UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: ***Análisis de la técnica de Transfer Learning en Machine Learning a través de un caso de estudio: La clasificación de productos en el Banco Alimentario de La Plata.***

AUTORES: ***Agustín De Luca y Matías Irigoitia***

DIRECTOR: ***Dra. Claudia Pons***

CODIRECTOR:

ASESOR PROFESIONAL:

CARRERA: **Licenciatura en Sistemas**

Resumen

La presente investigación propone analizar la técnica de transfer learning y la comparación de distintos modelos pre-entrenados, para determinar si estos pueden ser utilizados efectivamente en el caso de estudio, la clasificación de productos para el Banco Alimentario de La Plata, a partir del reconocimiento de imágenes.

Palabras Clave

Inteligencia artificial, Machine learning, Transfer learning, TensorFlow, Keras, Redes Neuronales Convolucionales, Dataset, Modelos pre-entrenados, COCO (common objects in context), MobileNet, Inception, RCNN, SSD, Python, Maxpooling.

Conclusiones

Se logró el objetivo de analizar la técnica de Transfer Learning en Aprendizaje Automático. Teniendo en cuenta los resultados obtenidos y las investigaciones realizadas, es posible afirmar que la aplicación de machine learning y transfer learning es factible de ser aplicada en el caso práctico propuesto, facilitando los procesos de entrada al sistema del Banco Alimentario. Aunque es de destacar que se necesita de una considerable inversión inicial para llevar a cabo la aplicación de este sistema y también que su mantenimiento es un proceso costoso.

Trabajos Realizados

Se analizó la técnica de Transfer Learning en Aprendizaje Automático.
Se hizo énfasis en modelos convolucionales. Se utilizo modelos pre-entrenados para detectar los objetos utilizando TensorFlow Object Detection como marco de trabajo.
Se desarrollaron pruebas para comparar los diferentes modelos pre-entrenados utilizando un dataset creado para esta tesina.
Se planteo una propuesta para el caso de estudio Banco Alimentario de La Plata.

Trabajos Futuros

En cuanto al algoritmo de detección de objetos, se recomienda hacer pruebas con una cantidad significativamente mayor de productos.
En cuanto al sistema, se puede extender el trabajo implementando el prototipo presentado. Agregar funciones como: exportar o importar los datos en otros formatos, analíticas, registro de entregas comederos y servicio de reportes.

Indices de Contenidos:

Índice de Figuras

Indice de Tablas

CAPÍTULO 1: INTRODUCCIÓN

1.1 Motivación

El aprendizaje automático o *machine learning* (su acrónimo en inglés, ML) es una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan, esto quiere decir, que logren mejorar su desempeño con la experiencia. De forma más concreta, los investigadores del ML buscan algoritmos y heurísticas para convertir muestras de datos en programas de computadora, sin tener que escribir los últimos explícitamente. Los modelos o programas resultantes deben ser capaces de generalizar comportamientos e inferencias para un conjunto más amplio de datos.

ML es un término del cual se está hablando e investigando desde hace décadas, pero no fue hasta estos últimos años, en que se ha observado un auge en las técnicas de aprendizaje automático, principalmente gracias a la disponibilidad de grandes volúmenes de datos y al gran avance de la capacidad de procesamiento disponibles, posibilitando de esa manera mantener algoritmos pre procesados, los cuales puedan ser usados en entornos reales más allá de las aplicaciones en entornos controlados como es el caso de los laboratorios.

Junto con el resurgimiento del ML también se han intensificado las investigaciones alrededor de numerosas técnicas y estrategias vinculadas. En particular, la técnica de transfer learning (su acrónimo en inglés, TL) es un problema de investigación en aprendizaje automático que se enfoca en almacenar el conocimiento adquirido al resolver un problema para aplicarlo a un problema diferente pero relacionado.

Los avances tecnológicos que vienen ocurriendo en los últimos años, han permitido que diferentes áreas o dominios se vean beneficiados, agilizando así por ejemplo tareas que de otra manera se realizaban de forma manual, como ejemplo de ello se puede ver que un área en donde ha impactado la tecnología es en el reconocimiento de productos para su clasificación y/o contabilización.

Con el objetivo de analizar la técnica de Transfer Learning en Aprendizaje Automático llevaremos adelante un caso de estudio: la Clasificación de Productos para el Banco Alimentario de La Plata (http://bancoalimentario.org.ar), el cual recibe y distribuye anualmente más de 500.000 toneladas de alimentos, y tiene por objetivo disminuir el hambre, la desnutrición y las malas prácticas alimentarias en la región, mediante el recupero de alimentos, para ser distribuidos a organizaciones comunitarias que prestan servicio alimentario a sectores necesitados
Hoy en día dicho Banco Alimentario aplica procesos manuales o con código de barras para realizar ingresos de productos, en lo cual vemos una veta clara para la aplicación de algoritmos de detección automática, en búsqueda de agilizar el proceso.

Existen distintas aplicaciones que brindan antecedentes sobre sistemas inteligentes para el procesamiento de productos de manera autónoma. En particular, algunas de estas aplicaciones son:

Scan&Go: En la región está funcionando el sistema inteligente de Scan&Go del supermercado Jumbo Cencosud, el cual utiliza código de barras para detectar los SKU[1] uno por uno. Este proceso puede ser mejorado, reemplazando la detección por código de barra por el reconocimiento de objetos a través de imágenes, utilizando algoritmos de aprendizaje supervisado.

DrishyamAI: Drishyam AI es una plataforma que brinda servicio a empresas para la identificación de productos basada en el reconocimiento de imágenes. Drishyam AI simplifica la atención al cliente y ayuda a las empresas a reducir los costos de soporte utilizando su plataforma de reconocimiento de imágenes basada en Deep Learning [2]para automatizar y agilizar el proceso de identificación de productos. La tecnología de reconocimiento de imágenes de Drishyam AI se puede incrustar o integrar en la aplicación de la empresa que contrate el servicio, o sus clientes pueden simplemente enviar fotos de sus productos por mensaje de texto (SMS) a Drishyam AI.

Smart Self-Checkout: Existe una empresa de origen Taiwanés NexCobot, que posee desarrolladores en China Taiwán, y US, esta empresa pertenece a la corporación Nexcom. La cual en su catálogo de servicios y productos posee un producto de Smart Self-Checkout, que implementa reconocimiento de productos a través de imágenes, para agilizar el procesamiento de productos ya sea en supermercados tiendas o sitios como el Banco de alimentos los cuales poseen la problemática de recibir grandes cantidades de productos diarios y tener que llevar un proceso de stock.

1.2 Objetivo

El objetivo general de esta tesina es el análisis de la técnica de Transfer Learning en Aprendizaje Automático a través de un caso de estudio, la Clasificación de Productos para el Banco Alimentario de La Plata (http://bancoalimentario.org.ar).
Particularmente se hará énfasis en algoritmos de aprendizaje automático, utilizando redes convolucionales[3] para detectar los objetos, clasificarlos y disponerlos para el consumo de

[1] SKU: El número/código de referencia (en inglés *stock keeping unit*,), también referido como código de artículo, es un número o código asignado a un elemento para poder identificarlo en el inventario físico o financiero, así como para referencias otros tipos de servicios.

[2] Deep learning: es un conjunto de algoritmos de *machine learning* que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial

[3] Redes convolucionales: es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico. Este tipo de red es una variación de un perceptron multicapa sin embargo, debido a que su aplicación es realizada en matrices bidimensionales, son muy efectivas para tareas de vision artificial como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

servicios, utilizando un *framework open source* como base para el entrenamiento y aprendizaje, el cual provee modelos pre entrenados para maximizar la eficacia y rehusar el conocimiento.
Para poder diseñar el prototipo de la aplicación de reconocimiento de productos a través de imágenes para el Banco Alimentario se realizaron entrevistas al director del Banco, Sr. Pedro Elizalde.
Desde el comienzo de la investigación hasta la finalización del prototipo del sistema de reconocimiento de objetos a través de imágenes, se fueron registrando las distintas lecciones aprendidas y problemas que se presentaron, los cuales han sido transcriptos en la tesina; en pos de que sean útiles para futuros desarrollos de este estilo.

1.3 Resultados Esperados

Durante el desarrollo de esta tesina se realizaron las siguientes acciones:

- Investigar sobre la técnica de Transfer Learning (TL) en aprendizaje automático; en particular TL sobre Redes Neuronales Convolucionales.

- Investigar sobre los modelos pre entrenados utilizados para este desarrollo, compararlos, evaluar su rendimiento y su capacidad de aprendizaje.

- Indagar sobre el Banco Alimentario de La Plata y como es su sistema actual para el ingreso de los productos.

Con los objetivos de:

- Proveer un prototipo para de detección de productos en base a imágenes, detallando el proceso de desarrollo, como la preparación del dataset[4] y entrenamiento. Dicho prototipo podrá ser adaptado para el consumo de una aplicación en dominios específicos, en particular para el Banco Alimentario, viéndolo, así como una alternativa al código de barra y al QR, aplicable también en cajas rápidas autónomas de procesamiento de productos.

- Crear entornos de pruebas para analizar estos modelos, de tal manera de entender sus particularidades y poder concluir formalmente sobre qué modelo es el mejor para dicha aplicación.

[4] Dataset: Colección de datos habitualmente tabulada.

1.4 Estructura de la tesina

La estructura de la tesina se describe a continuación:

El capítulo 2 está dedicado a la técnica de *Transfer Learning* y los modelos pre-entrenados. En este se describe qué es la técnica, cuál es el motivo y por qué es conveniente aplicarla. Se la compara con las técnicas de *Machine Learning*. Para finalizar el tema *Transfer Learning*, se explican las distintas estrategias y técnicas que pueden aplicarse en función del dominio, la tarea en cuestión y la disponibilidad de datos. Por último, se expone qué son los modelos pre-entrenados y las cuestiones a tener en cuenta al elegir un modelo pre-entrenado.

En el capítulo 3 se describe el caso de estudio de la presente tesina. El cual se presenta de acuerdo con la información extraída de las entrevistas con el director del Banco Alimentario de La Plata, Pedro Elizalde. Se detalla, qué es el Banco Alimentario, qué función cumple, cómo es su sistema actual para el procesamiento de mercaderías y cuáles son sus problemáticas.

En el capítulo 4 se explica en profundidad la técnica de ML, explicando características de cada modelo disponible para el *framework* TensorFlow, haciendo énfasis en redes de tipo RCNN y SSD, así también en extractores como *mobileNete inception*. Luego se ejecuta una prueba sobre cada modelo en circunstancias controlables detalladas en el presente capítulo, y de esta manera, se compara el entrenamiento de distintos modelos para determinar cuál obtiene mejores resultados, y así el más óptimo para seguir adelante con el desarrollo de reconocimiento de productos para el Banco de Alimentos.

En el capítulo 5 se describe el proceso de investigación y desarrollo de detección de objetos diseñado por los tesistas utilizando modelos pre-entrenados. Además, se añade una breve explicación sobre las principales tecnologías utilizadas: el lenguaje de programación Python, la librería Tensor Flow y la interfaz Keras.

En el capítulo 6 se plantea la propuesta informática para el caso de estudio, para automatizar el proceso de ingreso de mercadería con el uso de inteligencia artificial y cuáles son las dificultades para llevarla a cabo.

Finalmente, en el capítulo 7 se presentan las conclusiones y líneas de trabajo futuro.

En el anexo se transcriben las entrevistas realizadas al director del Banco Alimentario, en estas se explica el funcionamiento de la Institución de forma detallada, comentando hasta cómo se resuelven los casos que suceden con menos frecuencia y que son muy específicos.

CAPÍTULO 2: TRANSFER LEARNING

En este capítulo se define el concepto de *Transfer Learning* y sus características. Además, se comparan las técnicas de *Transfer Learning* con la de *Machine Learning*, se explican las distintas estrategias y técnicas que pueden aplicarse en función del dominio, la tarea en cuestión y la disponibilidad de datos. Para finalizar, se definen los modelos pre-entrenados y las cuestiones a tener en cuenta al elegir un modelo pre-entrenado.

Los humanos tenemos una capacidad inherente para transferir conocimiento a través de tareas. Es decir, reconocemos y aplicamos el conocimiento relevante de las experiencias de aprendizaje anteriores cuando nos encontramos con nuevas tareas. Cuanto más relacionada esté una nueva tarea con nuestra experiencia previa, más fácilmente podremos dominarla.
Los algoritmos de *machine learning* y *deep learning* convencional, hasta ahora, se han diseñado tradicionalmente para trabajar de forma aislada, estos algoritmos están entrenados para resolver tareas específicas.
El *transfer learning* intenta cambiar esto, desarrollando métodos para transferir el conocimiento aprendido en una o más tareas previas y usarlo para mejorar el aprendizaje en una nueva tarea, esto se ejemplifica en la Figura 1. Las técnicas que permiten la transferencia de conocimiento representan el progreso haciendo que el *machine learning* sea tan eficiente como el aprendizaje humano.

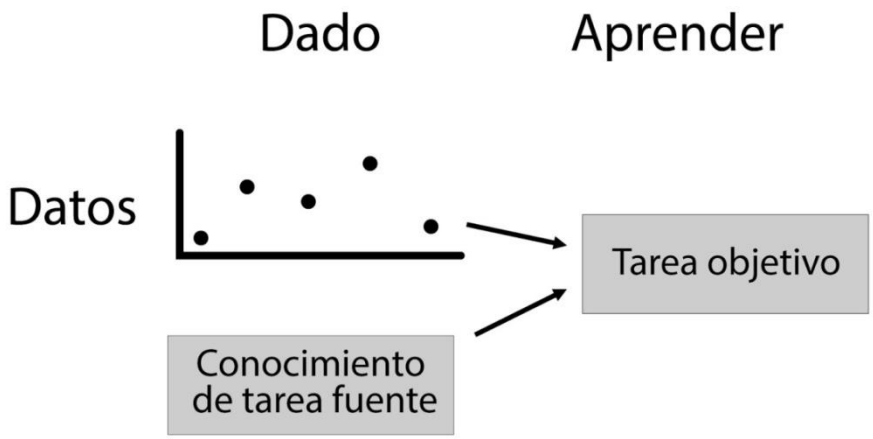

Figura 1. Transfer Learning
Adaptado de: Torrey L. y Shavlik J. (2009).

Los métodos de transferencia tienden a depender en gran medida de los algoritmos de *transfer learning* que se utilizan para aprender las tareas, y a menudo pueden considerarse simplemente extensiones de esos algoritmos. Parte del trabajo de transfer learning está en el contexto del aprendizaje inductivo e implica extender algoritmos de clasificación e inferencia bien conocidos, como redes neuronales, redes bayesianas y redes lógicas de Márkov. Otra área importante se

encuentra en el contexto del reinforcement learning e implica extender algoritmos como Q-learning y búsqueda de políticas.

El objetivo del *transfer learning* es mejorar el aprendizaje en la tarea objetivo, aprovechando el conocimiento de la tarea fuente. Existen tres medidas comunes por las cuales la transferencia podría mejorar el aprendizaje. Primero, el rendimiento inicial que se puede lograr en la tarea objetivo utilizando sólo el conocimiento transferido, antes de que se realice cualquier aprendizaje adicional, en comparación con el rendimiento inicial de un agente ignorante, el segundo es, la cantidad de tiempo que lleva aprender completamente la tarea objetivo dado el conocimiento transferido en comparación con la cantidad de tiempo para aprenderlo desde cero, y el tercero es el nivel de rendimiento final alcanzable en la tarea de destino en comparación con el nivel final sin transferencia, estas tres medidas se ilustran en la figura 2.

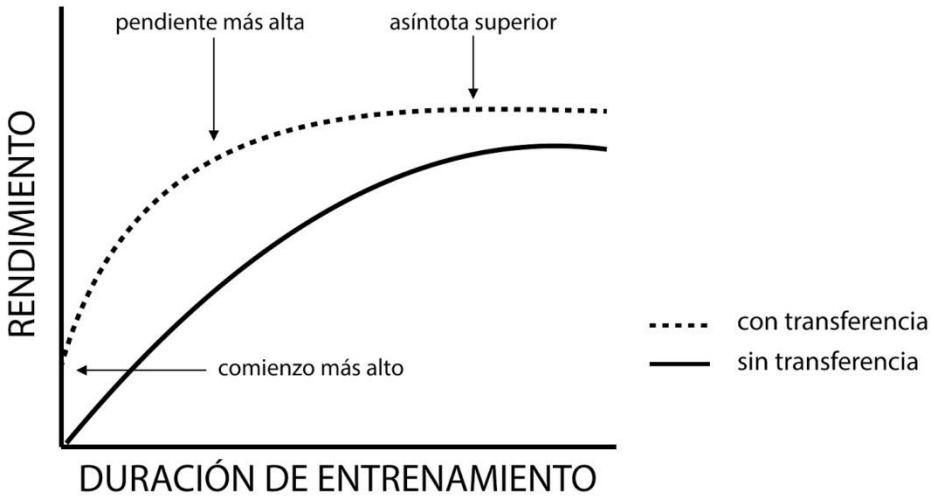

Figura 2. Tres formas de mejorar aprendizaje
Adaptado de: Torrey L. y Shavlik J.(2009)

El sitio oficial de Tensorflow define *transfer learning* como “Los modelos sofisticados de *deep learning* tienen millones de parámetros (pesos) y entrenarlos desde cero a menudo requiere grandes cantidades de datos de recursos informáticos. El *transfer learning* es una técnica que abrevia gran parte de esto al tomar una pieza de un modelo que ya ha sido capacitado en una tarea relacionada y reutilizarlo en un nuevo modelo[5].

