introducción:

Según la FAO mil millones de personas en el mundo pasan hambre mientras se desperdicia un tercio de los alimentos producidos en el mundo, lo que implica un grave problema ambiental . En el mundo una de las soluciones que se han implementados para esta problemática es la creación de bancos de alimentos.En Colombia se han creado hasta ahora 19 bancos de alimentos, los cuales “son organizaciones solidarias sin ánimo de lucro, que contribuyen a reducir el hambre y la desnutrición en el mundo, por medio de recepción de alimentos excedentarios del sector agropecuario, industrial, comercial, hoteles, restaurantes y/o personas naturales, para su debida distribución entre población en situación de vulnerabilidad.” (Asociación de Bancos de Alimentos de Colombia, s.f.) Propuesta de Trabajo de Grado Modalidad Exploratoria 3 Así los BA junto con la cooperación de 448 empresas, recuperan cientos de toneladas de alimentos en condiciones de consumo humano. Estas organizaciones se han agremiado en la Asociación Colombiana de Bancos de Alimentos (Abaco), la cual es una organización sin ánimo de lucro, que se encarga de representar, fortalecer las actividades, gestionar los recursos financieros, físicos y técnicos y reducir los desperdicios de alimentos en los bancos de alimentos y sus asociados. Deben contar con la logística requerida para la consecución, recepción, almacenamiento, separación, clasificación, conservación y distribución de los alimentos recibidos en donación; adicionalmente que cumplan los procesos misionales de gestión de donantes, productos, beneficiarios y sus estándares de calidad relacionados con las Buenas Prácticas de Manufactura y operación orientada al rescate de alimentos que: • Evidencian fallas en su presentación. • Su fecha de vencimiento es cercana. • Presentan baja rotación. • El estado de maduración exige pronto consumo. • Las empresas desean que lleguen a la población que necesita el bien de manera segura. (Asociación de Bancos de Alimentos de Colombia, s.f.) El objetivo de estas organizaciones finalmente es combatir el hambre, la mala alimentación y reducir las pérdidas de alimentos, definida según la Organización de las Naciones Unidas para la Alimentación y la Agricultura (FAO) como la disminución de la cantidad o calidad de los alimentos; donde gran parte de las pérdidas de alimentos son desperdicios, alimentos destinadas al consumo, que por mal manejo terminan desechados. Para reducir las pérdidas y desperdicios de alimentos los BA desarrollan tres actividades principales: La gestión de donantes: donde hay tres tipos de donaciones, que pueden ser donaciones de producto, de dinero o de tiempo. Gestión de productos: en este proceso se hace la recepción, acondicionamiento, saneamiento, manipulación y distribución de los productos, para luego repartirlas a las instituciones beneficiadas, en buen estado. Gestión de beneficiarios: los bancos de alimentos benefician a otras instituciones, este proceso requiere una serie de acciones como la selección, vinculación, atención, acompañamiento y seguimiento, suspensión o desvinculación de las mismas.

El machine learning es la rama de la inteligencia artificial que busca como dotar a las máquinas de la capacidad de aprendizaje entendida como
LA generalización DE CONOCIMIENTO A PARTIR DE UN CONJUNTO DE EXPERIENCIAS, lo cual la convierte en un componente nuclear que se intersecta con las otras áreas de inteligencia artificial.(Robótica, análisis del lenguaje natural, conversor voz-texto y texto voz y vision de maquina)
este aprendizaje puede dividirse en:

- Supervisado: El aprendizaje surge de mostrar qué valores se desean obtener dadas las entradas.

- No supervisado: A diferencia del supervisado, no se dan las entradas deseadas, lo que se espera es que el algoritmo por “iniciativa propia” agrupe los datos por categorías según patrones que vea en ellos, generando conocimientos solo con los datos de entrada.(clusterización)
- reforzado

Siendo los modelos versiones simplificadas de la realidad que a tra vez de escoger adecuadamente sus parámetros modelan los datos que se tienen y para los cuales se quiere crear ese modelo, el packing learning busca algoritmos que seteen de manera adecuada esos parámetros minimizando la función de error.

MARCO TEÓRICO

Modelo de regresión lineal

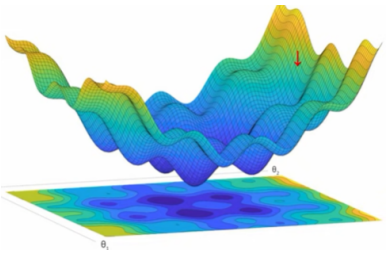
Imaginemos un modelo en el que la salida depende de y varios parámetros de entrada, para encontrar el hiperplano que mejor se ajuste a nuestros datos, se aplica el error cuadrático medio que penaliza con mayor severidad a los puntos o datos que se alejan más de nuestro modelo y cuya ecuación vectorial luce ascii, y al derivar e igualar a cero podemos encontrar su valor mínimo encontrando así los parámetros deseados para el modelo que mejor se ajustara

$$(Y - XW)^T(Y - XW)$$

pero qué sucede si la función de error es no-convexa, esta aproximación a la función de error ya no es tan útil pues habrá puntos en los que la derivada será 0 sin que sea un mínimo global

marco teórico

Metodo del descenso del gradiente

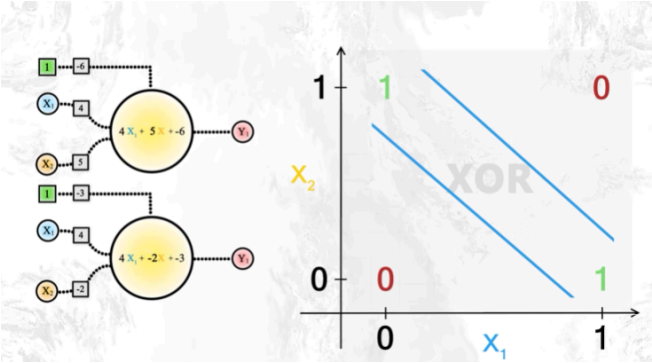


suponiendo que estamos en una posición cualquiera, determinamos derivadas parciales para cada uno de los parámetros, que en conjunto forman un vector denominado gradiente, que indica hacia donde aumenta la pendiente, por tanto este será restado a los parámetros par ir de poco en poco cuesta abajo acá entonces aparece un nuevo concepto la tasa de

