28] T. Y. Lin et al., “Microsoft COCO: Common objects in context,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.
[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” Int. J. Comput. Vis., vol. 88, no. 2, pp. 303–338, 2010.
[30] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes Challenge 2012 (VOC2012),” 2012. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/. [Accessed: 10-Sep-2019].
[31] A. Rosebrock, “Simple object tracking with OpenCV,” 2018. [Online]. Available: https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/. [Accessed: 10-Sep-2019].
[32] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg, “Accurate scale estimation for robust visual tracking,” BMVC 2014 - Proc. Br. Mach. Vis. Conf. 2014, 2014.
[33] D. King, “Dlib C++ Library Python API,” 2019. [Online]. Available: http://dlib.net/python/index.html. [Accessed: 10-Oct-2019].
[34] Intel, “Deep Learning For Computer Vision,” 2019. [Online]. Available: https://software.intel.com/en-us/openvino-toolkit/deep-

learning-cv. [Accessed: 10-Oct-2019].
[35] Intel, “Intel® Neural Compute Stick 2,” 2019. [Online]. Available: https://software.intel.com/en-us/neural-compute-stick. [Accessed: 10-Oct-2019].
[36] Raspberry Pi, “Raspberry Pi 3 Model B+,” 2018. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b- plus/. [Accessed: 10-Oct-2019].
[37] Python, “About Python,” 2019. [Online]. Available: https://www.python.org/about/. [Accessed: 10-Oct-2019].
[38] A. Rosebrock, “Faster video file FPS with cv2.VideoCapture and OpenCV,” 2017. [Online]. Available: https://www.pyimagesearch.com/2017/02/06/faster-video-file-fps-with-cv2-videocapture-and-opencv/. [Accessed: 10-Oct-2019].

Anexos

lr0: 0.01 # initial learning rate (SGD=1E-2, Adam=1E-3)

left: 0.2 # final Nonrecycler learning rate (lr0 * lrf)

momentum: 0.937 # SGD momentum/Adam beta1

weight_decay: 0.0005 # optimizer weight decay 5e-4

warmup_epochs: 3.0 # warmup epochs (fractions ok)

warmup_momentum: 0.8 # warmup initial momentum

warmup_bias_lr: 0.1 # warmup initial bias lr

box: 0.05 # box loss gain

cls: 0.5 # cls loss gain

cls_pw: 1.0 # cls BCELoss positive_weight

obj: 1.0 # obj loss gain (scale with pixels)

obj_pw: 1.0 # obj BCELoss positive_weight

iou_t: 0.20 # IoU training threshold

anchor_t: 4.0 # anchor-multiple threshold

anchors: 3 # anchors per output layer (0 to ignore)

fl_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)

hsv_h: 0.015 # image HSV-Hue augmentation (fraction)

hsv_s: 0.7 # image HSV-Saturation augmentation (fraction)

hsv_v: 0.4 # image HSV-Value augmentation (fraction)

degrees: 0.0 # image rotation (+/- deg)

translate: 0.1 # image translation (+/- fraction)

scale: 0.5 # image scale (+/- gain)

shear: 0.0 # image shear (+/- deg)

perspective: 0.0 # image perspective (+/- fraction), range 0-0.001

flipud: 0.0 # image flip up-down (probability)

fliplr: 0.5 # image flip left-right (probability)

mosaic: 1.0 # image mosaic (probability)

mixup: 0.0 # image mixup (probability)