A continuación definimos los principales conceptos involucrados en esta temática:

[5] Para ver más consutar: https://www.tensorflow.org/?hl=es-419.

Red neuronal:
Paradigma de aprendizaje y procesamiento automático inspirado en el funcionamiento del sistema nervioso humano. Una red neuronal está compuesta por un conjunto de neuronas interconectadas entre sí mediante enlaces. Cada neurona toma como entradas las salidas de las neuronas de las capas antecesoras, cada una de esas entradas se multiplica por un peso, se agregan los resultados parciales y mediante una función de activación se calcula la salida. Esta salida es a su vez es entrada de la neurona a la que precede.

Red Bayesiana:
Modelo probabilístico que relaciona un conjunto de variables aleatorias mediante un grafo dirigido, son redes gráficas sin ciclos en el que se representan variables aleatorias y las relaciones de probabilidad que existan entre ellas que permiten conseguir soluciones a problemas de decisión en casos de incertidumbre.

Red Lógica de Márkov:
La lógica de Márkov es un lenguaje para representar conocimiento que combina lógica y probabilidades, creando un sistema eficiente de inferencia y aprendizaje. La inferencia implica *grounding* (instanciación)y búsqueda. La instanciación transforma la representación lógica en cláusulas proposicionales con pesos que codifican una Red Márkov, formando la red lógica de Márkov (MLN) instanciada.

Reinforcement learning:
Es un área de aprendizaje automático relacionada con la forma en que los agentes de software deben tomar medidas en un entorno para maximizar la noción de recompensa acumulativa. El *reinforcement learning* es uno de los tres paradigmas básicos de aprendizaje automático, junto con el *supervised learning* y el *unsupervised learning*.

Q-learning:
Es una técnica de *reinforcement learning* utilizada en aprendizaje automático. El objetivo del Q-learning es aprender una serie de normas que le diga a un agente qué acción tomar bajo qué circunstancias. No requiere un modelo del entorno y puede manejar problemas con transiciones estocásticas y recompensas sin requerir adaptaciones.

Ya hemos discutido brevemente que los humanos no aprenden todo desde cero y aprovechan y transfieren su conocimiento de dominios previamente aprendidos a dominios y tareas nuevas. Dado el gran interés por alcanzar la verdadera inteligencia general artificial (AGI por sus siglas en inglés), el *transfer learning* es algo que los científicos e investigadores de datos creen que puede avanzar en nuestro progreso hacia la AGI.
El *transfer learning* no es un concepto muy reciente. En el taller de 1995 sobre los Sistemas de procesamiento de información neural (NIPS por sus siglas en inglés), el *workshop Learning to Learn: Knowledge Consolidation and Transfer in InductiveSystems*, proporcionó la motivación inicial para la investigación en este campo. Desde entonces, términos como *Learning to Learn, Knowledge Consolidation, e Inductive Transfer,* se han usado indistintamente con el transfer learning. Invariablemente, diferentes investigadores y textos académicos proporcionan distintas

definiciones como la que sostienen Goodfellow, Bengio y Courville (2016) al referirse al *transfer learning* en el contexto de la generalización. Su definición es la siguiente:

"*Situación en la que se aprovecha lo aprendido en un entorno para mejorar la generalización en otro entorno.*" (p.536)

2.1 Comparación con *Machine Learning*

El *transfer learning* no es un concepto nuevo y específico del *machine learning*. Existe una gran diferencia entre el enfoque tradicional de construcción y capacitación de modelos de *machine learning* y el uso de una metodología que sigue los principios de *transfer learning*, esto se puedo apreciar en la figura 3.

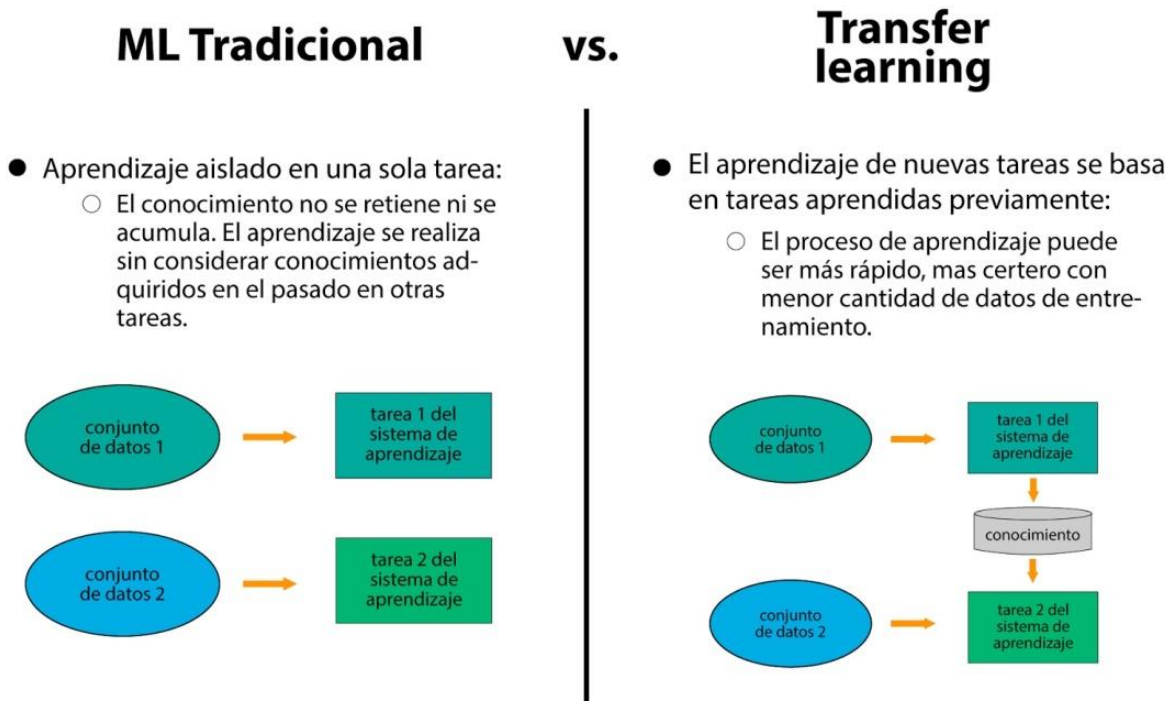

Figura 3. Traditional Machine Learning vs Transfer Learning
Adaptado de: Sarkar (2018)

El aprendizaje tradicional está aislado y ocurre únicamente en función de tareas específicas, conjuntos de datos y capacitación de modelos aislados separados entre ellos. No se retiene ningún conocimiento que pueda transferirse de un modelo a otro. Con *transfer learning*, se puede aprovechar el conocimiento (características, pesos, etc.) de modelos previamente entrenados para entrenar modelos más nuevos e incluso abordar problemas como tener menos datos para la tarea más nueva.

Por ejemplo, asumamos que nuestra tarea es identificar objetos en imágenes dentro de un dominio restringido de un restaurante. Llamemos a esta tarea T1. Dado el conjunto de datos para esta tarea, entrenamos un modelo y lo ajustamos para que funcione bien en puntos de datos invisibles del mismo dominio. Los algoritmos tradicionales de ML supervisados fallan cuando no tenemos suficientes ejemplos de capacitación para las tareas requeridas en dominios dados. Supongamos que ahora debemos detectar objetos de imágenes en un parque o un café (por ejemplo, tarea T2). Idealmente, deberíamos poder aplicar el modelo entrenado para T1, pero en realidad, enfrentamos una degradación del rendimiento y modelos que no se generalizan bien. Esto sucede por una variedad de razones, que podemos llamar colectivamente como el sesgo del modelo hacia los datos y el dominio del entrenamiento.

El *transfer learning* debería permitirnos utilizar el conocimiento de las tareas aprendidas previamente y aplicarlas a las más nuevas y relacionadas. Si tenemos significativamente más datos para la tarea T1, podemos utilizar su aprendizaje y generalizar este conocimiento (características, pesos) para la tarea T2 (que tiene significativamente menos datos). En el caso de problemas en el dominio de la visión por computadora, ciertas características de bajo nivel, como bordes, formas, esquinas e intensidad, se pueden compartir entre tareas y, por lo tanto, permiten la transferencia de conocimiento entre tareas. Además, como se representa en la Fig. 3, el conocimiento de una tarea existente actúa como una entrada adicional al aprender una nueva tarea objetivo.

2.2 Definición formal

En su trabajo los autores Pan y Yang (2010) utilizan el dominio, la tarea y las probabilidades marginales para presentar un marco para comprender el *transfer learning*. El marco se define de la siguiente manera:

Un dominio, **D**, se define como una tupla de dos elementos que consiste en espacio de características, **C**, y probabilidad marginal, **P(X)**, donde **X** es un punto de datos de muestra. Por lo tanto, podemos representar el dominio matemáticamente como **D = {C, P (X)}**

- Espacio de características: χ
- Distribución marginal: $P(X)$, $X = \{x_1, \ldots, x_n\}$, $x_i \epsilon \chi$

Aquí $\mathbf{x_i}$ representa un vector específico. Una tarea, **T**, por otro lado, puede definirse como una tupla de dos elementos del espacio de la etiqueta, **γ**, y la función objetivo, **η**. La función objetivo también se puede denotar como **P (γ | X)** desde un punto de vista probabilístico.

Para un dominio **D** dado, una tarea está definida por dos componentes:

$T = \{\gamma, P(Y|X)\} = \{\gamma, \eta\}$ $Y = \{y_1, \ldots, y_n\}$, $y_i \epsilon \gamma$

- Un espacio de etiqueta: **γ**
- Una función predictiva **η**, aprendida de vectores de características o pares de etiquetas.
- para cada vector de características en el dominio, **η** predice su etiqueta correspondiente: $\eta(x_i) = y_i$

Por lo tanto, armados con estas definiciones y representaciones, podemos definir el *transfer learning* de la siguiente manera:

Dado un dominio de origen DO, una tarea de origen correspondiente TO, así como un dominio de destino DD y una tarea de destino TD, el objetivo de *transfer learning* ahora es permitirnos aprender la distribución de probabilidad condicional objetivo P (YT | XT) en DD con la información obtenida de DO y TO donde DO ≠ DD o TO ≠ DO.
En la mayoría de los casos, se supone que hay disponible un número limitado de ejemplos de destino etiquetados, que es exponencialmente menor que el número de ejemplos de origen etiquetados.

2.3 Estrategias de transfer learning

Existen diferentes estrategias y técnicas de *transfer learning* que pueden aplicarse en función del dominio, la tarea en cuestión y la disponibilidad de datos.

Como se puede apreciar en la Figura 4, los métodos de *transfer learning* se pueden clasificar según el tipo de algoritmos tradicionales de *machine learning* involucrados, como:

- ***Inductive Transfer learning***: en este escenario, los dominios de origen y destino son los mismos, pero las tareas de origen y destino son diferentes entre sí. Los algoritmos intentan utilizar los sesgos inductivos del dominio de origen para ayudar a mejorar la tarea de destino. Dependiendo de si el dominio de origen contiene datos etiquetados o no, esto se puede dividir en dos subcategorías, similar al aprendizaje multitarea y el aprendizaje autodidacta, respectivamente.
- ***Unsupervised Transfer Learning***: esta configuración es similar a la transferencia inductiva en sí misma, con un enfoque en tareas no supervisadas en el dominio de destino. Los dominios de origen y destino son similares, pero las tareas son diferentes. En este escenario, los datos etiquetados no están disponibles en ninguno de los dominios.
- ***Transductive Transfer Learning***: en este escenario, hay similitudes entre las tareas de origen y de destino, pero los dominios correspondientes son diferentes. En esta configuración, el dominio de origen tiene muchos datos etiquetados, mientras que el dominio de destino no tiene ninguno. Esto puede clasificarse aún más en subcategorías, en referencia a configuraciones donde los espacios de características son diferentes o las probabilidades marginales.

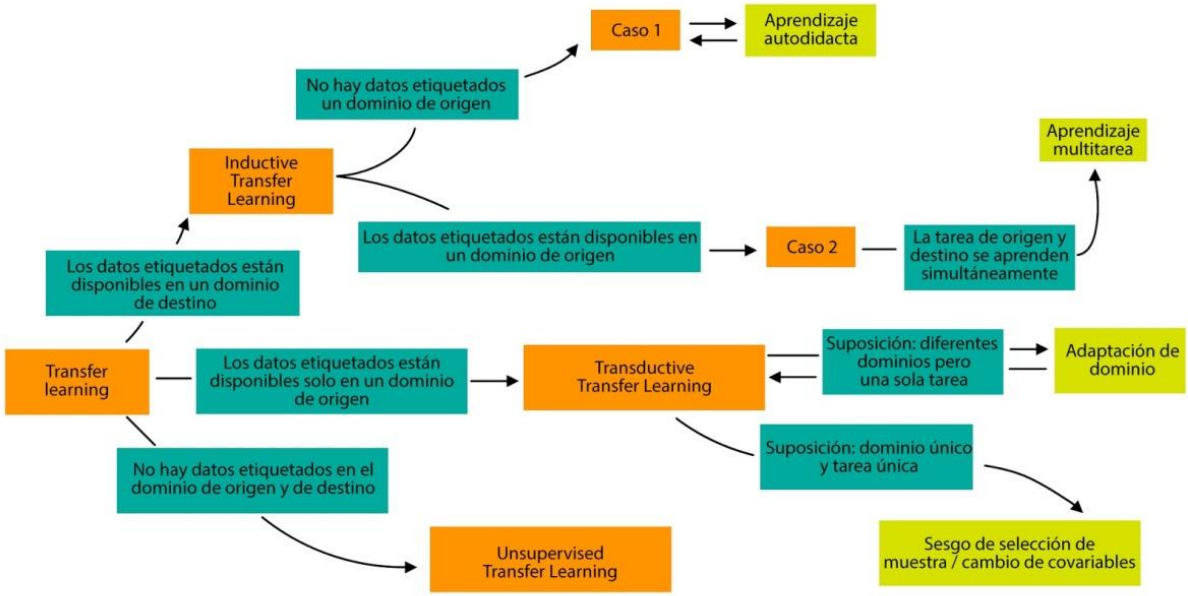

Figura 4. Estregias de transfer learning
Adaptado de Pan y Yang Fellow (2010)

Las tres categorías de transferencia discutidas en la sección anterior describen diferentes configuraciones donde el transfer learning se puede aplicar y estudiar en detalle.

A continuación, se describe qué se debe transferir a través de las categorías vistas:

- ***Instance transfer***: la reutilización del conocimiento del dominio de origen a la tarea de destino suele ser un escenario ideal. En la mayoría de los casos, los datos del dominio de origen no se pueden reutilizar directamente. Más bien, hay ciertas instancias del dominio de origen que se pueden reutilizar junto con los datos de destino para mejorar los resultados.
- ***Feature-representation transfer***: este enfoque tiene como objetivo minimizar la divergencia de dominios y reducir las tasas de error mediante la identificación de buenas representaciones de características que pueden utilizarse desde los dominios de origen a destino. Dependiendo de la disponibilidad de datos etiquetados, se pueden aplicar métodos supervisados o no supervisados para transferencias basadas en la representación de características.
- ***Parameter transfer****:* este enfoque funciona bajo el supuesto de que los modelos para tareas relacionadas comparten algunos parámetros o distribución previa de hiperparámetros. A diferencia del aprendizaje multitarea, donde tanto las tareas de origen como las de destino se aprenden simultáneamente, para el aprendizaje de transferencia, podemos aplicar un peso adicional a la pérdida del dominio de destino para mejorar el rendimiento general.
  Relational-knowledge transfer: a diferencia de los tres enfoques anteriores, la transferencia de conocimiento relacional intenta manejar datos que no son independientes e idénticamente distribuidos, como los datos que no son independientes e idénticamente distribuidos. En otras

palabras, datos, donde cada punto de datos tiene una relación con otros puntos de datos; por ejemplo, los datos de redes sociales utilizan técnicas de transferencia de conocimiento relacional.

2.4 Modelos Pre-entrenados

En pocas palabras, un modelo pre-entrenado es un modelo creado por alguien más y entrenado con abundantes datos para resolver un problema similar. En lugar de construir un modelo desde cero para resolver un problema similar, utiliza el modelo entrenado en otro problema como punto de partida.
Por ejemplo, si se desea construir un sistema de reconocimiento de plantas. Puede dedicar meses a construir un algoritmo de reconocimiento de imagen aceptable desde cero o puede tomar un modelo de inicio (un modelo pre-entrenado) de Google que se construyó sobre datos de ImageNet para identificar imágenes en esas fotos.
Un modelo pre-entrenado puede no ser 100% preciso en su aplicación, pero ahorra enormes esfuerzos necesarios para reinventar los mismos conceptos una y otra vez.