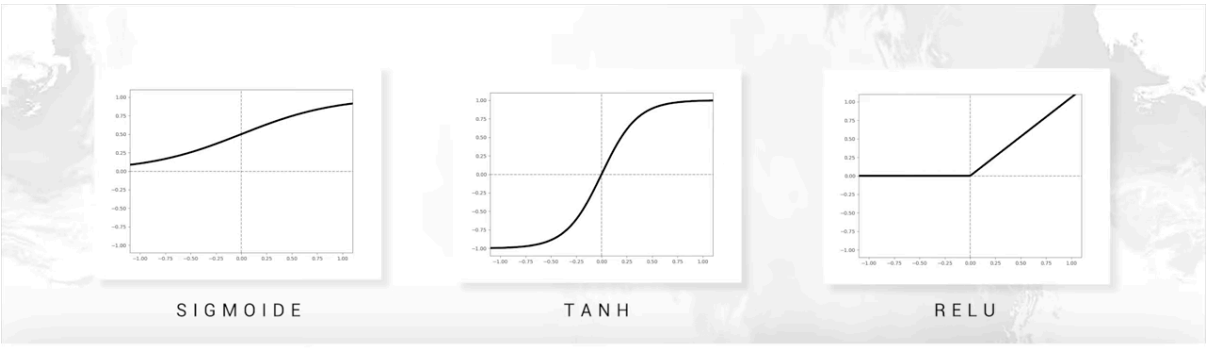
aprendizaje, este indica cuanto se desciende en la dirección del gradiente antes de re calcularlo este es fundamental pues si es muy pequeño es más fácil caer en en un mínimo local, si es muy grande, pasamos de un máximo a otro.

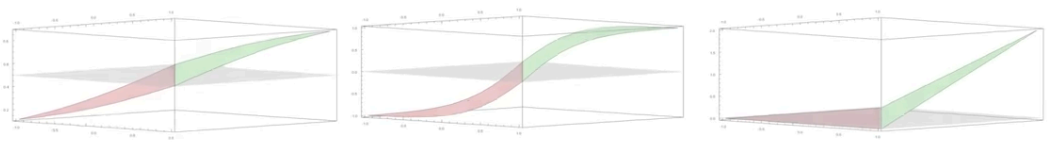
La neurona

la neurona imita el comportamiento de un algoritmo de regresión lineal, a través del cual intenta encontrar una frontera entre las salidas no deseadas y las salidas deseadas, esto tiene sus limitantes por supuesto, así, para resolver el problema de una compuerta xor se necesitan 2 neuronas conectadas en paralelo