El objetivo de entrenar una red neuronal es identificar los pesos correctos para la red mediante múltiples iteraciones hacia adelante y hacia atrás. Al usar modelos pre-entrenados que han sido entrenados previamente en grandes conjuntos de datos, podemos usar directamente los pesos y la arquitectura obtenidos y aplicar el aprendizaje en nuestra declaración del problema. Esto se conoce en inglés como *Transfer Learning.* Transferimos el aprendizaje del modelo pre-entrenado a nuestro enunciado del problema específico.

Se debe tener mucho cuidado al elegir qué modelo pre-entrenado se debe usar en cada caso. Si el problema es muy diferente del que se formó el modelo pre-entrenado, la predicción que se obtiene será muy inexacta. Por ejemplo, un modelo previamente entrenado para el reconocimiento de voz funcionará muy mal si se trata de usar para identificar objetos.

Por suerte existen muchas arquitecturas pre-entrenadas que están directamente disponibles para su uso en la biblioteca de Keras[6]. El conjunto de datos de ImageNet se ha utilizado ampliamente para construir varias arquitecturas, ya que es lo suficientemente grande (1.2M de imágenes) para crear un modelo generalizado. La especificación del problema es entrenar un modelo que pueda clasificar correctamente las imágenes en 1000 categorías de objetos separadas. Estas 1000 categorías de imágenes representan clases de objetos que se encuentran en la vida cotidiana, como especies de perros, gatos, diversos objetos domésticos, tipos de vehículos, etc.

Estas redes pre-entrenadas demuestran una gran habilidad para generalizar a imágenes fuera del conjunto de datos ImageNet a través del *transfer learning*. Es posible realizar modificaciones

[6]Keras es una API de deep learning escrita en Python, que se ejecuta sobre la plataforma de machine learning TensorFlow. Fue desarrollado con un enfoque en permitir una experimentación rápida. (Recuperado de: https://keras.io/about/)

en el modelo preexistente ajustando el modelo. Dado que se supone que la red pre-entrenada ha sido entrenada bastante bien, no es deseable modificar los pesos tan pronto y demasiado. Para modificarlos, generalmente se usa una tasa de aprendizaje más pequeña que la utilizada para entrenar inicialmente al modelo.

2.5 Conclusión

Durante este capítulo hemos indagado en la teoría de *Transfer Learning* y modelos pre-entrenados. Hemos descrito qué es *Transfer learning*, comparado con *Machine Learning* y hemos definido qué son los modelos pre-entrenados.

Llegamos a la conclusión de que *Transfer Learning* es una técnica muy útil que nos será de ayuda junto a los modelos pre-entrenados de Keras, ya que no contamos con un conjunto de imágenes tan grande ni con el poder de cómputo necesario para entrenar nuestro modelo desde cero y además tampoco resulta conveniente re-inventar conceptos ya elaborados previamente, sino que se recomienda reutilizar para propósitos particulares.

En los próximos capítulos se explicará cómo aplicar *Transfer Learning* con modelos pre-entrenados y que estrategia seguir dependiendo de cada caso.

CAPÍTULO 3: EL BANCO ALIMENTARIO

En este capítulo se describe el caso de estudio que motiva la presente tesina. El cual se presenta de acuerdo con la información extraída de las entrevistas con el director del Banco Alimentario de La Plata, el Sr. Pedro Elizalde (Anexo 1 y Anexo 2). En estas, Elizalde detalla qué es el Banco Alimentario, qué funciones cumple, cómo trabajan, qué problemas tienen y cómo una solución basada en *Machine Learning* para el registro de los ingresos de productos al Banco los beneficiaría y las dificultades a tener en cuenta.

El Banco Alimentario de La Plata es una sociedad civil sin fines de lucro, reconocida como persona jurídica. El Banco fue fundado en el año 2000, es decir que hasta el momento tiene 20 años de existencia.

Emplea un modelo que se basa en experiencias internacionales, el cual pretende poner en valor alimentos. Se fundamenta en recuperar alimentos y ponerlos a disposición de los sectores vulnerables. Los alimentos recuperados son alimentos que se encuentran en perfectas condiciones con respecto a sus valores nutricionales y que, por diversas situaciones, como puede ser comerciales o de producción, iban a ser descartados.

La alimentación es un sector muy delicado, por lo que el Banco Alimentario intenta tener protocolos y normas que permitan ser muy estrictos en todos los procesos que realizan. Inclusive cuenta con auditorías con respecto a los aspectos de gestión del Banco.

En este momento en Argentina existen 14 bancos de alimentos operativos. Estos integran una red con fines de coordinar para mejorar los desempeños y conseguir una mayor cantidad de donantes.
A su vez, el Banco Alimentario de La Plata integra una red global de bancos de alimentos, la *Global Food Banking*.
Esto quiere decir que el Banco Alimentario de La Plata está integrado y protocolizado de acuerdo con normas de buenas prácticas de manufactura y que tienen que ver con lo legal en los alimentos.

De forma simplificada, la función del Banco Alimentario es conseguir donaciones, ingresarlas al depósito para clasificarlas, y realizar la logística para enlazar la mercadería a los comedores, según con qué productos se cuenta y las necesidades que poseen las instituciones.
El Banco cuenta con una base de datos de instituciones que brindan servicios alimentarios. Este registro permite saber la composición de los servicios que brindan y a qué población atiende, lo que les posibilita realizar la logística para repartir la mercadería de forma equitativa y siempre que se pueda, saciar las necesidades de cada comedor.

El Banco Alimentario de La Plata cuenta con un personal estable que está integrado por un director ejecutivo, una persona encargada del área de administración, una persona encargada del área de logística, una persona encargada del área social, una persona encargada del área de comunicación, un jefe de depósito, un chofer y un ayudante encargados de uno de los vehículos del Banco.
Aparte del personal estable, el Banco Alimentario cuenta con personal voluntario que le da al modelo mucha potencia.

3.1 Sistema Actual: Kolsen

Actualmente el Banco recibe los productos, previamente acordada la fecha, los ingresan al edificio y son registrados en una planilla. Por cada tipo de producto que reciben, registran sus datos, como lo son el tipo de producto, el proveedor, la fecha de ingreso, la fecha de vencimiento, el peso y la cantidad de unidades.
Luego de llenar la planilla, los datos son pasados al sistema informático, llamado Kolsen, desarrollado para los bancos de alimentos del país que forman parte de la red.
En el sistema Kolsen también tienen registrados los comedores, la cantidad de personas que asisten y sus necesidades. Con estos datos y con la información de los alimentos almacenados, los operarios generan remitos con los productos que se van a entregar a cada comedor. Una vez listo el remito pueden empezar a armar el pedido físico y luego son retirados por cada uno de los comedores.
Además, el sistema también es utilizado para realizar métricas y llevar a cabo auditorías.

3.2 Procesamiento de mercadería

En este punto se explicará más detalladamente el proceso del ingreso de la mercadería al Banco Alimentario.
Al Banco Alimentario llegan los productos desde un grupo variado de donantes, que pueden ser desde industrias de alimentos, supermercados, productores agropecuarios, etc.
A los donantes se les entrega un recibo especificando la donación.
Las donaciones llegan de variadas maneras, como puede ser *pallets* o bolsones. Además, los productos pueden venir envasados, en bolsones o sueltos como en el caso de las frutas y verduras. Por este motivo, el Banco Alimentario registra los productos de distintas maneras, algunos recolectando los datos del envase, en otros casos además debe pesarlo, en pocos casos leyendo el código de barra con una pistola. Es decir, no se tiene una manera unificada de tomar los datos de las donaciones.
Para registrar el ingreso de un producto es necesario indicar el proveedor, la marca, una descripción, el tipo, la cantidad, el peso por unidad, el peso en total, la fecha de ingreso y la fecha de vencimiento.

Una vez tomados los datos de los productos, se le entrega al donante un comprobante con los detalles de la donación.
Ya con la donación ingresada al sistema, se puede enlazar los productos con cada comedor, teniendo en cuenta como prioridad los alimentos con corta fecha de vencimiento y siendo equitativos con los comedores, teniendo en cuenta la cantidad de personas que asisten y las necesidades, ya que brindan distintos servicios.
Para armar una entrega, en el sistema se genera un remito indicado el comedor beneficiario y los productos con sus pesos y cantidades.
Finalizado el remito, se arma la entrega físicamente y por último se avisa al comedor para que pasen a retirarla.

3.3 Problemáticas

Si bien la manera de trabajar de los operarios está probada y mecanizada, hay puntos en los que se podría mejorar y agilizar el proceso.
Por un lado, los productos de las donaciones son registrados primeramente en una planilla a mano y luego son cargados al sistema, lo que hace que el registro se haga dos veces.
Por otro lado, la carga de las donaciones en el sistema es manual, por lo que está sujeta a fallos del operador.

Además, Kolsen es un sistema informático de aproximadamente 20 años y que no recibe actualizaciones, tanto para resolver inconsistencias en la versión actual como para permitir integraciones con otros sistemas, generando limitaciones de escalabilidad.

3.4 Solución propuesta

El objetivo general de esta tesina es el análisis de la técnica de Transfer Learning en Aprendizaje Automático. Para ello se pondrán en práctica los conceptos teóricos estudiados a través de este caso de estudio. Se construirá un prototipo para de detección y clasificación de productos alimenticios a partir de imágenes, detallando el proceso de desarrollo seguido.
Además de constituir una prueba de concepto, este prototipo intentará brindar una herramienta que facilite el accionar cotidiano del Banco Alimenticio de La Plata, contribuyendo a la meritoria tarea que éste lleva adelante.

CAPÍTULO 4: MODELOS DE DETECCIÓN DE OBJETOS

4.1 Introducción

En esta tesina se ha estado hablando de TL y de la capacidad para reutilizar modelos pre entrenados en las aplicaciones desarrolladas.
Es por ello que en este capítulo se enfocará en compararlos de manera tal de poder tener una perspectiva más clara sobre qué modelo es mejor para el fin propuesto en la investigación.
Para ello, se definirá entorno de pruebas, también se elegirá el hardware a utilizar, por último se diseñarán las pruebas en sí, así como los parámetros que se tomarán en cuenta para la comparación.

Se utiliza *object detection Tensorflow*, un *framework* que aporta herramientas que serán de ayuda para el entrenamiento.
Durante este capítulo se irán desarrollando los conceptos de las herramientas citadas desde diferentes perspectivas.
Inicialmente se enfocará el análisis a los modelos, ya que con el avance tan brusco de la tecnología cualquier comparación queda obsoleta rápidamente.
Además, sabiendo que existen decenas de modelos disponibles, los tesistas han decidido encarar una investigación sobre la elección del mejor modelo apuntando a la solución de detección de objetos para el Banco Alimentario.

4.1.1 Conceptos básicos que afectan el entrenamiento de una red neuronal

Es muy difícil hacer una comparación justa entre los diferentes detectores de objetos. No hay una respuesta directa sobre qué modelo es el mejor. Antes de realizar la selección indicada, se debe conocer qué factores afectan al rendimiento y para ello es preciso examinar algunas de las opciones que afectan el rendimiento al entrenar.

- El Proceso de resolución de cada imagen: cada imagen es una matriz de 3 dimensiones que debe ser recorrida en la *convolución,* a través del kernel, dependiendo del tamaño de la imagen y la configuración de la convolucional. Este proceso puede variar el tiempo de procesamiento.
- El Tiempo de entrenamiento Insumido en la Puesta en Marcha: el tiempo de entrenamiento sobre la capacidad de cómputo es un factor clave para obtener el rendimiento deseado de la red considerada.
- Tipo de software utilizado: La selección del software depende la tecnología utilizada, así como de la aptitud de este de aprovechar las bondades del hardware disponible para entrenar.

- Número de clases definidas: el número de clases simboliza cada objeto individual que la red podrá reconocer. Se definen la cantidad de imágenes por clase que tendrá la red, (en el caso de esta investigación fueron 30 en promedio).
  Se, tienen *N* clases, y *X es* la cantidad de imágenes por cada clase del dataset. El dataset crecerá en *N*X,* aumentando considerablemente la red.
- Tamaño del dataset: relacionado con lo que se menciona en el punto anterior. Si se construye un dataset de mayor tamaño, ello exigirá un más prolongado tiempo de entrenamiento.
- El uso de multi escalas en las imágenes utilizando ImageAugmentation: Muchas veces es difícil obtener buena calidad de datos para cada clase, y de esta manera mantener balanceado el dataset en cuanto a distribución de calidad por clase. Existen herramientas para rotar e invertir cada imagen del dataset, así, de esta manera se logra aumentar la calidad del aprendizaje para cada clase, todo ello pagando el precio de un aumento considerable del dataset.
- Configuraciones en el entrenamiento o también denominados hiper parámetros: son parámetros de configuración que afectan al aprendizaje tales como batchsize, input imageresize y learning rate[7].
- Calidad del labeling: para cada imagen del dataset se debe identificar manualmente la ubicación del objeto en la imagen. Así, la red neuronal obtiene las características de éste ignorando lo demás. Se ha usado ImgLabel y este proceso será descripto en profundidad en las siguientes secciones.

4.2 Red neuroral convolucional

Las redes neuronales convolucionales (CNN) se las puede encontrar en todas partes. Podría decirse que es la arquitectura de aprendizaje profundo más popular en los usuarios de redes neuronales convolucionales.

El reciente aumento de interés en el aprendizaje profundo se debe a la inmensa popularidad y eficacia de las redes neuronales convolucionales (convnets), y así es que la atracción que han tenido los usuarios en CNN comenzó con AlexNet en 2012 (Krizhevsky, Sutskever y Hinton, 2012). y ha crecido exponencialmente desde entonces.

En solo tres años, los investigadores pasaron de AlexNet de 8 capas a ResNetde 152 capas (He, Zhang, Ren y Sun, 2015).

Actualmente, CNN es el modelo de referencia utilizado en todos los problemas relacionados con las imágenes. Si se analiza la precisión de la herramienta, supera con creces a la competencia que queda muy atrás en la inclinación de los usuarios.

También se aplica con éxito a sistemas de recomendación, procesamiento del lenguaje natural y más. La principal ventaja de CNN en comparación con sus predecesores es que detecta automáticamente las funciones importantes sin supervisión humana. Por ejemplo, dadas muchas imágenes de perros y gatos, aprende las características distintivas de cada clase por sí mismo.

[7] Learning rate: es un parámetro de ajuste en un algoritmo de optimización que determina el tamaño del paso en cada iteración mientras se mueve hacia un mínimo de una función de pérdida

CNN también es computacionalmente eficiente. Utiliza operaciones especiales de convolución y agrupación y realiza el intercambio de parámetros. Esto permite que los modelos de CNN se ejecuten en cualquier dispositivo, lo que los hace universalmente atractivos.

Todas estas bondades de la herramienta permiten sospechar que se está frente a una tecnología superior a las disponibles en el mercado, y es por eso que se tiene disponible un modelo muy poderoso y eficiente que realiza la extracción automática de características para lograr una precisión sobrehumana[8].

Los tesistas esperan descubrir las potencialidades de esta notable técnica en los trabajos encarados en esta investigación.

4.2.1 Arquitectura básica

Todos los modelos de CNN siguen una arquitectura similar, como se muestra en la siguiente figura 5.

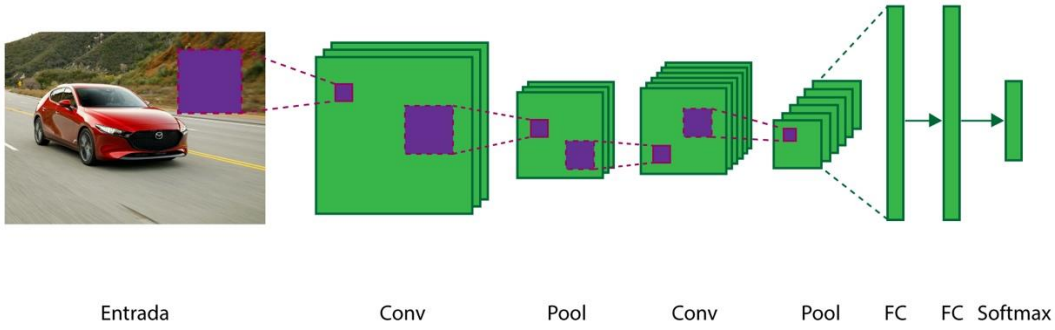

Figura 5. Estructura Básica de una red convolucional

Elaboración propia (2020)

Hay una imagen de entrada con la que se está trabajando. Se realizan una serie de operaciones de convolución y agrupaciónes *max pooling*, seguidas de una serie de capas completamente

[8]Actualmente, los modelos de CNN clasifican las imágenes mejor que los humanos

conectadas. Si se está re1alizando una clasificación multi clase, la salida es *softmax*. El análisis en detalle de cada componente será explicado a continuación.