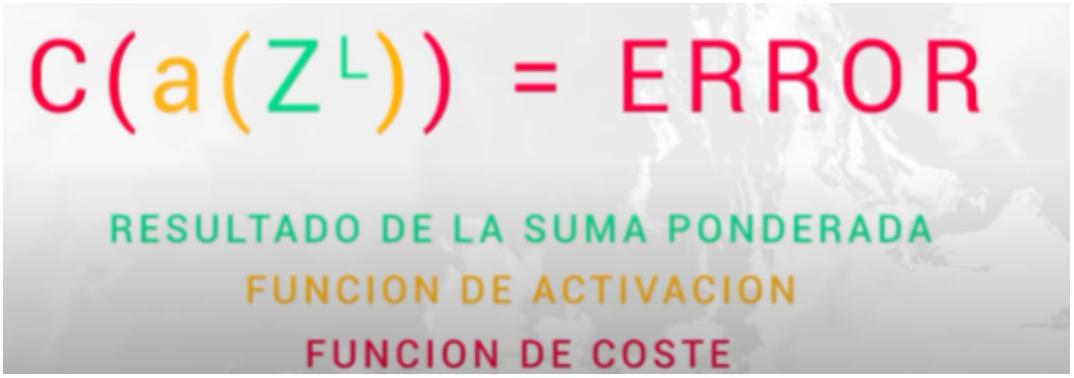


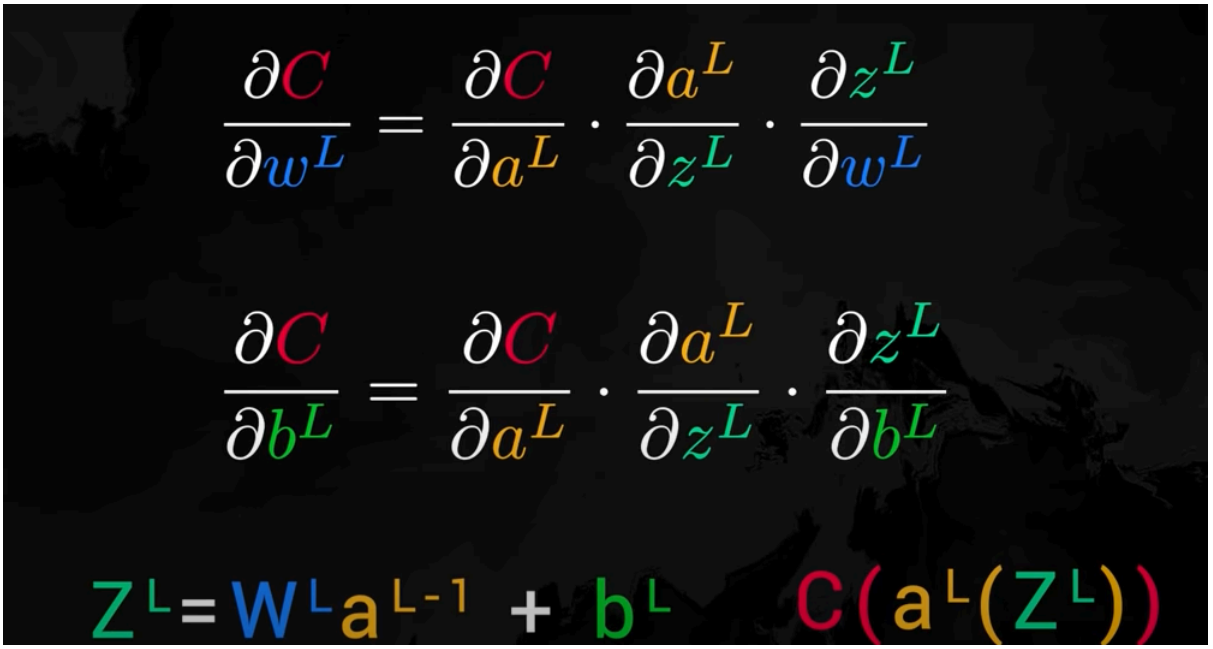
La funcion de activacion
sin embargo estas neuronas hacen procesos lineales , si acumulamos muchas capas de estas seria como si solo tuvieramos una capa lineal. asique para poder resolver problemas no lineales , como la construccion de una frontera en forma decirculo entre 2 tipos de datos necesitamos que estas neuronas hagan procesos no lineales, asi las funciones de activacion rean deformaciones no lineales a la salida que permiten esto, a continuacion las mas utilizadas y las deformaciones que aplicaron al plano generado or la neurona

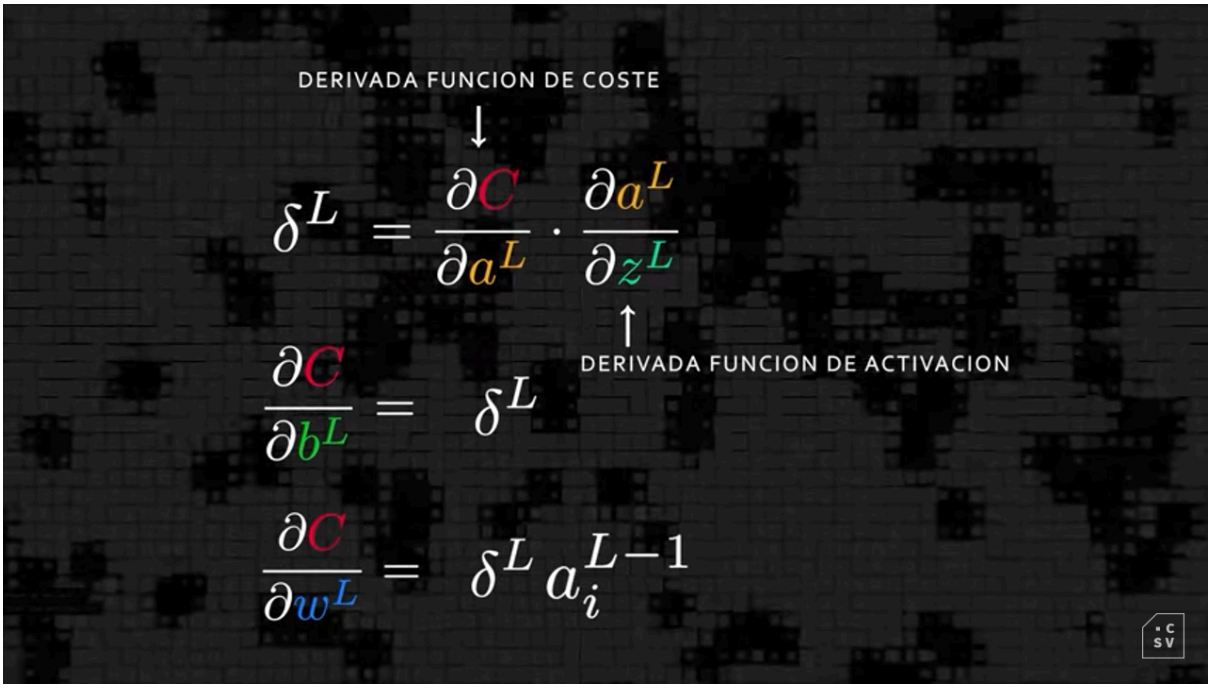




por

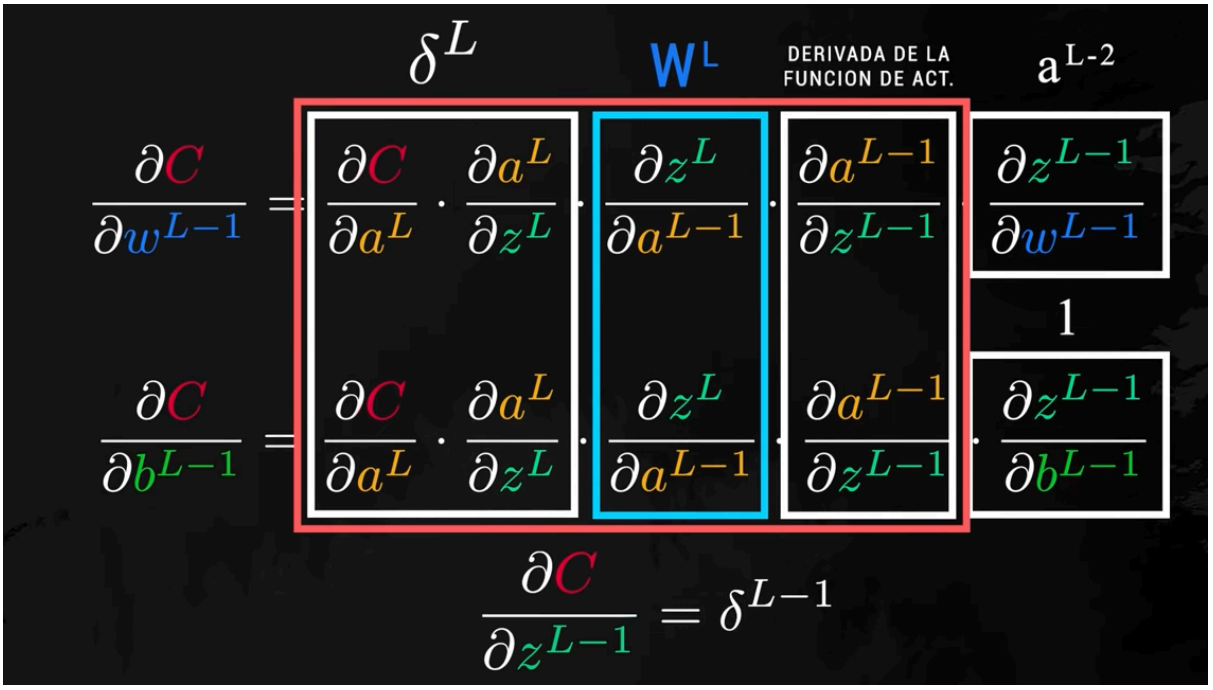




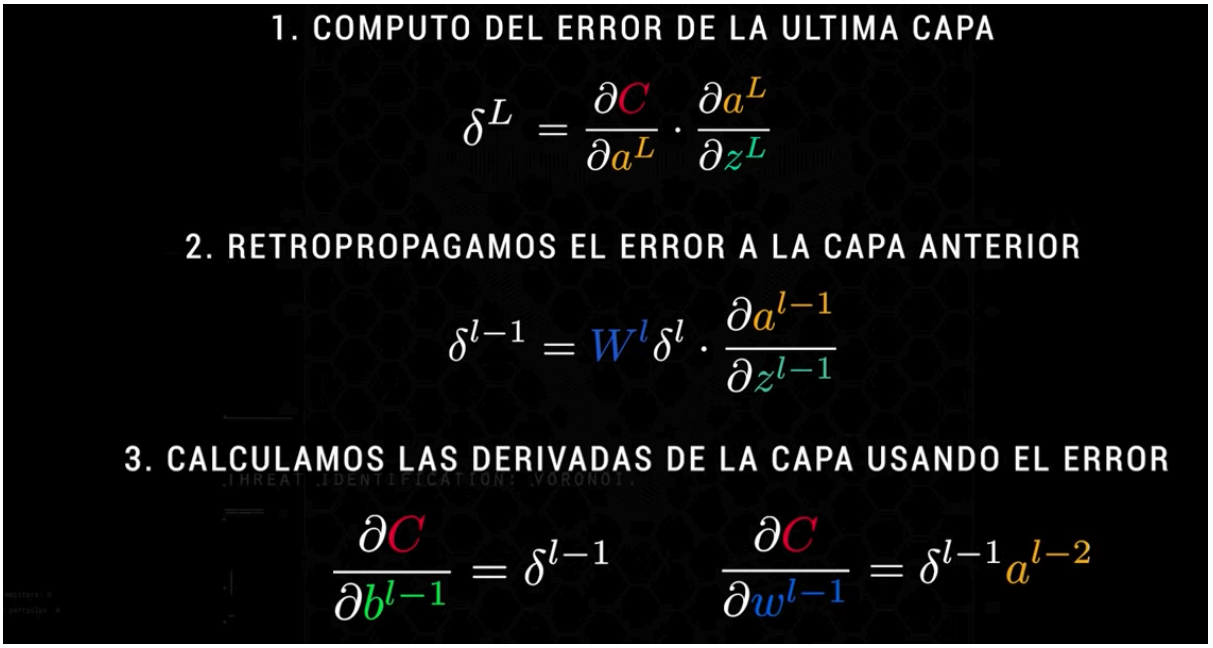


sigma es el error imputado a esa capa es y al intentar realizar el mismo procedimiento para ña capa previa encontramos que WL es la matriz que nos habla de como se comporta la

suma ponderada de una capa cuando se paria el output de una de las neuronas de la capa previa



asi el algoritmo se puede resumir en los siguientes 3 pasos



Overfitting

Se refiere a el fenómeno en el que la inteligencia artificial se ajusta tan bien a los datos de entrenamiento que pierde flexibilidad por lo que su conocimiento deja de ser generalizable lo cual hace que al hacer el testing el desempeño de la red sea bastante malo, aun cuando en el entrenamiento era muy bueno.

Es una buena práctica para evitar esto tomar el total de nuestros datos, de manera desorganizada y dividirlos en dos grupos: un grupo de testeo y otro de entrenamiento. si se entrena al tiempo que se testea es esperable que el error sea más bajo en training que en testeo y que ambos bajen continuamente en el punto en que el error de testeo empieza a empeorar cuando el de entrenamiento sigue disminuyendo es ahí cuando se empieza a entrar en esta fase de overfitting

Primeras aproximaciones al problema con opencv y propuesta de solución

A Través de procesos de inteligencia artificial se pueda identificar el tipo de alimento , y su estado así como un aproximado de “tiempo útil del alimento” para que esto sea en últimas un criterio que haga más eficiente los procesos de distribución inicialmente. En una primera aproximación al problema se toman datasets de kaggle de manzanas y bananos en estado maduro y en descomposición y se analizan características como el histograma de cada uno de los canales , posteriormente se pasa a escala de grises la imagen se le aplica un suavizado con un filtro gaussiano para eliminar posible ruido y facilitar la posterior detección de bordes con el algoritmo de canny, posteriormente se hace binarización de la imagen con un threshold de 128, para facilitar el hallar contornos, las áreas de estos contornos por último se hallan momentos de la imagen y centroides de la misma . esto finalmente para tener una idea de las características que más importancia pueden tener a la hora de hacer clasificación con una red neuronal.