4.2.1.1 Convolución

El bloque de construcción principal de CNN es la capa convolucional. La convolución es una operación matemática para fusionar dos conjuntos de información. En el caso de esta investigación, la convolución se aplica a los datos de entrada utilizando un filtro de convolución llamado kernel para producir un mapa de características.

Debido a que son utilizados muchos términos, es necesario visualizarlos uno por uno. Como ilustra la figura 11 la capa azul es el input, el recuadro sombrado 3x3 es el kernel y la capa verde es el output resultante luego de aplicar el filtro 3x3.

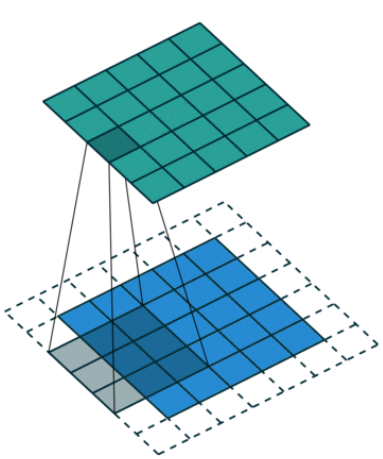

Figura 6. Ilustración de una convolucion denotando la entrada y si salida aplicando un filtro (kernel)
Adaptado de: Kunlun bai (2019)

Kernel: el tamaño del núcleo define el campo de visión de la convolución. Una opción común para 2D es 3, es decir, 3x3 píxeles.

Stride: el paso define el tamaño del paso del kernel al atravesar la imagen. Si bien su valor predeterminado generalmente es 1, se puede usar un paso de 2 para reducir la resolución de una imagen similar a MaxPooling.

Padding: el relleno define cómo se maneja el borde de una muestra. Una convolución (medio) rellenada mantendrá las dimensiones de salida espacial iguales a la entrada, mientras que las convoluciones sin relleno recortan algunos de los bordes si el núcleo es mayor que 1.
Canales de entrada y salida: una capa convolucional toma un cierto número de canales de entrada (I) y calcula un número específico de canales de salida (O). Los parámetros necesarios

para dicha capa se pueden calcular mediante I * O * K, donde K es igual al número de valores en el núcleo.

4.2.1.1.1 Depth-wise Convolution

En esta convolución, se aplica un filtro de profundidad 2-d en cada nivel de profundidad del tensor de entrada. Este concepto será mejor entendido mediante un ejemplo. Si se supone que el tensor de entrada utilizado en esta investigación es 3 * 8 * 8 (input_channels * width * height), el filtro será de 3 * 3 * 3. En una convolución estándar, también se haría la convolución directamente en la dimensión de profundidad y esto se aprecia en la figura 7.

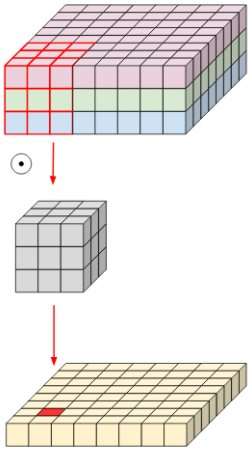

Figura 7. Normal Convolution
Adatado de: Atul Pandey (2018).

En convolución en profundidad, se usa cada canal de filtro solo en un canal de entrada. En el ejemplo, se tiene un filtro de 3 canales y una imagen de 3 canales. Lo que se hace es dividir el filtro y la imagen en tres canales diferentes. Luego se vincula la imagen correspondiente con el canal correspondiente y posteriormente se vuelve a apilarlos (Figura 8)

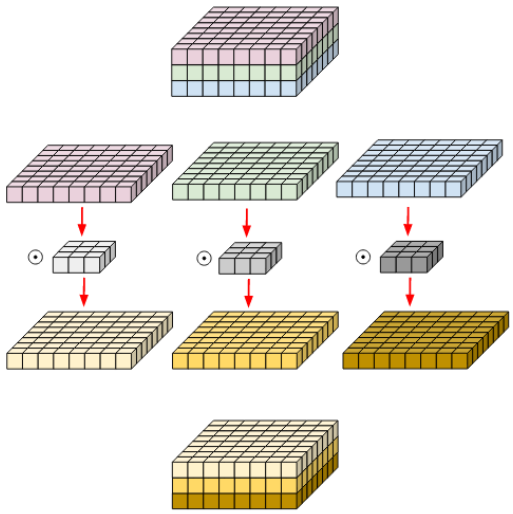

Figura 8. Depth wise convolution
Adatado de: Atul Pandey (2018)

Para producir el mismo efecto con convolución normal, lo que se hace es seleccionar un canal, hacer que todos los elementos sean cero en el filtro excepto ese canal, luego ejecutar la convolución. Se Necesitarán tres filtros diferentes, uno para cada canal. Aunque los parámetros siguen siendo los mismos, esta convolución le brinda tres canales de salida con solo un filtro de 3 canales, mientras que necesitaría tres filtros de 3 canales si utilizara la convolución normal.

Conocido esto, se puede explicar la principal diferencia entre Inception y mobilenet:

Mobilenet usa Depth Wise separable convolution mientras que Inception usa convolución estándar. Esto da como resultado un menor número de parámetros en MobileNet en comparación con Inception. Sin embargo, esto también produce una ligera disminución en el rendimiento.
En una convolución estándar, el filtro funciona en todos los canales M de la imagen de entrada y genera N mapas de características, es decir, la multiplicación matricial entre la entrada y el filtro es multidimensional.

Para que el lector entienda mejor estos conceptos, si asemeja al filtro a un cubo de tamaño Dk x Dk x M, luego, en convolución estándar, cada elemento del cubo se multiplicará con el elemento correspondiente en la matriz de características de entrada y, finalmente, después de la multiplicación, los mapas de características serán agregados a Salida N mapas de características.

Sin embargo, en una Depth Wise separable convolution, los filtros de canal único M funcionarán en un solo cubo en la función de entrada y una vez que se obtengan las salidas de filtro M, un

filtro puntiagudo de tamaño 1 x 1 x M funcionará en él para dar N mapas de características de salida.

4.2.1.1.2 Depth-wise Separable Convolution

Esta convolución se originó a partir de la idea de que la *Depth-wise Convolution* se puede separar, de allí el adjetivo “separable”. Un ejemplo de ello es el del filtro Sobel, utilizado en el procesamiento de imágenes para detectar bordes. Puede separar la dimensión de alto y ancho de estos filtros. En el filtro Gx (véase la figura 9) se puede ver como producto de matriz de [1 2 1] se transponer con [-1 0 1].

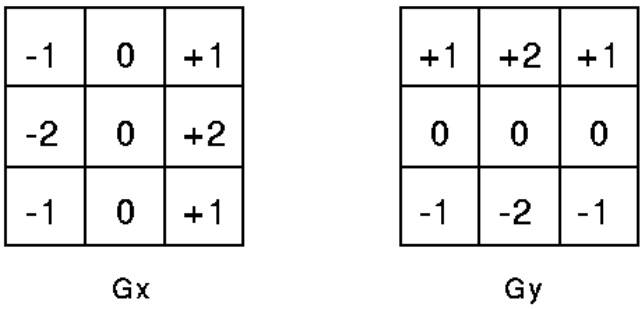

Figura 9. Filtro Sobel

Nota. Gx para borde vertical, Gy para detección de borde horizontal. Adatado de: Atul Pandey (2018).

Aquí se puede observar que el filtro se había disfrazado. Muestra que tenía 9 parámetros pero en realidad tiene 6. Esto ha sido posible debido a la separación de sus dimensiones de altura y ancho. La misma idea aplicada para separar la dimensión de profundidad de la horizontal (ancho * alto) permite obtener una convolución separable en profundidad, en este caso se hace una convolución en profundidad y luego se usa un filtro 1 * 1 para cubrir la dimensión de profundidad (figura 10).

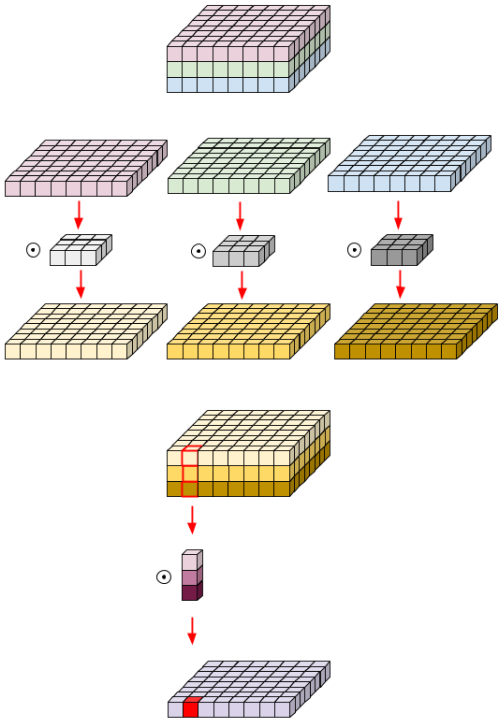

Figura 10. Depth wise separable convolution
Adatado de: Atul Pandey (2018).

Es importante observar la cantidad de parámetros que son reducidos por esta convolución, para generar el mismo número de canales.

Para producir uno de los canales se necesitan 3 * 3 * 3 parámetros para realizar una convolución en profundidad y 1 * 3 parámetros para realizar una mayor convolución en la dimensión de profundidad.

Además, si se necesitan 3 canales de salida, solo son necesarios 3 filtros de profundidad 1 * 3 que nos dan un total de 36 (= 27 +9) parámetros.

Mientras que, para la misma cantidad de canales de salida en convolución normal, son necesarios 3 * 3 * 3 * 3 filtros que dan un total de 81 parámetros.

Al tener demasiados parámetros, el modelo tiene inclinación a memorizar en vez de aprender y, por lo tanto, al ajustarse demasiado. La convolución separable en profundidad soluciona este problema.

4.2.1.2 Agrupación de capas (max pooling)

Después de una operación de convolución, es común realizar una agrupación de capas (conocido en inglés como *max pooling*) para reducir el peso. Esto nos permite reducir el número de parámetros, lo que acorta el tiempo de entrenamiento y reduce el sobreajuste.

La agrupación de capas reduce la resolución de cada mapa de características de forma independiente, reduciendo la altura y el ancho, manteniendo la profundidad intacta.

El tipo más común de *pooling* es la *max pooling*, que solo toma el valor máximo en la ventana de agrupación. Al contrario de la operación de convolución cuando se usa el kernel como filtro, la agrupación no tiene parámetros.

Desliza una ventana sobre su entrada y simplemente toma el valor máximo en la ventana. De manera similar a una convolución, se especifica el tamaño de la ventana y el paso.

La Figura 11 refleja el resultado de la operación max pooling usando una ventana de 2x2 y un paso 2. Cada color denota una ventana diferente. Dado que tanto el tamaño de la ventana como el paso son 2, las ventanas no se superponen.

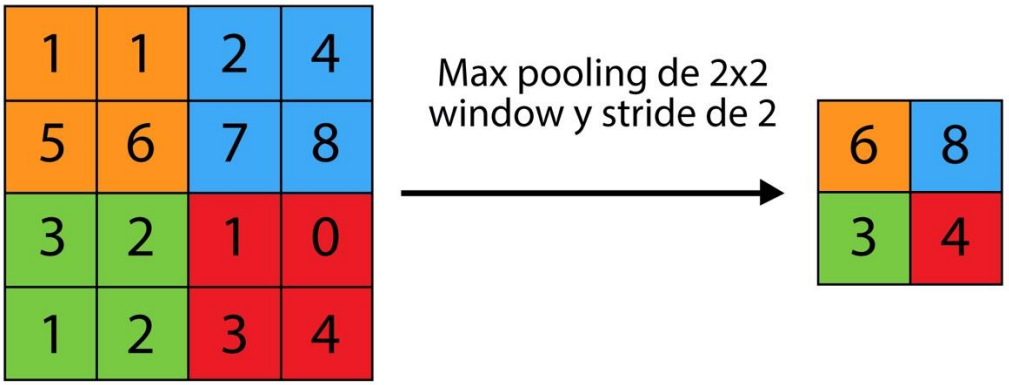

Figura 11. Max pooling reflejando parámetros de entrada y salida. Elaboración propia (2020)

4.2.1.3 Fully connected

Después de las capas de convolución y agrupación, se utiliza un par de capas completamente conectadas para resumir la arquitectura de CNN. La salida de las capas de convolución y agrupación son volúmenes tres dimensiones, pero una capa completamente conectada (*Fully connected* en inglés) espera un vector de números en una dimensión.

De esta manera se aplana la salida de la capa de agrupación final a un vector y eso se convierte en la entrada de la capa completamente conectada.

El aplanamiento es una operación para simplemente organizar el volumen 3 dimensiones de números en un vector en una dimensión.

4.2.1.4 Softmax

Softmax lleva esta idea al plano de las clases múltiples. Es decir, *softmax* asigna probabilidades decimales a cada clase en un caso de clases múltiples. Esas probabilidades decimales deben sumar 1.0. Esta restricción adicional permite que el entrenamiento converja más rápido. Por ejemplo, si se vuelve a analizar la imagen de la Figura 1, *softmax* puede producir las siguientes probabilidades de una imagen que pertenezca a una clase particular: *Softmax* se implementa a través de una capa de red neuronal justo antes de la capa de resultado. La capa de *softmax* debe tener la misma cantidad de nodos que la capa de resultado[9].

4.3 Descripción de los modelos usados en esta Tesina

Existen decenas de modelos s disponibles y los tesistas han seleccionado tres modelos como se muestra en la tabla 1 para esta investigación, uno de tipo Faster Region-based Convolutional Neural Network abreviado como: Fast R-CNN y, dos de Single Shot Multibox Detection (SSD), entre estos tres existen variantes acopladas a la arquitectura que se son mobileNet y Incepcion.

En esta investigación se usarán modelos para detectar productos en pruebas controladas. Para ello se desarrollará una comparación válida y justa entre los dos modelos SSD MobileNet y Faster-RCNN.
Los modelos serán entrenados con los datos de entrada (dataset), el concepto de dataset será desarrollado más adelante.
Se aplicará TL utilizando los modelos previamente entrenados proporcionados por el *TensorFlow Model Zoo*. Previamente, estos modelos fueron entrenados exhaustivamente utilizando *common object in context* (COCO) conjunto de datos.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
faster_rcnn_inception_coco	58	28	Boxes
ssd_mobilenet_coco	31	22	Boxes
ssd_inception_coco	42	24	Boxes

[9] Redes Neuronales de clases múltiples: sofmax. *Aprendizaje automático*. Recuperado de: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax?hl=es#:~:text=Softmax%20lleva%20esta%20idea%20al,probabilidades%20decimales%20deben%20sumar%201.0.

Tabla 1. Comparación de Modelos Utilizados
Adaptado de: repositorio Github

El *TensorFlow Model Zoo* proporciona una colección de modelos de detección previamente entrenados utilizando conjunto de datos open source, como el conjunto de datos common object in context (COCO) dataset (Lin, *et al*., 2015).
Estos modelos pueden ser útiles paraout-of-the-box inference, si la investigación está interesada en categorías que ya están en esos conjuntos de datos. También son útiles para inicializar sus modelos cuando entrena en nuevos conjuntos de datos.

COCO dataset: Contiene 91 categorías de objetos comunes, de las cuales 82 tienen más de 5,000 instancias etiquetadas. En total, el conjunto de datos tiene 2,500,000 instancias etiquetadas en 328,000 imágenes. A diferencia del popular conjunto de datos ImageNet, COCO tiene menos categorías, pero más instancias por categoría. Esto puede ayudar a aprender modelos de objetos detallados capaces de una localización 2D precisa.

Además, una distinción crítica entre un conjunto de datos y otros es el número de instancias etiquetadas por imagen que puede ayudar a aprender información contextual.
La detección de un objeto implica dos cosas, primero indicar que el objeto que pertenece a una clase específica está presente y segundo ubicarlo en la imagen. La ubicación de un objeto normalmente está representada por un cuadro delimitador (este proceso será descripto en las siguientes secciones).

4.3.1 Faster Region-based Convolutional Neural Network (Fast R-CNN)

Una red Fast R-CNN representada en la figura 12 toma como entrada una imagen completa y un conjunto de propuestas de objetos. La red primero procesa la imagen completa con varias capas convolucionales y *maxpooling*, como se ha visto anteriormente, para producir un mapa de características. Luego, para cada propuesta de objeto, una capa de agrupación de región de interés (RoI) extrae un vector de características de longitud fija del mapa de características.