Finalmente después de una primera discusión de los resultados a este punto y de la idea con el profesor se propone como solución: La creación de una red neuronal convolucional a través del framework pythorch que inicialmente pueda diferenciar entre 3 estados de la fruta, planteando los cimientos para una escalabilidad que permita el fácil entrenamiento de la red para que sea aplicable a varios alimentos perecederos

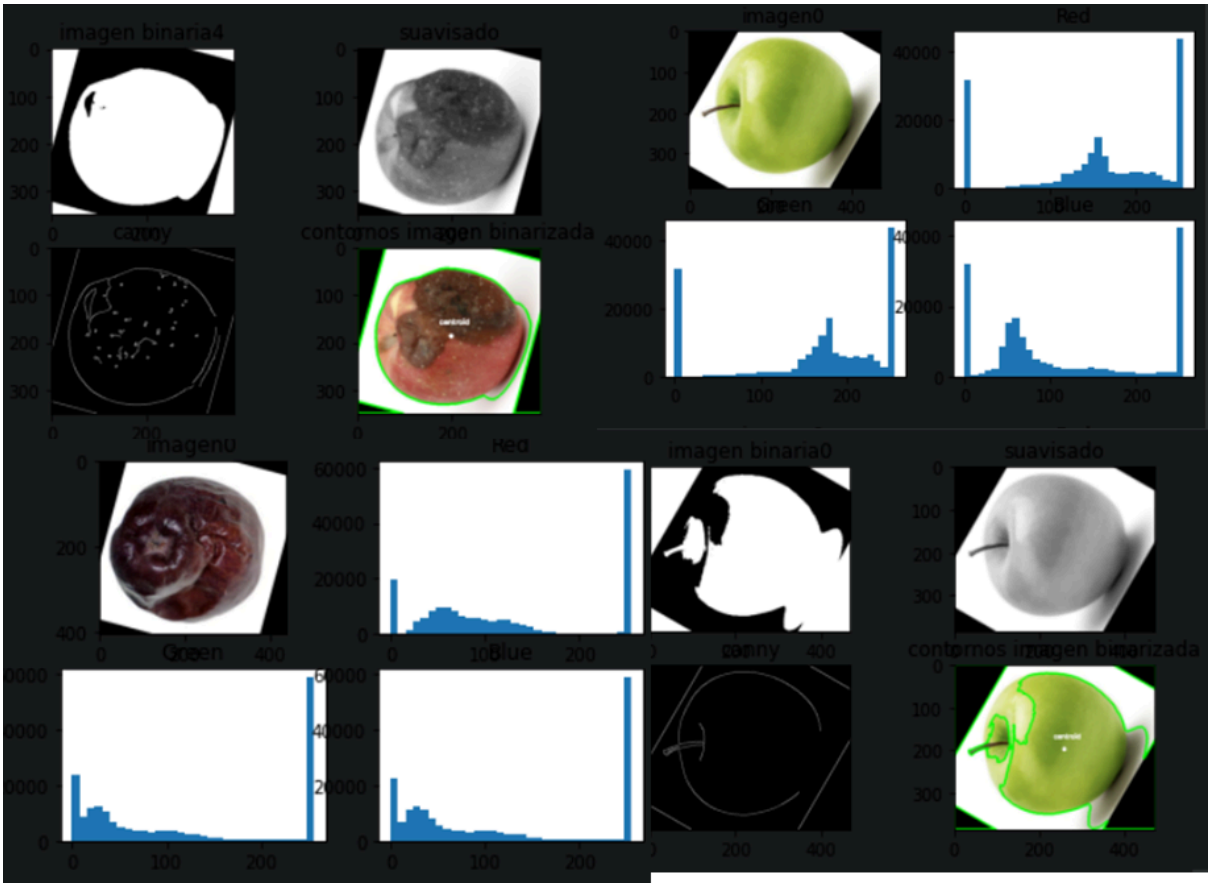


imagen binaria4
suavisado
imagen0
Red
canny
contornos imagen binarizada
Green
Blue
imagen0
Red
imagen binaria0
suavisado
Green
Blue
canny
contornos imagen binarizada