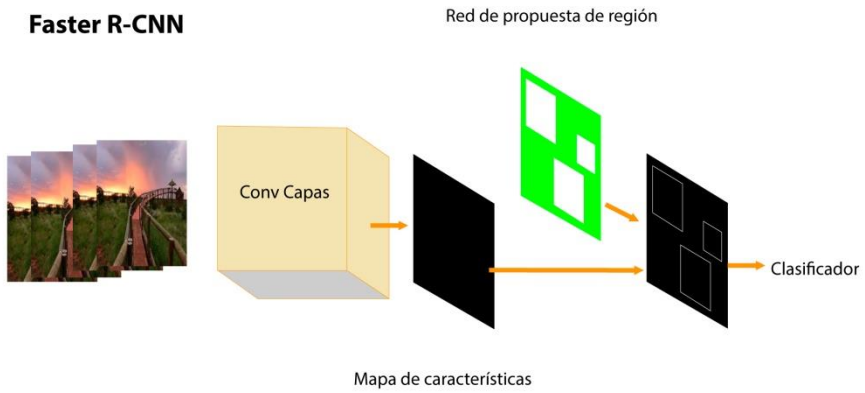

Figura 12. División de capaz Faster RCNN

Adaptado de Ren, He, Girshick, y Sun (2016)

Cada vector de características alimenta a una secuencia de capas completamente conectadas (fc) que finalmente se ramifican en dos capas de salida hermanas: una que produce estimaciones de probabilidad *softmax* sobre K clases de objetos más una clase de "fondo" general y otra capa que genera valores para cada una de las K clases de objetos. Cada conjunto de valores codifica posiciones de cuadro delimitador refinadas para una de las clases K.

4.3.2 Single-Shot Multibox Detection (SSD)

Aprovechando los clasificadores establecidos de redes neuronales convolucionales profundas, como la red VGG, los autores de esta investigación proponen la Single-Shot Multibox Detection (SSD) (Liu, *et al.,* 2016) de disparo único para una tarea de detección de objetos.

SSD ofrece un nuevo enfoque de detección de objetos que se separa de los enfoques basados en regiones anteriores (R-CNN, Fast RCNN y Faster R-CNN). SSD incorpora predicción de clase y predicción de cuadro delimitador en un solo proceso, por lo tanto, la velocidad de detección de SSD es significativamente más rápida que el enfoque Faster R-CNN.

La arquitectura SSD se la puede apreciar en la figura 6. Consiste en una red neuronal convolucional base4 seguida de las capas convolucionales de múltiples cajas. El VGG base y las capas de cuadro múltiple predicen la presencia de instancias de clase de objeto y sus ubicaciones de cuadro delimitador.

Hay seis capas de predicción que disminuyen en el tamaño del mapa de características que permite que el SSD maneje la detección de objetos en múltiples escalas. El total de mapas de características de seis capas con los tamaños: 382, 192, 102, 5 2, 3 2 y 12, se implementa con los siguientes números de cuadros predeterminados: 4, 6, 6, 6, 4 y 4. Por lo tanto, hay 8732 detecciones por clase, es decir, 382 × 4 + 192 × 6 + 102 × 6 + 52 × 6 + 32 × 4 + 12 × 4.

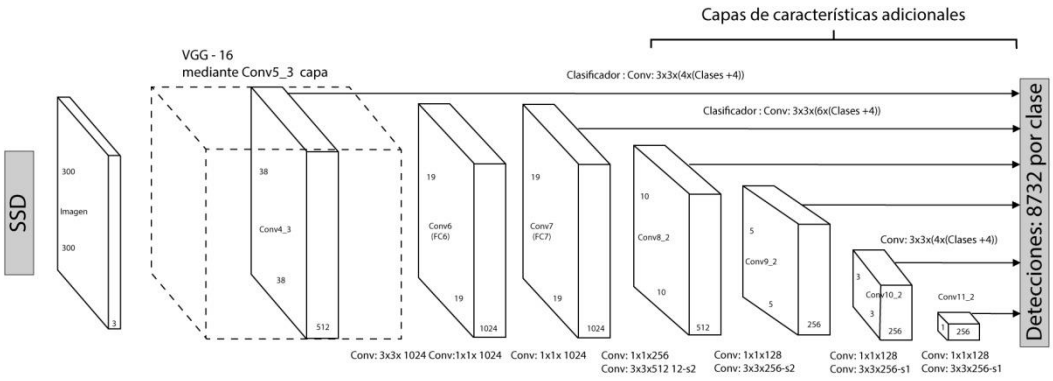

Figura 13. Single Shot Multibox Detection SSD
Adaptado de: Liu, Anguelov, Erhan, Christian Szegedy, Reed, Fu, Berg (2016)

La red neuronal convolucional base se refiere a una red CNN sin capas de clasificación. En estas detecciones, muchos serán candidatos poco probables con una probabilidad muy pequeña y se eliminarán mediante la supresión no máxima (NMS).
Los experimentos con SSD se llevan a cabo utilizando la red VGG base con el modelo de Clases de objetos visuales (VOC) PASCAL y la implementación más reciente de MobileNet de Google Tensorflow con el mod de conjunto de datos COCO pre-entrenado que se está utilizando en esta investigación.

4.4 *Transfer Learning* y las estrategias de uso de modelos pre-entrenados

Cuando se utiliza un modelo pre-entrenado para necesidades propias, se comienza eliminando el clasificador original, para luego agregar uno que se adapta a los propósitos propios, y finalmente se debe ajustar el modelo de acuerdo con una de tres estrategias:

Entrena a todo el modelo. En este caso, utiliza la arquitectura del modelo pre-entrenado y lo entrena de acuerdo con su conjunto de datos. Se está aprendiendo el modelo desde cero, por lo que se necesitará un gran conjunto de datos (y mucha potencia computacional).

Entrena algunas capas y deja las otras congeladas. Las capas inferiores del modelo se refieren a características generales (independientes del problema), mientras que las capas superiores se refieren a características específicas (dependientes del problema).
Se debe balancear esta dicotomía eligiendo cuánto se quiere ajustar los pesos de la red (una capa congelada no cambia durante el entrenamiento). Por lo general, si tiene un conjunto de datos pequeño y una gran cantidad de parámetros, dejará más capas congeladas para evitar el sobreajuste. Por el contrario, si el conjunto de datos es grande y el número de parámetros es pequeño, se puede mejorar el modelo entrenando más capas para la nueva tarea, ya que el sobreajuste no es un problema.

Congelar la base convolucional. Este caso corresponde a una situación extrema del intercambio entrenamiento / congelación. La idea principal es mantener la base convolucional en su forma original y luego usar sus salidas para alimentar el clasificador. Se está utilizando el modelo pre-entrenado como un mecanismo de extracción de características fijas, que puede ser útil si se tiene poca potencia computacional, el conjunto de datos es pequeño y / o el modelo pre-entrenado resuelve un problema muy similar al que se quiere resolver.
La Figura 14 presenta estas tres estrategias de manera esquemática.

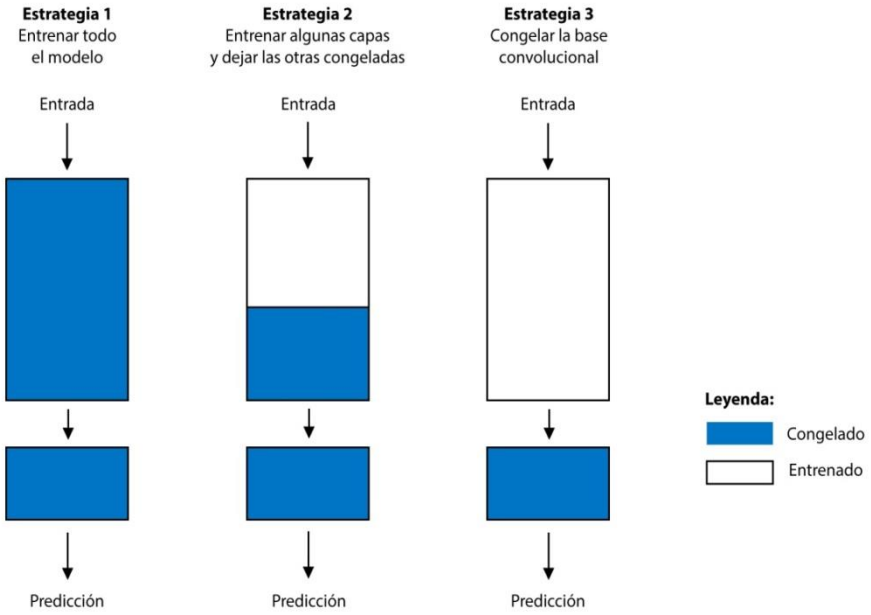

Figura 14. Estrategias de ajuste
Adaptado de: Marcelino (2018)

A diferencia de la estrategia 3, cuya aplicación es sencilla, la estrategia 1 y la estrategia 2 requieren que se tenga cuidado con la tasa de aprendizaje utilizada en la parte convolucional. La tasa de aprendizaje es un hiper parámetro que controla cuánto ajustan los pesos de la red. Cuando se usa un modelo pre-entrenado basado en CNN, es inteligente usar una pequeña tasa de aprendizaje porque las altas tasas de aprendizaje aumentan el riesgo de perder conocimiento previo. Asumiendo que el modelo pre-entrenado ha sido bien entrenado, mantener una tasa de aprendizaje pequeña asegurará que no distorsione los pesos de la CNN pronto y demasiado.

Se brindarán mayores detalles sobre CNN más adelante.

4.4.1 Proceso de *Transfer Learning* aplicados a los modelos de esta Tesina

Desde una perspectiva práctica, todo el proceso de *Transfer Learning* se puede resumir de la siguiente manera:

Selección de un modelo pre-entrenado. De la amplia gama de modelos pre-entrenados que están disponibles, se debe elegir uno que se adapte a su problema. Por ejemplo, si está utilizando Keras, inmediatamente se tiene acceso a un conjunto de modelos, como VGG , InceptionV3 y MobileNet, entre otros.

Clasificación del problema de acuerdo con la Matriz de similitud de tamaño. La Figura 15 muestra una matriz que guía en las elecciones. Esta matriz clasifica el problema de visión por computadora teniendo en cuenta el tamaño del conjunto de datos y su similitud con el conjunto de datos en el que se formó el modelo previamente entrenado. Como regla general, se considera que un conjunto de datos es pequeño si tiene menos de 1000 imágenes por clase. Con respecto a la similitud del conjunto de datos, debe prevalecer el sentido común. Por ejemplo, si la tarea es identificar gatos y perros, ImageNet sería un conjunto de datos similar porque tiene imágenes de gatos y perros. Sin embargo, si la tarea es identificar células cancerosas, ImageNet no puede considerarse un conjunto de datos similar.

Afinar el modelo. Aquí se puede usar la Matriz de similitud de tamaño para guiar la elección y luego consultar las tres opciones que se mencionaron antes sobre la reutilización de un modelo previamente entrenado. La Figura 16 proporciona un resumen visual de cada uno de los cuatro cuadrantes de esta matriz:

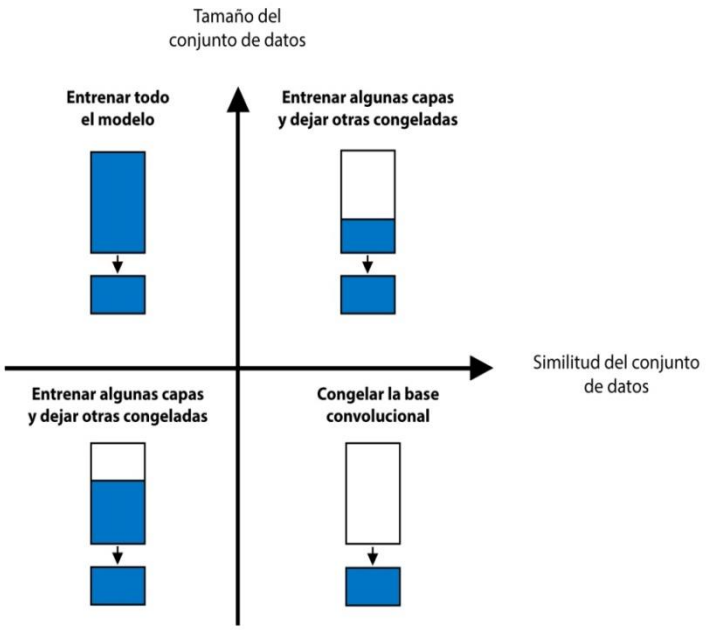

Figura 15. Mapa de decisión para ajustar modelos pre-entrenados
Adaptado de: Marcelino (2018)

Cuadrante 1. Gran conjunto de datos, pero diferente del conjunto de datos del modelo pre-entrenado. Esta situación lleva a la Estrategia 1. Como se tiene un gran conjunto de datos, se puede entrenar un modelo desde cero y hacer lo que quiera. A pesar de la diferencia de conjunto de datos, en la práctica, aún puede ser útil inicializar el modelo a partir de un modelo previamente entrenado, utilizando su arquitectura y pesos.

Cuadrante 2. Gran conjunto de datos y similar al conjunto de datos del modelo pre-entrenado. Esta es la situación ideal. Cualquier opción funciona. Probablemente, la opción más eficiente es la Estrategia 2. Como se tiene un gran conjunto de datos, el sobreajuste no debería ser un problema, por lo que se puede aprender todo lo que se quiera. Sin embargo, dado que los conjuntos de datos son similares, se puede evitar el esfuerzo de aprendizaje al aprovechar el conocimiento previo. Por lo tanto, debería ser suficiente entrenar el clasificador y las capas superiores de la base convolucional.

Cuadrante 3. Conjunto de datos pequeño y diferente del conjunto de datos del modelo pre-entrenado. Al contrario de el cuadrante 2, esta es la peor situación. En este caso se debe optar por la estrategia 2. Se hará difícil encontrar un equilibrio entre la cantidad de capas para entrenar y congelar. Si se profundiza, el modelo se puede sobreajustar, si permanece en el extremo poco profundo del modelo, no aprenderá nada útil. Probablemente, se deberá profundizar más que en el Cuadrante 2 y se deberá considerar las técnicas de aumento de datos.

Cuadrante 4. Conjunto de datos pequeño, pero similar al conjunto de datos del modelo pre-entrenado. En consecuencia, se debería optar por la Estrategia 3. Solo se necesita eliminar la

última capa completamente conectada (capa de salida), ejecutar el modelo previamente entrenado como un extractor de características fijas y luego usar las características resultantes para entrenar a un nuevo clasificador.

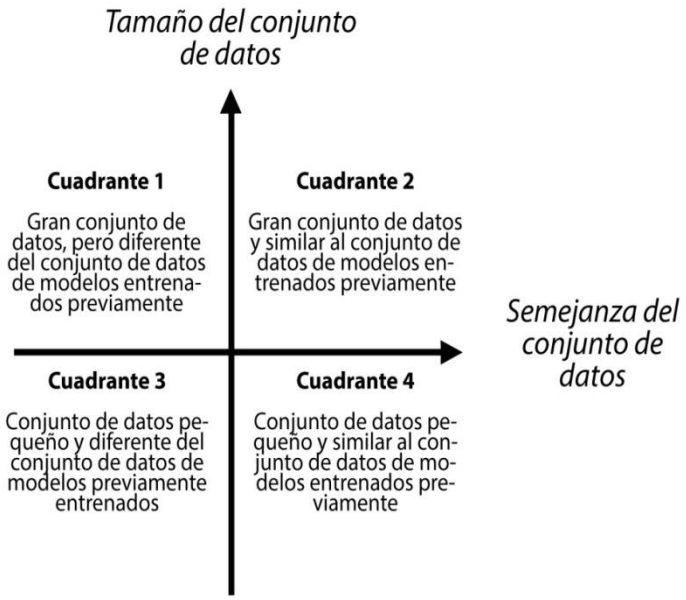

Figura 16. Matriz de similitud de tamaño

Adaptado de: Marcelino (2018)

4.5 Construcción del entorno de prueba

Como se ha visto anteriormente, donde se ha detallado los modelos disponibles para hacer las pruebas, a continuación se procederá a detallar como serán y en qué condiciones. El hardware más importante que se necesita para la máquina de aprendizaje es la Unidad de Procesamiento Gráfico (GPU) ya que el aprendizaje implica una gran cantidad de cálculos matemáticos definitivos.

Para realizar un cálculo de grandes dimensiones como es la multiplicación de matrices, la Unidad Central de Procesamiento (CPU) normal es inadecuada ya que podría consumir una mayor duración de la fase de entrenamiento. Por lo tanto, en este proyecto se ha utilizado hardware basado en GPU.

La lista de hardware y software que se utilizó en este trabajo de investigación fue:

NVIDIA GeForce GTX 1080 TI GPU

CUDA versión 10, CUDNN versión 7 Sistema operativo Windows 10 el marco de trabajo TensorFlow versión 1.13 Intel core i7 7700k 3.9ghz con una arquitectura de cuatro núcleos con hyperthreading.

Entre las diversas herramientas que provee Tensorflow está la posibilidad de decidir si utilizar CPU o GPU.
Se ha decidido no utilizar gpu para esta investigación porque los tesistas no tienen disponible una de ella y se han hecho unas pruebas de entrenamientos solamente con cpu si bien el tiempo de entrenamiento aumenta considerablemente se han logrado los objetivos propuestos. Otra consideración es que utilizando solo cpu es más accesible. De esta manera esta aplicación puede ser mantenida con menor requerimiento de hardware.