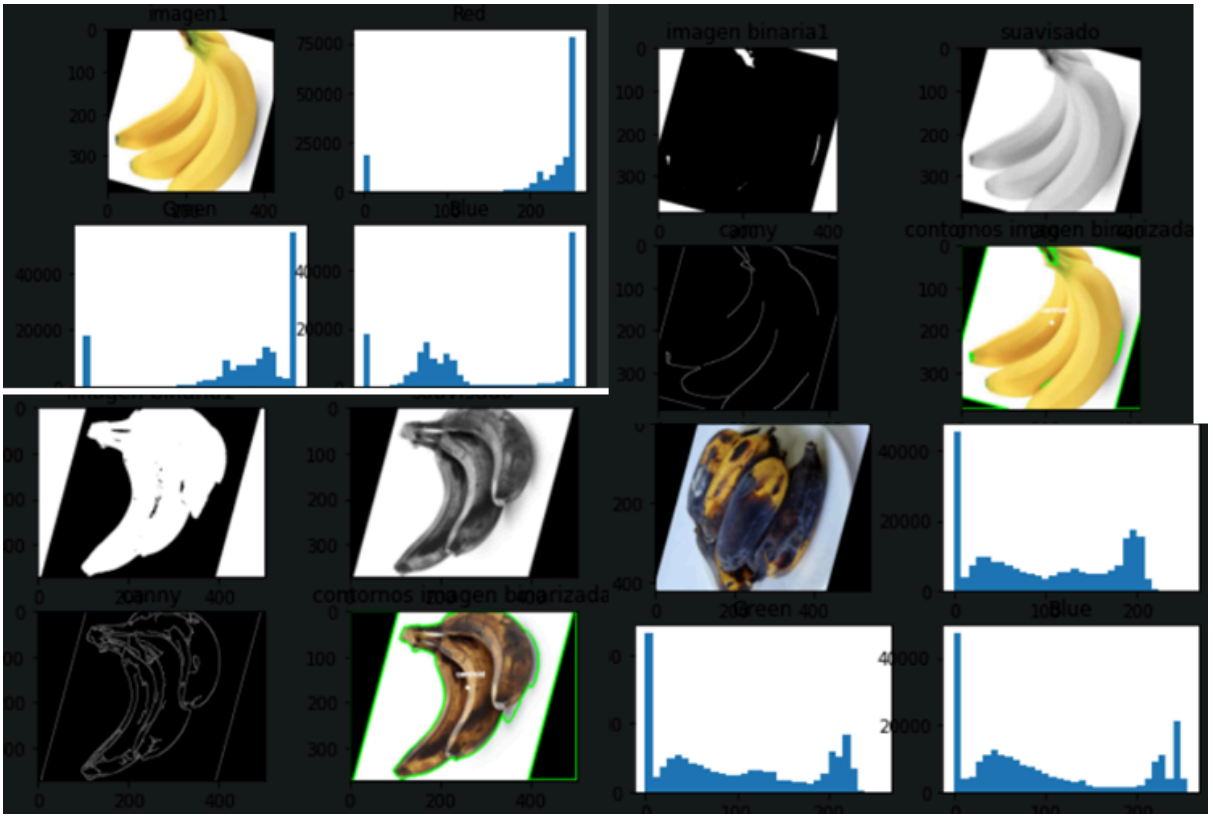


imagen1
Red
imagen binaria1
suavisado
Green
Blue
canny
contornos imagen binarizada
canny
contornos imagen binarizada
Green
Blue

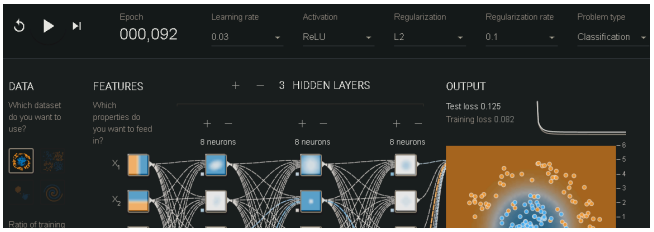
Procedimiento y Resultados

En los primeros dias no habia nada luego luis ordeno los datasets

Dataset y Datareaders

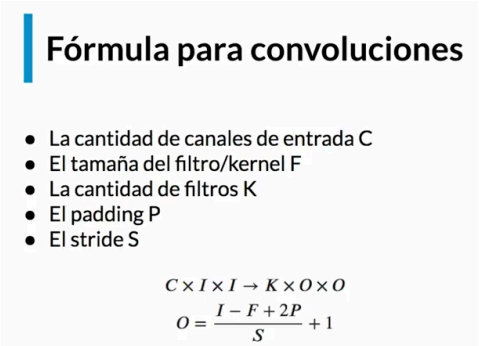
Una vez cargados los datasets, se hace split en 3 grupos aproximadamente 80% para trainning 10% y 10% para validación y testing,en el dataset de mandarinas (505 imágenes). en el de bananos por tener una mayor cantidad de data (4700 imágenes)se hace 98% trainning 1 % validation y 1% testing
En pytorch un dataloader se encarga de subdividir el datasets en mini baches, se usa el parámetro shuffle =True Esto es útil puesto que por la naturaleza de nuestro dataset y de la manera que fue cargado cabía la posibilidad de que existieran labels del mismo valor de manera consecutiva. Que esto no suceda se comprueba más adelante en el código graficando e imprimiendo los labels de uno de los batches. El batch size se elige de 32, lo recomendable es usar potencias de 2 en un rango de 2^1 -2^9 , por lo que 2^5 parece un buen punto medio de partida por último el parámetro workers se setea a un valor de 4 permitiendo hacer evaluación paralela de 4 subprocesos aumentando la velocidad del entrenamiento. y validación.

ARQUITECTURA Y PRIMEROS EXPERIMENTOS



Se construye una arquitectura convolucional de 3 capas ocultas compuesta por 32 neuronas cada una entre una capa fully connected qué es la de entrada y otra fully connected qué es la de salida y después de la cual se aplica una función de activación softmax ideal para problemas de clasificación multiclase. para asegurarnos de no modificar el tamaño de la imagen se elige un kernel size de 3 un padding de 1 y un stride de 1, esto se asegura para

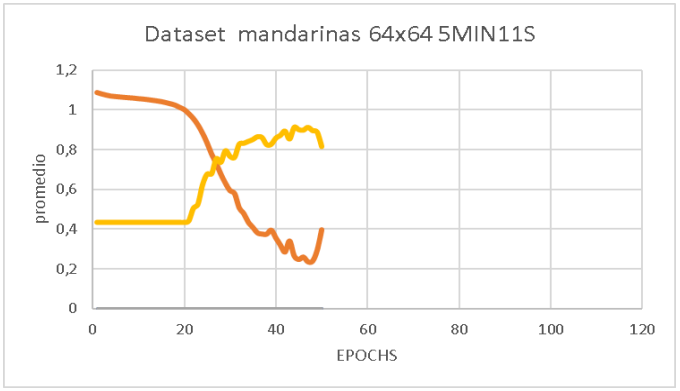
todos los tamaños de imagen usados pues todos son potencias de 2.