En cuanto al software, se ha utilizado el repositorio IA de Tensorflow Object Detection. *Git* para versionar el código, utilizando branchs apuntado a gestionar las diferentes configuraciones que se necesitan a los fines de ejecutar cada modelo.*Anaconda* y así manejar las versiones de python.

Las librerías de python que se están utilizando son:
protobuf, pillow, lxml, Cython, Jupyter, matplotlib, pandas, Opencv-python

Los detalles más específicos en cuanto al proceso de configuración de este serán detallados en el siguiente capítulo.

4.5.1 Definición del Dataset

Un conjunto de datos (dataset) es una colección de datos. En otras palabras, un conjunto de datos corresponde al contenido de una única tabla de base de datos, o una única matriz de datos estadísticos, donde cada columna de la tabla representa una variable particular y cada fila corresponde a un miembro dado del conjunto de datos en cuestión.
Es el caso de esta tesina, cada dato es una imagen de un producto asociado a una metadata o *label* como se refleja en la figura 17 el cual ha sido creado en el proceso de creación de dataset y *labeling* que se entrará en detalle en el siguiente capítulo.

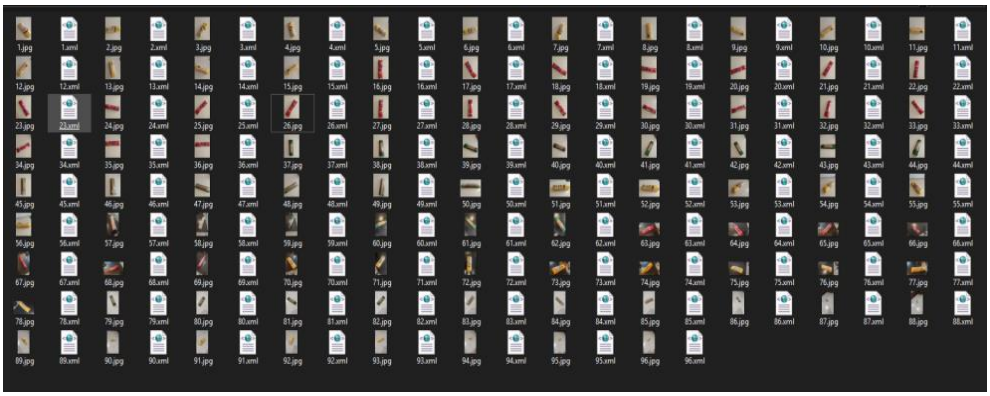

Figura 17. Dataset

Imágenes con un xml de metadatos. Elaboración propia (2020)

En los proyectos de ML, depende en gran medida de los datos, sin datos, es imposible que una “IA” aprenda. Es el aspecto crucial que hace posible el entrenamiento de algoritmos. No importa cuán grande sea el equipo de AI o el tamaño del conjunto de datos, si el conjunto de datos no es lo suficientemente bueno, la red no funcionara.

4.5.1.1 Dataset de Entrenamiento y Dataset de Validación

El conjunto de datos de entrenamiento es el que se usa para entrenar un algoritmo para comprender cómo aplicar conceptos como redes neuronales, para aprender y producir resultados. Incluye tanto los datos de entrada como la salida esperada.

El conjunto de datos de validación se utiliza para evaluar qué tan bien se entrenó su algoritmo con el conjunto de datos de entrenamiento. En los proyectos de IA, no se puede usar el conjunto de datos de entrenamiento en la etapa de prueba porque el algoritmo ya sabrá de antemano el resultado esperado, que no es el objetivo de esta tesina.

Es importante saber que del conjunto de datos completo debe ser fraccionado en dos subconjuntos, uno de un 80~90% del conjunto original para el entrenamiento en sí y el 10~20% restante para su validación.

4.5.1.2 Data augmentation

Una característica importante del software que está disponible para los usuarios y es usada en esta tesina es *data augmentation*, que consiste en aplicar cambios geométricos a todas las imágenes del dataset como volteos, recortes, rotaciones, traducciones de imágenes, cambios de color con el fin de aumentar nuestro dataset hasta 10 veces del tamaño inicial.

Ampliar el conjunto de datos con métodos de aumento de datos no solo es útil para el desafío de los datos limitados sino también puede reducir el *overlifting*[10] y mejorar la generalización de los modelos de esta tesina porque aumenta la diversidad de nuestro conjunto final.

4.5.2 Descripción las pruebas:

El objetivo principal de las pruebas no es buscar performance determinado en un modelo, es simplemente elegir el mejor para nuestro objetivo en base a ciertas métricas que se describirá más adelante.
Como se menciona anteriormente se tienen tres modelos a comparar en pruebas controladas de manera justa, serán pruebas de entrenamiento, midiendo capacidad de cada modelo de aprender con el mismo dataset (97 imágenes) tanto de entrenamiento (77 imágenes) como de validación (17 imágenes), durante un periodo de 160 minutos.
Se utilizará como se definió anteriormente técnicas de *data augmentation* para incrementar el tamaño de nuestro dataset inicial en cada etapa.

4.5.2.1 Parámetros a evaluar

Durante las pruebas y en el transcurso del entrenamiento de cada modelo en la investigación de esta tesina, se van a monitorear dos métricas fundamentales para el objetivo propuesto.

- Función de Pérdida (*Loss function*): Esta medición será realizada en cada iteración, con mucha frecuencia sobre el dataset de entrenamiento a través de *TensorBoard* mientras se ejecuta el entrenamiento.
- *Accuracy*: Por otro lado, se tiene el *accuracy* que será medido cada 5 minutos cuando se ejecuta el algoritmo de validación sobre del dataset de validación. Se observarán y se tomarán nota de las métricas durante el proceso de 160 minutos.

4.5.2.1.1 Loss Function

Las CNN aprenden mediante una función de pérdida (loss function). Es un método para evaluar qué tan bien un algoritmo específico modela los datos dados.
El entrenamiento en sí consiste en ajustar los pesos de cada neurona utilizando la función de optimización en base a la función de pérdida.
La función de perdida siempre debe ser una función logarítmica decreciente tendiendo a cero, de otra manera significa que se originó un problema en el entrenamiento.

[10] Overlifting o sobreajuste: es el efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado. Recuperado de: https://es.wikipedia.org/wiki/Sobreajuste

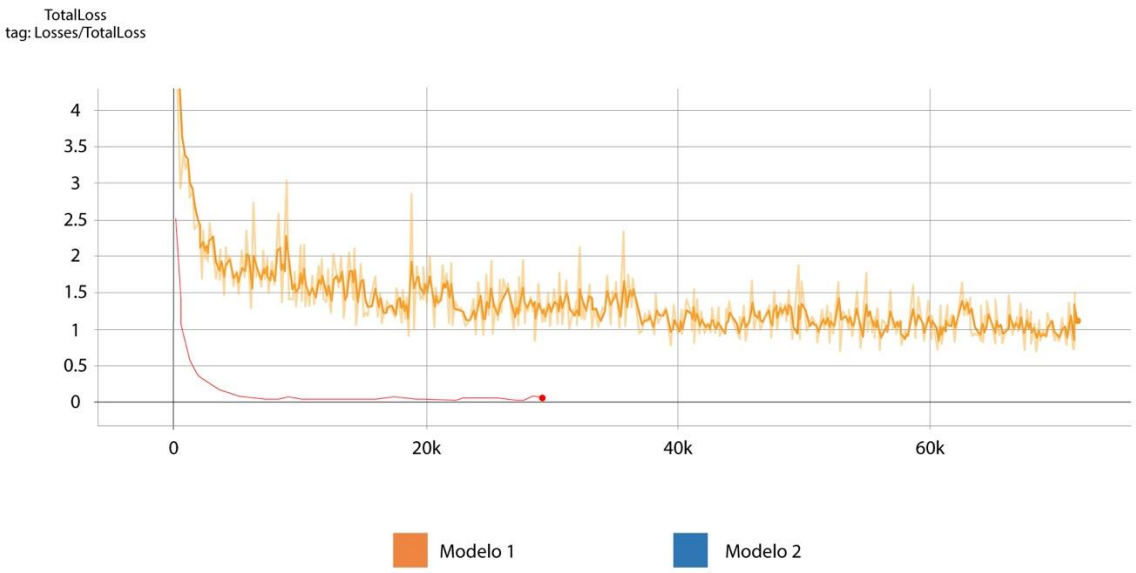

Figura 18. Ejemplo visual de comparación entre funciones de pérdida

Fuente: elaboración propia (2020)

No existe una función de pérdida para todos los algoritmos en el aprendizaje automático, ello significa que cada modelo responde y se comporta diferente en base a su arquitectura como se simboliza en la figura 18.
Allí se tiene una comparación de dos modelos sobre las funciones de perdida. No se puede afirmar o establecer un valor deseado en la función de pérdida, pero si se puede esperar un patrón de algoritmo decreciente tendiendo a 0 a lo largo del entrenamiento durante los 160 min.

4.5.2.1.2 *Accuracy*

La exactitud de clasificación o tasa de aserción (*Accuracy* en inglés) es lo que usualmente se quiere decir cuando se usa el término exactitud.
La definición de la misma indica la relación entre el número de predicción es correctas y el número total de muestras de entrada.
No se puede establecer un valor deseado de *accuracy*, ya que al igual que la función de pérdida, es una función logarítmica pero creciente tendiendo a 1 o 100%.
Se acercará a 1 en base a la calidad del dataset, el tiempo de entrenamiento, y la capacidad computacional.

4.6 Resultados obtenidos

4.6.1 RCNN

El entrenamiento se detiene manualmente a los 160 minutos con una pérdida de 0.027 a ese punto, La Figura 19 muestra la función de pérdida de entrenamiento. Para Fast-RCNN, este valor se logra a los 4388 pasos.
Se logra así una velocidad de ejecución por paso de 2.18 segundos. En cuanto al *accuracy* sobre el dataset de validación se logra un 94% de tasa de precisión, reconociendo con éxito 16 imágenes de 17 como se refleja en la figura 20.

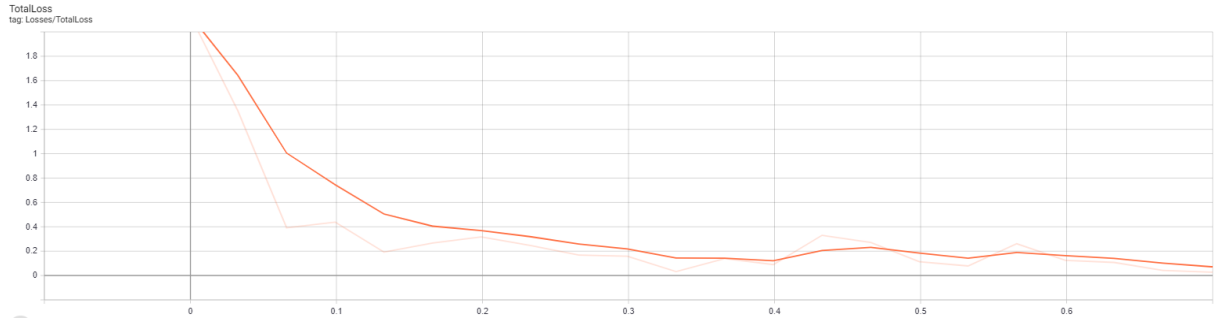

Figura 19. Funcion de Pérdida, TensorBoard
(Elaboración propia, 2020)

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.929
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 1.000
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000

Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000

Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.929
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.947
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.947
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.947
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000

Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000

Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.947
```

Figura 20. Accuracy
Un promedio de 94.7%. (Elaboración propia, 2020)

4.6.2 SSD MobileNet

El entrenamiento se detiene manualmente a los 160minutos con una pérdida de 2.264 a ese punto, La Figura 21 muestra la función de pérdida de entrenamiento. Para SSD MobileNet, este valor se logra a los 479 pasos.
Se logra así una velocidad de ejecución por paso de 20 segundos. En cuanto al *accuracy* sobre el dataset de validación se logra un 73% de tasa de precisión, reconociendo con éxito 12 imágenes de 17 como se refleja en la Figura 22.

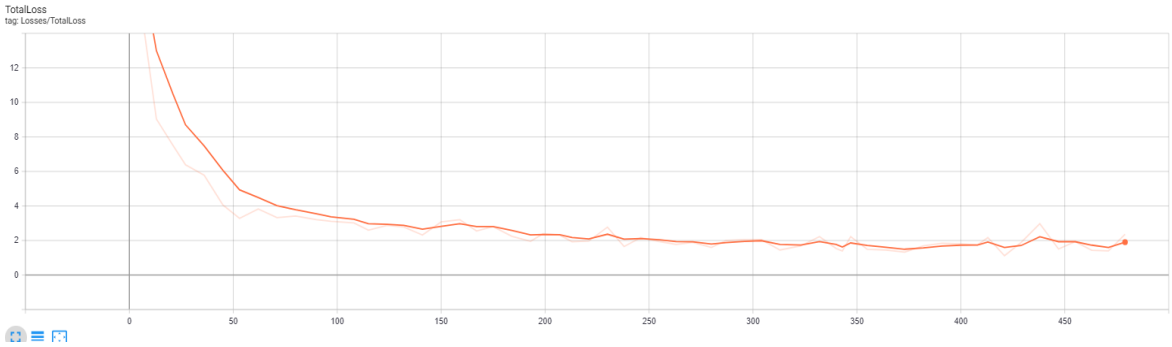

Figura 21. Funcion de Perdida, TensorBoard

(Elaboración propia, 2020)

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.587
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.817
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.729
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.587
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.710
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.734
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.734
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.734
```

Figura 22. Accuracy un promedio de 73.4%
(Elaboración propia, 2020)

4.6.3 SSD Inception

Con respecto a SSD Inception, el entrenamiento se detiene manualmente a los 160 minutos con una pérdida de 1.736 a ese punto, La Figura 23 muestra la función de pérdida de entrenamiento. Para SSD Inception, este valor se logra a los 401 pasos.

Se logra así una velocidad de ejecución por paso de 24 segundos. En cuanto al *accuracy* sobre el dataset de validación se logra un 65% de tasa de precisión, reconociendo con éxito 11 imágenes de 17 como se refleja en la Figura 23.

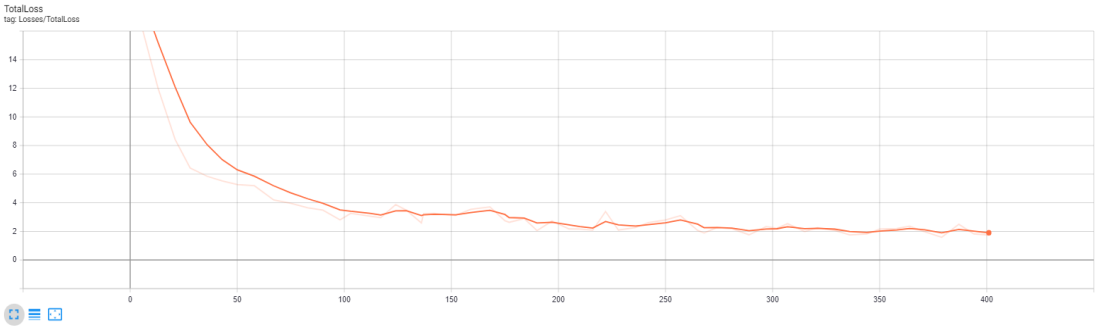

Figura 23. Función de Perdida, TensorBoard
(Elaboración propia, 2020)

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.621
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.985
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.745
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.621
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.656
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.659
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.659
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.659
```

Figura 24. Accuracy un promedio de 65.9%
(Elaboración propia, 2020)

A lo largo de este capítulo los tesistas han investigado algunos aspectos profundos del *machine learning*, tipos de modelos, extractores y convoluciones. Se han seleccionado tres modelos, y se los han comparado en entornos equivalentes.

Se han comparado sus entrenamientos unos con otros, para decidir cuál es el que adquiere conocimiento más rápido y logra mejores rendimientos, de esta manera se puede elegir el óptimo para seguir adelante con el desarrollo de reconocimiento de alimentos para el Banco.

4.7 Conclusiones

El objetivo de este capítulo fue el de comparar los tres modelos planteados en iguales circunstancias, en un lapso de entrenamiento establecido. Con estos detalles se ha llegado a conclusiones significativas.
Los experimentos sugirieron que el enfoque de TL en tareas de detección de objetos ofrece un resultado excelente. Se pudo lograr una mayor precisión a través del aprendizaje profundo
En este trabajo, se logró comparar bajo rigor científico los tres modelos y demostrar sus prestaciones. Además, referido a la detección de objetos marco, se pudo identificar la ubicación de los alimentos con un número moderado de muestras.

De las pruebas realizadas con los tres modelos, se puede concluir que Faster-RCNN es el modelo que brindo mayor precisión con un *accuracy* del 94.7% como muestra la Tabla 2.
El modelo FRCNN funciona bien en la detección de objetos y rápida convergencia durante la fase de entrenamiento.
En cuanto a SDD produjo un mayor número de falsos negativos en nuestros experimentos, que no es deseable en la detección de ningún objeto en el marco de referencia.

Modelos	*Accuracy*
Faster_rcnn_inception_coco	*94.7%*
Ssd_mobilenet_coco	*73.4%*
ssd_inception_coco	*65.9%*

Tabla 2. Resultados
Elaboración propia (2020)

Para lograr la investigación de este capítulo se necesitó crear tanto el entorno como el dataset que se ha utilizado. En el próximo capítulo se explicará con mayor detalle sobre el proceso de desarrollo de la aplicación del *framework tensor flow object detection models*, detallando las configuraciones y los problemas que se ha tenido durante la investigación.

CAPÍTULO 5: RECONOCIMIENTOS DE PRODUCTOS

5.1 Introducción

En este capítulo se describe el proceso de desarrollo de detección de objetos diseñado por los tesistas utilizando modelos pre entrenado y, además, el *Framework Object Detection* de *TensorFlow*.

En el Anexo 4 se describen las características del lenguaje TensorFlow o Keras que utilizan Python como lenguaje de programación, así como de sus herramientas.

Al utilizar el *Framework TensorFlow Object Detection* se tienen algunas ventajas las cuales serán descriptas a continuación:

1. Permite cambiar de modelo para entrenamiento de manera fácil: Como los tesistas han hecho en el modelo de pruebas, se puede cambiar el modelo de uno a otro rápidamente, esto puede ser útil, ya que brinda flexibilidad en las aplicaciones. Si se quiere utilizar esta aplicación (*app* abreviado) en un celular o se percibe que en el futuro saldrán mejores modelos pre-entrenados, el usuario puede directamente migrar la aplicación sin mayores inconvenientes.
2. Existe un amplio soporte de la comunidad de programadores de lenguajes abiertos para resolver situaciones de errores y problemas de performance. Al estar estandarizado el proceso de desarrollo existe un marco de conocimientos para poder avanzar y resolver problemas.
3. Se dispone de mucha documentación: se puede encontrar una vasta cantidad de contenidos acerca de reconocimiento de productos con *TensorFlow ObjectDetection* ya sean videos, cursos y detalles de performance.
4. *TensorFlow* o Keras son productos muy maduros: ambos son ampliamente utilizados. Son las dos alternativas más robustas hoy en día para la aplicación desarrollada por los tesistas.

5.2 Configuración del Dataset

El proceso diseñado por los tesistas se inicia con la creación del dataset de imágenes. Se han utilizado tres clases de productos las cuales han sido definidas por el label Map el cual es un archivo de configuración que define la interfaz entre la red y el dataset, uniendo cada clase con la metadata de cada imagen.

```
item {
  id: 1
  name: 'spaguetti matarazzo'
}

item {
  id: 2
  name: 'polenta presto pronta 750g'
}

item {
  id: 3
  name: 'spaguetti canale'
}
```

Figura 25. LabelMap

Nota. Definiendo nuestras tres clases (elaboración propia, 2020)

Por otra parte, se tiene el archivo de *metadata/annotation*, en el cual, por cada imagen del dataset, aquél contiene diversos datos, donde uno de ellos es el nro. de clase *name,* que además es el más importante, ya que hace el enlace entre esta imagen y su clase en la red.

```
<annotation>
  <folder>train</folder>
  <filename>17.jpg</filename>

<path>C:\Users\agust\Documents\ia\models\research\object_detection\images\train\17.jpg</path
>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>4032</width>
    <height>3024</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>spaguetti matarazzo</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>857</xmin>
      <ymin>134</ymin>
      <xmax>3948</xmax>
      <ymax>2703</ymax>
    </bndbox>
```

Figura 26. . Estructura de las metadata de cada imagen del dataset
(Elaboración propia, 2020)

El archivo que contiene la anotación de cada imagen es un XML, que posee información de donde se ubica el objeto en la imagen, y a qué clase pertenece utilizando el campo *name*.

La configuración de cada *annotation* por cada imagen es un progreso largo, y se debe hacer cuidadosamente ya que cada input afecta el rendimiento de la red soporte del proceso.

Para este proceso se utiliza una herramienta llamada *LabelImg* como referencia la imagen de la figura 27. Esta herramienta es muy útil, ya que aporta una interfaz de edición por imagen, que permite hacer el proceso de *labeling* para todo el dataset, imagen por imagen. La captura de pantalla diseñada se la presenta como una muestra de su interfaz:

Figura 27. ImgLabel

*Nota.*Herramienta con una interfaz para ejecutar el procesamiento de labeling (Elaboración propia, 2020).

Cuando se tiene un dataset con sus imágenes y su metadata correspondiente tal como se referencia en la figura 28, queda para el programador un directorio con todas las imágenes y su .xml correspondiente.

Abajo se muestra el directorio en la siguiente imagen gráfica.

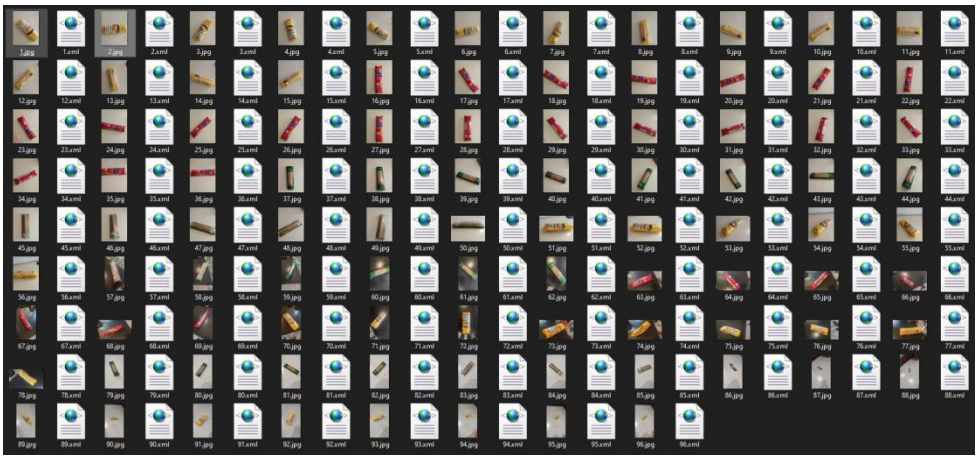

Figura 28. Dataset definido

(Elaboración propia, 2020)

5.3 Complicaciones

Al inicio de la ejecución de la aplicación, realizada mediante un proceso manual, han surgido diversas complicaciones y ello ha obligado a que la ejecución de cada iteración deba ser minuciosa.

Algunas de las fallas que se han detectado en el proceso de ejecución han sido:

- Algunas veces la aplicación *LabelMap* no funciona bien, se ha tenido que investigar esta falla utilizando los procedimientos de reinstalar la aplicación y el lenguaje python.

- Ha sido necesario realizar la operación de reducción del tamaño de las imágenes, ya que éstas fueron capturadas por un celular de 4 MB de espacio ocupado por imagen almacenada, y se ha tenido que reducirlas con la herramienta de edición de windows, obteniendo así un promedio 400 KB de espacio ocupado por imagen almacenada. Es preciso tener en cuenta que la resolución de cada imagen afecta en el rendimiento de las convoluciones. El detalle que proporciona una cámara 12 mega píxeles es excesivo al buscar patrones, si bien las neuronas poseen la funcionalidad para reducir imágenes, este proceso demora el entrenamiento haciéndolo una por una cada vez que itera en la red, por lo tanto, es más eficiente hacer la reducción en esta etapa.

Una vez que el proceso de input ha sido terminado, se ha procedido a configurar el framework como lo indica el siguiente punto.

5.4 Configuración del Framework

Para cumplir esta operación, es necesario descargar el repositorio de Tensor Flow Object Detection Models*. La ruta elegida es models\research\object_detection, y posteriormente se ha activado el entorno de anaconda con python 1.15. Cuando se necesitan dependencias se las puede instalar utilizando pip tal como muestra la figura 29, utilizando la siguiente línea de comandos.*

```
conda create -n tensorflow1 pip python=3.5
activate tensorflow1
pip install --ignore-installed --upgrade tensorflow-gpu
conda install -c anaconda protobuf
pip install pillow
pip install lxml
pip install Cython
pip install jupyter
pip install matplotlib
pip install pandas
pip install opencv-python
```

Figura 29. Instalar dependencias
(Elaboración propia, 2020)

Una vez logrado construir el entorno con todas las dependencias instaladas, se ha procedido a agregar el módulo de slim disponible de la carpeta models del repositorio que ha sido descargada anteriormente a las variables de entorno PYTHONPATH, en la siguiente figura 30 se muestra la línea de comandos para comprobar el estado de la configuración.

```
set
PYTHONPATH=C:/tensorflow1/models;C:/tensorflow1/models/research;C:/tensorflow1/mode
ls/research/slim
echo %PYTHONPATH%
set PATH=%PATH%;PYTHONPATH
echo %PATH%
```

Figura 30. Configurar variables de entorno
(Elaboración propia, 2020)

5.4.1 El Proceso de Compilación de los Protobuf

Es importante entender qué los buffers de protocolo, generalmente conocidos como Protobuf, son un protocolo desarrollado por Google para permitir la construcción en serie y aleatoria de datos estructurados.
Google lo desarrolló con el objetivo de proporcionar una mejor manera, en comparación con XML, para que los sistemas se comuniquen.
Por lo tanto, los desarrolladores se centraron en hacerlo más simple, también más pequeño y rápido, así como más fácil de mantener que XML.
Se ha observado, como se indica en este artículo, que este protocolo incluso ha superado a JSON con un mejor rendimiento y capacidad de mantenimiento, así como un tamaño más pequeño.

Diferencias de JSON y Protobuf

Es importante tener en cuenta que, aunque los mensajes JSON y Protobuf se pueden usar indistintamente, estas tecnologías se diseñaron con diferentes objetivos.
JSON, cuyo significado es JavaScript ObjectNotation, es simplemente un formato de mensaje que surgió de un subconjunto del lenguaje de programación JavaScript.
Los mensajes JSON se intercambian en formato de texto y, hoy en día, son completamente independientes, también son compatibles con, prácticamente, todos los lenguajes de programación.

Protobuf, por otro lado, es más que un formato de mensaje, también es un conjunto de reglas y herramientas para definir e intercambiar estos mensajes.
Google, el creador de este protocolo, lo ha hecho de código abierto y proporciona herramientas para generar código para los lenguajes de programación más utilizados, como JavaScript, Java, PHP, C #, Ruby, Objective C, Python, C ++ y Vamos.
Además de eso, Protobuf tiene más tipos de datos que JSON, como enumeraciones y métodos, y también se usa mucho en RPC (llamadas a procedimiento remoto).

Algunas consideraciones Referidas a la Velocidad de *Protobuf* Respecto a JSON

Hay varios recursos en línea que muestran que *Protobuf* funciona mejor que JSON y XML, aunque siempre es importante verificar si este es el caso para las necesidades del usuario.

La razón principal para crear estos dos escenarios, Java a Java y JavaScript a Java, fue medir cómo se comportaría este protocolo en un entorno empresarial como Java y también en un entorno donde JSON es el formato de mensaje nativo.

Es decir, lo que los tesistas muestran aquí son datos de un entorno donde JSON está integrado y debe funcionar extremadamente rápido (motores de JavaScript) y de un entorno donde JSON no es un ciudadano de primera clase, lo que implica que las funciones no pueden ser manipuladas como un valor. Para analizar y comparar las velocidades de Protobuf y JSON, se debe tener en cuenta el volumen y características de los datos que se han reunido en los experimentos.

Los detalles de los mismos son los siguientes:

Tensor Flow Object Detection utiliza *protobuf* para archivos de configuración convirtiendo archivos *.pya.proto, ya que* ésto es necesario para la compilación del proceso de entrenamiento. Se han ejecutado los siguientes comandos (ver figura 31) dentro de *models\research\object_detection* para hacer esta conversión:

```
protoc --python_out=. .\object_detection\protos\anchor_generator.proto
.\object_detection\protos\argmax_matcher.proto
.\object_detection\protos\bipartite_matcher.proto
.\object_detection\protos\box_coder.proto
.\object_detection\protos\box_predictor.proto
.\object_detection\protos\eval.proto
.\object_detection\protos\faster_rcnn.proto
.\object_detection\protos\faster_rcnn_box_coder.proto
.\object_detection\protos\grid_anchor_generator.proto
.\object_detection\protos\hyperparams.proto
.\object_detection\protos\image_resizer.proto
.\object_detection\protos\input_reader.proto
.\object_detection\protos\losses.proto
.\object_detection\protos\matcher.proto
.\object_detection\protos\mean_stddev_box_coder.proto
.\object_detection\protos\model.proto
.\object_detection\protos\optimizer.proto
.\object_detection\protos\pipeline.proto
.\object_detection\protos\post_processing.proto
.\object_detection\protos\preprocessor.proto
.\object_detection\protos\region_similarity_calculator.proto
.\object_detection\protos\square_box_coder.proto
.\object_detection\protos\ssd.proto
.\object_detection\protos\ssd_anchor_generator.proto
.\object_detection\protos\string_int_label_map.proto
.\object_detection\protos\train.proto
.\object_detection\protos\keypoint_box_coder.proto
.\object_detection\protos\multiscale_anchor_generator.proto
.\object_detection\protos\graph_rewriter.proto

python setup.py build
python setup.py install
```

Figura 31. Compilar protos
(Elaboración propia, 2020)

5.4.2 Convertir de XML a csv

Para optimizar los metadatos de las imágenes de este trabajo, éstos han sido transformadas a .csv utilizando el script disponible dentro del repositorio *object_detection/xml_to_csv.py*, teniendo en cuenta que por defecto la aplicación desarrollada utiliza los xml ubicados en la carpeta *object_detection/images*
Como salida ha sido creada en la misma carpeta los siguientes archivos *images/test_labels.csv* y*images/train_labels.csv,* tal como se muestra la estructura en la figura 32.

Comando:
python xml_to_csv.py

```
filename,width,height,class,xmin,ymin,xmax,ymax
1.jpg,4032,3024,polenta presto pronta 750g,807,1,3738,2746
2.jpg,4032,3024,polenta presto pronta 750g,1482,1,2802,2996
23.jpg,4032,3024,spaguetti matarazzo,673,228,3835,2599
```

Figura 32. Ejemplo de nuestra medata formateada en un .csv
(Elaboración propia, 2020)

5.4.3 TFRecords

Algunas consideraciones a tener en cuenta sobre los datos son que, si el usuario está trabajando con grandes conjuntos de datos, el uso de un formato de archivo binario para el almacenamiento de ellos puede tener un impacto significativo en el rendimiento de su canal de importación y, en consecuencia, en el tiempo de entrenamiento de su modelo.
Los datos binarios ocupan menos espacio en el disco, requieren menos tiempo para copiarse y pueden leerse de manera mucho más eficiente desde el disco. Esto es especialmente cierto si sus datos se almacenan en discos giratorios, debido al rendimiento de lectura / escritura mucho más bajo en comparación con los SSD.
Un archivo *TFRecord* almacena sus datos como una secuencia de cadenas binarias. Esto significa que debe especificar la estructura de sus datos antes de escribirlos en el archivo. Tensorflow proporciona dos componentes para este propósito: tf.train.Example y tf.train.Sequence Example. Debe almacenar cada muestra de sus datos en una de estas estructuras, luego serializarlos y usar un tf.python_io.TFRecordWriter para escribirlos en el disco.
Con los protos compilados, se ha procedido luego a compilar los annotations de cada imagen en TFRecords de manera de optimizar su formato y lectura como se muestra en la figura 33.
De esta manera se ha pasado de un *.xml* a un *.record* optimizado para el procesamiento en la red.