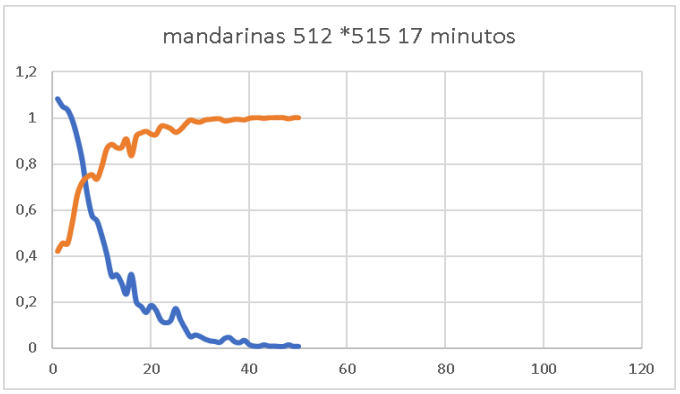


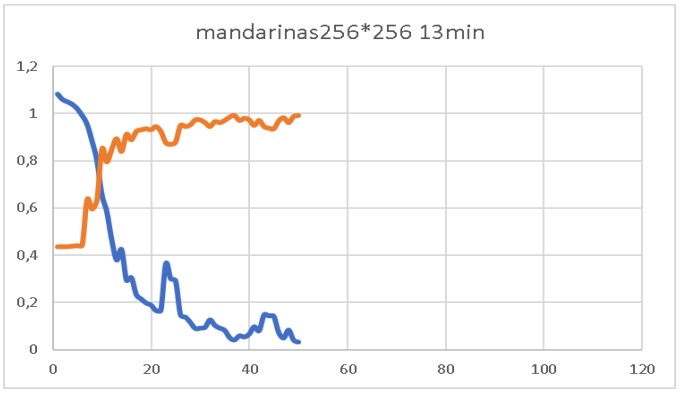
Posteriormente a cada capa convolucional se aplica un max pooling con un kernel size de 2. luego se hace un stanking . Luego se usa como función de activación para nuestras capas convolucionales la función relu con el fin de evitar el Vanishing Gradient Problem esto sucede porque filtra ya que al filtrar los inputs negativos. al llegar a la última capa fully connected la capa de clasificación multiclase se aplica una capa softmax para asegurar que la suma de probabilidades sea 1 haciendo que el entrenamiento converja más rápido. Por último la función de pérdida se modela utilizando negative log likelihood esto porque es el estándar en los problemas de clasificación multiclase, que es nuestro caso.
El tamaño de la red fue definido teniendo en cuenta, que no existe un estándar para la profundidad de redes, que a mayor profundidad mayor tiempo de entrenamiento por el problema de vanishing gradiente y la intuición que nos dio algunos algunos experimentos realizados en tensor flow playground playground.

Como optimizador se decidió usar Stochastic gradient descent, que añade aleatoriedad al proceso de GD y por tanto mejora el desempeño del algoritmo, además de ser la base para optimizadores más complejos y posiblemente veloces como adagraf, aladelta,adam.

Se hicieron entrenamientos con el dataset de mandarinas, reduciendo el tamaño inicial 1024x1024 pixels a 512,256,128 y 64 utilizando interpolación bilineal para tener una idea de hasta qué punto podemos permitirnos la “pérdida” de información con el fin de mejorar la velocidad de entrenamiento, y hacernos una idea de cuantas epochs eran necesarias para tener entrenamientos más o menos exitosos.







REGULARIZACIÓN Y OPTIMIZACIÓN DE HIPERPARAMETROS

Se aplicó Normalización y Estandari de los inputs a la primera capa, esto llevo los valores que podría tomar un píxel de un rango de 0 a 255 a un rango de 0 a 1, luego se aplicó una capa de Dropout a cada capa convolucional con un dropout rate de 0.5 y por tanto un factor de experticia F=2. Después de este dropout se agregan capas de bach normalization usa una media y una desviación estándar , luego multiplica por un parámetro aleatorio y luego suma otro parámetro aleatorio, estos 4 parámetros serán optimizados durante el entrenamiento.

Esto último con el fin de evitar una situación similar a la de exploiting gradient por los impactos que se podrían producir si al actualizar los pesos uno de los pesos fuese extremadamente grande comparada con los otros pesos de otras neuronas.

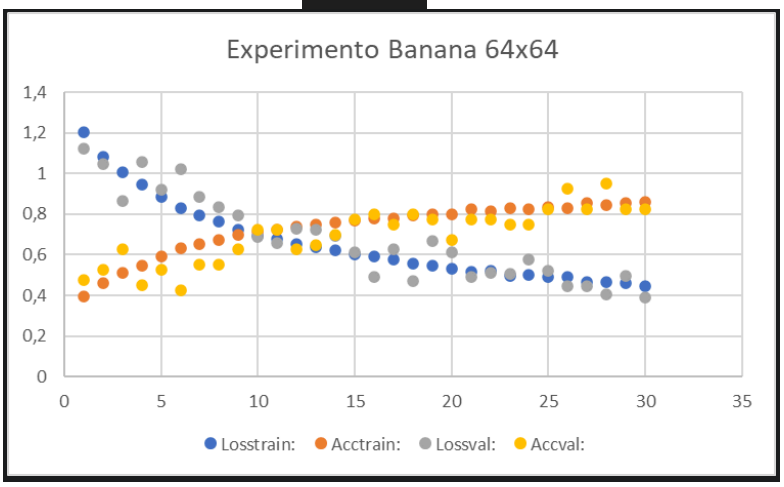
Posteriormente par ala optimización de hiperparametros se hizo investigacion de las tecnicas existentes encontrando.

Babysitting = que consiste en mover las perillas (los hiper parametros) de anera manual y encontrar los que mejor sirvan.

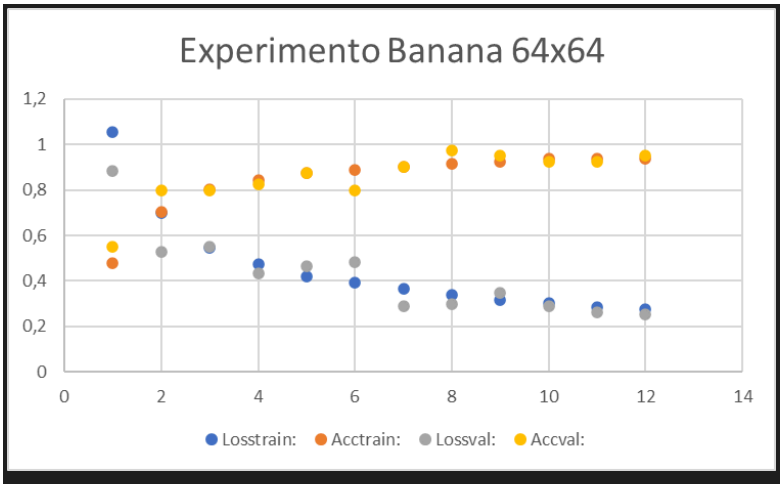
Grid search= identificar las n dimensiones de los hiperparametros el rango de cada uno y quede como una consola de perillas suele ser más útil en ML tradicional
Random search= permite testear valores según su importancia, definiendo no solo el rango sino también una buena escala. Batch size potencia de 2 learning rate de forma logarítmica. Este proceso suele ser bastante tardado. teniendo en cuenta Las limitaciones de tiempo sumadas con los problemas que se tuvo con el uso de la gpu de colab, se decidio guiados por la jerarquía de hiperparametros propuesta por Andrew Yan-Tak Ng es un profesor asociado en el departamento de Ciencias de la Computación y del departamento de Ingeniería Electrónica por cortesía de la Universidad de Stanford, y trabaja como director del laboratorio de Inteligencia Artificial en Stanford.

Se tomó el hiper parámetro de mayor jerarquía se hicieron pruebas de entrenamiento variando de manera logarítmica haciendo uso de la técnica babysitting lo cual nos permitió tener una idea del rango.

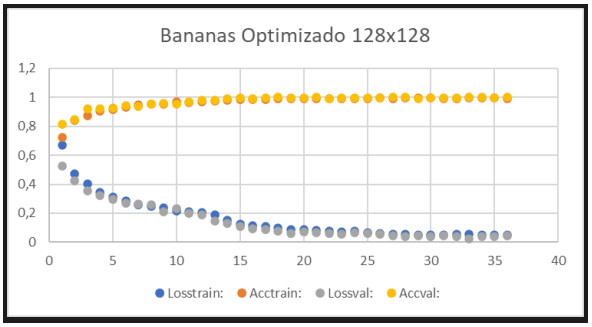
Learning rate `1e-6`



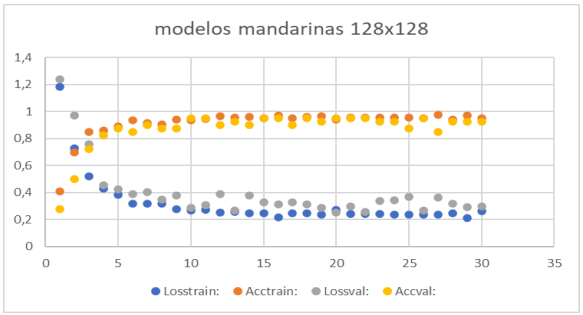
Learning rate `1e-5`

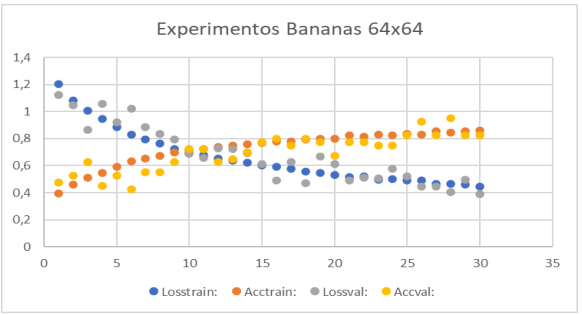


Posteriormente se hizo optimización del hiper parámetro de manera aleatoria en un rango definido



y de manera no aleatoria en el mismo rango





Conclusiones Y Aprendizajes

- Importante tener en cuenta las limitaciones de gpu con las que dispone la versión gratis de google colab. esto nos produjo algunos problemas durante el proceso del entrenamiento
- en caso de over o under fitting es importante revisar la calidad de los datos, y la distribución de estos en los batches de validación y testeo
- Es importante entender que no existe bala de plata ni librería perfecta, no habrá modelos que funcionen para absolutamente todos los casos y es importante entender el funcionamiento de las implementaciones que puedan tener cualquier librería
- Se tuvo problemas con la graficación de datos usando tensor board en google cola integrado con mytorch, en este caso es una buena práctica tener claro cómo se definen las funciones de métricas para imprimir los datos de manera adecuada por consola y a través de un archivo de texto hacer su graficación en excel o directamente a través de librerías como matplotlib
- Respecto al overfitting, con una época alrededor de 40, se comienza a estabilizar la pérdida.
- Entre más resolución tengan las imágenes a entrenar, más tiempo toma la red neuronal para entrenamiento pero así mismo tendrá la posibilidad de hacer distinción de detalles más finos en la forma de predecir, por ejemplo, con imágenes burdas distinguirá un estado de maduración del otro pero muy a groso modo. Con una mayor resolución puede distinguir que un banano esté más maduro que otro.
- Esperábamos un sobreentrenamiento en las imágenes de bananos de 128, al evaluarlo con el conjunto de datos de inferencia y revisar su desempeño este mejora

dependiendo de la cantidad de épocas que se emplee, el mejor resultado lo obtuvimos con 12 épocas.

- En general los modelos y la arquitectura parecen tener un buen fitting, el trabajo faltante consiste en refactorización de código para testing de diferentes modelos y deployment, además este informe es una buena guía para la adquisición de nuevos datasets, y el entrenamiento de diferentes modelos. por lo que el objetivo parece haber sido cumplido a satisfacción