```
python xml_to_csv.py
python generate_tfrecord.py --csv_input=images/train_labels.csv
--image_dir=images/train --output_path=train.record
python generate_tfrecord.py --csv_input=images/test_labels.csv
--image_dir=images/test --output_path=test.record
```

Figura 33. Compilación de nuestras annotations
(Elaboración propia, 2020)

5.4.4 Los Modelos Pre-entrenados

Como se ha descripto en capítulos anteriores, *Transfer Learning* es un concepto que será utilizado en la implementación de los desarrollos de este trabajo a través de la aplicación de modelos pre entrenados.
Como se ha explicado anteriormente, un modelo pre-entrenado es un modelo que fue entrenado con un ingente conjunto de datos de referencia para resolver un problema similar al que el usuario quiere abordar.

Por múltiples razones como, el costo computacional del entrenamiento de este tipo de modelos, así como por la complejidad a la hora de elegir la arquitectura óptima de ellos, conducen a elegir a los métodos de *Transfer Learning*.
Además, debe tenerse en cuenta de que son aplicaciones bien conocidas y también son precisas. En consecuencia, todo ello ha llevado a convertirlas en una práctica común para el programador.

Ejemplos de ello son VGG-16, VGG-19, ResNet-50, SeNet-50, etc.

Se ha utilizado modelos disponibles para el *framework* de trabajo, Se ha descargado el faster_rcnn_inception_v2_coco, que como se ha investigado anteriormente, ha sido el que mejor se ha desempeñado en las pruebas realizadas de tres clases.

5.4.5 Entrenamiento y validación

Los tesistas han demostrado que, una vez logrado construir el entorno con la entrada configurada, así como el dataset procesado con las imágenes almacenadas, y además tener los *TFRecords* generados, se está en condiciones de proceder con el entrenamiento.

Para ello se ejecuta el siguiente comando en la figura 34 y en la figura 35 se detona la ejecución:

```
python train.py --logtostderr --train_dir=training/
--pipeline_config_path=training/faster_rcnn_inception_v2_pets.config
```

Figura 34. Ejecutar el entrenamiento

(Elaboración propia, 2020)

```
INFO:tensorflow:global step 492: loss = 1.4958 (10.717 sec/step)
I0824 23:30:09.643883  8888 learning.py:507] global step 492: loss = 1.4958 (10.717 sec/ste
p)
INFO:tensorflow:global step 493: loss = 2.1034 (12.219 sec/step)
I0824 23:30:21.875176  8888 learning.py:507] global step 493: loss = 2.1034 (12.219 sec/ste
p)
INFO:tensorflow:global_step/sec: 0.05505
I0824 23:30:45.930850 15508 supervisor.py:1099] global_step/sec: 0.05505
INFO:tensorflow:global step 494: loss = 2.4201 (24.024 sec/step)
I0824 23:30:47.316146  8888 learning.py:507] global step 494: loss = 2.4201 (24.024 sec/ste
p)
INFO:tensorflow:Recording summary at step 494.
I0824 23:30:58.020522 15864 supervisor.py:1050] Recording summary at step 494.
INFO:tensorflow:global step 495: loss = 2.2403 (17.766 sec/step)
I0824 23:31:05.083635  8888 learning.py:507] global step 495: loss = 2.2403 (17.766 sec/ste
p)
INFO:tensorflow:global step 496: loss = 1.8766 (9.008 sec/step)
I0824 23:31:14.092545  8888 learning.py:507] global step 496: loss = 1.8766 (9.008 sec/step
)
INFO:tensorflow:global step 497: loss = 1.8939 (11.671 sec/step)
I0824 23:31:25.778296  8888 learning.py:507] global step 497: loss = 1.8939 (11.671 sec/ste
```

Figura 35. Imagen de ejecución

Nota. Mostrando el error en cada iteración y información detallada de cada paso, (Elaboración propia, 2020).

Cada proceso de entrenamiento crea un Modelo de Inferencia del cual se detalla en el siguiente apartado, *TensorFlow* aporta ventajas para manipular y trabajar con modelos en general, como guardar puntos de control (*checkpoints* en inglés), y poder restaurarlos en determinado punto.

5.4.6 Modelo de inferencia

Como lo indica (Krizhevsky et al., 2017) Las redes neuronales son computacionalmente muy caras. A continuación se presenta la arquitectura de alexnet, que es una red neuronal convolucional relativamente simple:

El cálculo de la cantidad de variables necesarias para la predicción se describe en la figura 36 a continuación:

```
conv1 layer:  (11*11)*3*96 (weights) + 96 (biases)          = 34944
conv2 layer:  (5*5)*96*256 (weights) + 256 (biases)         = 614656
conv3 layer:  (3*3)*256*384 (weights) + 384 (biases)        = 885120
conv4 layer:  (3*3)*384*384 (weights) + 384 (biases)        = 1327488
conv5 layer:  (3*3)*384*256 (weights) + 256 (biases)        = 884992
fc1 layer:    (6*6)*256*4096 (weights) + 4096 (biases)      = 37752832
fc2 layer:    4096*4096 (weights) + 4096 (biases)           = 16781312
fc3 layer:    4096*1000 (weights) + 1000 (biases)           = 4097000
```

Figura 36. Ejemplo de cálculo de peso de las capas
(Elaboración propia, 2020)

Es decir que será necesario más de 60 millones de parámetros para calcular la predicción en una imagen.

También se tiene un número similar de gradientes que se calculan y se utilizan para realizar la propagación hacia atrás durante el entrenamiento.
Los modelos de *Tensorflow* contienen todas estas variables, por esta razón no se necesitan los gradientes cuando el usuario implementa su modelo en un servidor web.
De esta manera, no se precisa llevar toda esta carga, ya que la congelación es el proceso para identificar y guardar todos los elementos necesarios (gráficos, pesos, etc.) en un solo archivo que se puede usar fácilmente.

Un modelo de *Tensorflow* está conformado de:

modelo-ckpt.meta: Contiene el gráfico completo. [Contiene un buffer de protocolo *MetaGraphDef* serializado. Contiene *graphDef* que describe el flujo de datos, anotaciones para variables, canalizaciones de entrada y otra información relevante]
model-ckpt.data-0000-of-00001: Contiene todos los valores de las variables (pesos, sesgos, marcadores de posición, gradientes, hiper parámetros, etc.).
modelo-ckpt.index: metadatos. [Es una tabla inmutable (tensoflow :: table :: Table). Cada clave es un nombre de un tensor y su valor es un *BundleEntryProto* serializado. Cada *BundleEntryProto* describe los metadatos de un tensor]

Optimización para inferencia:
Para reducir la cantidad de cálculo necesario cuando la red se usa solo para inferencias, se puede eliminar algunas partes de un gráfico que solo se necesitan para el entrenamiento. Por ejemplo:

1. Eliminación de operaciones utilizadas solo para entrenamiento, como guardar puntos de control.
2. Eliminación de partes del gráfico no utilizados.
3. Eliminar operaciones de depuración como *CheckNumerics*.

Esto podría ser aplicable en la implementación en un servidor web en el cual se podría deshacerse de metadatos innecesarios, gradientes y variables de entrenamiento innecesarias y encapsular todo en un solo archivo. Este único archivo encapsulado (extensión .pb) se llama "*graphDef* congelado". Es esencialmente un búfer de protocolo *graphDef* serializado escrito en el disco. En otras palabras, un proceso de optimización.

5.4.7 Checkpoints

Una parte importante también que brinda el *framework* de *Object Detection* de *TensorFlow* con los modelos de Inferencia son los *checkpoints*, puntos de control, cada 5 min guardan *snapshot* del estado actual del entrenamiento guardando todos los archivos relacionados para luego poder retornar en caso de una interrupción, esta es una característica muy útil cuando tienes riesgos externos en la ejecución.

Los puntos de control se guardan en la carpeta *training/checkpoints*como*model.ckpt.index* y *model.ckpt.meta* como se ilustra en la figura 37 respectivamente.

```
model_checkpoint_path: "model.ckpt-487"
all_model_checkpoint_paths: "model.ckpt-296"
all_model_checkpoint_paths: "model.ckpt-328"
all_model_checkpoint_paths: "model.ckpt-363"
all_model_checkpoint_paths: "model.ckpt-400"
all_model_checkpoint_paths: "model.ckpt-487"
```

Figura 37. Checkpoints guardados 1

(Elaboración propia, 2020)

5.4.8 Exportar el Modelo

Para terminar el entrenamiento, alcanzando así un *accuracy* aceptable del 95% en base al dataset de validación. Se prueba el modelo con imágenes que no pertenezcan al dataset para reconocer los productos entrenados. Pero primero se exporta el modelo y se optimiza para producción.

```
python export_inference_graph.py --input_type image_tensor --pipeline_config_path
training/ssd_inception_v2_coco.config --trained_checkpoint_prefix
training/model.ckpt-487 --output_directory inference_graph
python Object_detection_image.py
```

Figura 38. . Exportar modelo de inferencia
(Elaboración propia, 2020)

Como se refleja en la figura 38, este comando crea los archivos de inferencia dentro del directorio */inference_graph que luego se utiliza para probar el modelo entrenado como se detalla en el siguiente apartado.*

5.4.9 Ejecución con datos reales

Para probar una imagen, se selecciona una, se la copia dentro del *file system* y se pasa la ruta de ella dentro del script *Object_detection_image.py.*
Una vez hecho esto, se ejecuta el siguiente comando, el cual da como *output* reconocimiento exitoso – fracasado de un producto en la imagen, pero también de manera gráfica como se detalla gráficamente más abajo.

```
python Object_detection_image.py
```

Las figuras 39,40 y 41 muestran algunos ejemplos.

Figura 39. Spaghetti canale 99% objeto detectado

Figura 40. Polenta presto pronta 750g 98% objeto detectado

Figura 41. Spaguetti matarazzo 98% objeto detectado

5.5 Dificultades Presentadas

5.5.1 Problemas al compilar los Protobuff:

Al ejecutar el comando para compilar los protos, este comando hace la conversión archivo por archivo, pero a veces había algunos archivos que quedaban sin su transpilación.
Luego al ejecutar el entrenamiento, éste daba errores en la compilación. Luego se ha observado este error en las primeras ejecuciones, y se percataron que faltaban archivos sin transpilar.
Se ha logrado replicar los procedimientos del comando previo y simplemente lograron direccionarlo a los archivos que faltaban transpilar.

5.5.2 Memoria insuficiente (Allocation of exceeds 10% of system memory).

Los autores han tenido este error antes de empezar el entrenamiento, luego han probado establecer una variable de entorno, ya que, investigando en internet han visto que en varios hilos se recomendaba, el siguiente comando:

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

Como este comando no resolvió el problema, han probado de bajar en *batch_size,* a 1, estaba en 3.
El *batchsize* es una hiper variable que decide cuántas imágenes, en este caso, van a pasar por las red a la vez por cada época, esto genera mayor consumo de recursos para lograrlo.

5.5.3 Errores de sintaxis

Los autores han sufrido varios errores de sintaxis, al no utilizar herramientas de *linteo* o validación de sintaxis antes de ejecutar los scripts, como errores en las referencias en los *paths* de configuración referenciando a archivos dentro del *framework.*
Especialmente en el *training/pipeline.config que han debido aplicar referencias al modelo pre entrenado y al dataset.*

```
fine_tune_checkpoint:
"C:/Users/agust/Documents/ia/models/research/object_detection/faster_rcnn_inception
_v2_coco_2018_01_28/model.ckpt"

train_input_reader: {
  tf_record_input_reader {
    input_path:
"C:/Users/agust/Documents/ia/models/research/object_detection/train.record"
  }
label_map_path:
"C:/Users/agust/Documents/ia/models/research/object_detection/training/labelmap.pbt
xt"
}
```

Figura 42. Configuración de rutas, con modelo pre entrenado y dataset
(Elaboración propia, 2020)

5.5.4 Inactividad del Tensorboard (Tensorboard no está activo)

Otra dificultad que se ha encontrado es que ejecutando el comando `tensorboard --logdir=training`, no se logra activar el servidor de TensorBoard que de esta manera supervisa así el entrenamiento.
El error que detalla la imagen 43 se ha detectado.

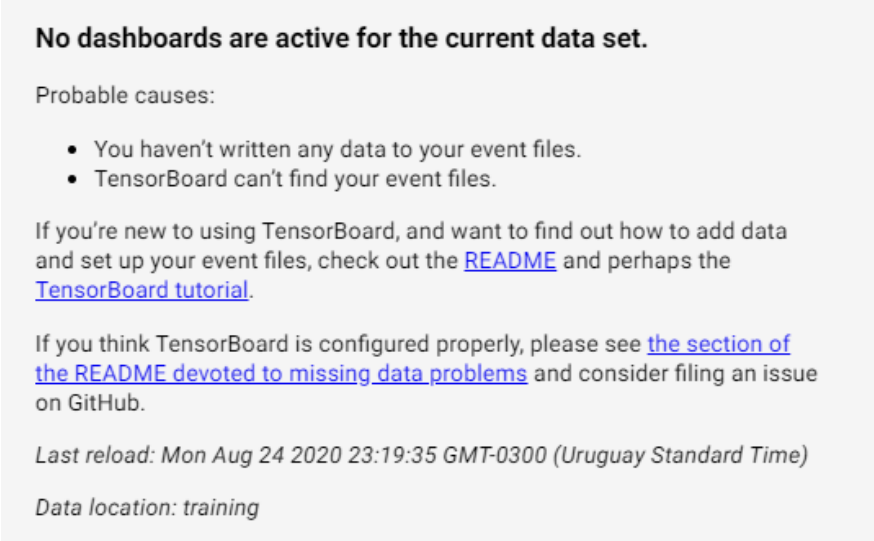

Figura 43. Error de tensorboard
Nota. Al ingresar al local host: 6006 (elaboración propia, 2020)

Los autores han identificado al problema, que fue que ha sido ejecutado el comando no dentro del directorio del framework*/object_detection*, sino desde cualquier otro lugar.
Tensorboard necesita "escuchar" archivos temporales para recolectar datos, ya que de otra manera no puede trabajar.

5.5.5 Problemas para obtener el *accuracy*

Resolver este problema ha llevado los tesistas más de un mes de trabajo, ya que no se había podido obtener el *accuracy* del modelo necesario.
Al investigar, se observó que se estaba ejecutando un script de entrenamiento deprecado (ver fig. 44), *train.py, que ya no* es más utilizado ya que no incluye ciertas estadísticas como el *accuracy.*
Al utilizar *model_main.py*. Se logró empezar a obtener los *accuracy* de los modelos.

Comando deprecado:

```
python train.py --logtostderr --train_dir=training/
--pipeline_config_path=training/faster_rcnn_inception_v2_pets.config
```

Comando actual:

```
python model_main.py -alsologtostderr --model_dir=training\
--pipeline_config_path=training/pipeline.config
```

Figura 44. Comandos
(Elaboración propia, 2020)

5.5.6 Versionar con excesivo tamaño

Se ha estado utilizando git como herramienta de control histórico de cambios, lograron guardar todos los cambios y el proceso de desarrollo de toda la investigación y entrenamiento de los modelos.
El problema que han tenido ha sido con el tamaño de los archivos, ya que existen muchos archivos que pesan más de 100MB lo cual *Git* esto no soporta.

La solución lograda es *Git Large File Storage,* lo cual ha permitido guardar archivos muy pesados sin tener que analizar línea a línea los cambios.
Esto es muy útil ya que para los archivos grandes que solo interesa su existencia y cuando se han creado, su contenido no es importante, ya que son archivos encriptados gigantes generados por compilación.

A lo largo de este capítulo los tesistas han comentado el desarrollo e investigación realizado en el período de tiempo que les ha insumido el trabajo.
Se han resaltado algunos obstáculos que se han tenido, a pesar de ello, han tenido satisfacciones de los resultados obtenidos con este framework de trabajo.

Se ha diagramado el proceso por el cual se ha podido reconocer un producto, y se ha enumerado los pasos que se han hecho en el capítulo.
Se considera que es una herramienta muy útil, ya que se podría extender el conjunto de datos (dataset) para reconocer centenares de productos, y también utilizar los puntos de control para guardar el progreso de cada entrenamiento.
Los siguientes pasos para el futuro apuntarían a llevar este servicio a un nodo *cloud* y exportarlo en una REST API para enviar imágenes y que se reciba la respuesta de reconocimiento exitoso o fallido, y de esa manera sería la implementación de esta tecnología.

Otro punto a destacar es que se ha hecho esta implementación solamente con CPU, el *framework* soporta GPU de manera de tener mayor poder de cómputo en caso de ser necesaria.

En el próximo capítulo se hablará del caso de estudio, relacionado con una propuesta para aplicar estos conocimientos de manera de agilizar los procesos manuales que actualmente manejan.

CAPITULO 6: Propuesta para el Banco Alimentario

6.1 Introducción:

En este capítulo se plantea la propuesta informática del caso de estudio, para automatizar el proceso de registro del ingreso de productos al Banco Alimentario de la ciudad de La Plata, con la utilización de *Machine Learning* y cuáles son las dificultades para llevarla a cabo.

Como posible solución a las problemáticas existentes en el proceso de ingreso de mercadería al sistema actual del Banco Alimentario, pensamos en un sistema coexistente que facilite el cargado de los productos ingresantes, utilizando inteligencia artificial para el reconocimiento de estos a través de imágenes. El sistema también ayudaría en las tareas de generar el remito para el donante y en la validación de los datos cargados.
Constaría con un sistema *cloud* con el algoritmo de reconocimiento de imágenes, el cual recibe imágenes de productos, los reconoce y retorna información referida a estos, como el código de producto, el nombre y su peso unitario. Además, debe contar con un dataset con aproximadamente 15 imágenes de los productos que habitualmente el Banco Alimentario recibe. Para esto no es necesario que estén cargadas desde el comienzo las fotografías de todos los productos, puede ser una tarea que se haga progresivamente hasta llegar a su totalidad.

6.2 Interfaces de UI

Como interfaz, pensamos en una aplicación para *smartphones*, donde los operarios puedan generar órdenes. Una orden es una donación, contiene un proveedor, la fecha de ingreso al sistema y los productos que conforman la donación, con sus cantidades y fecha de vencimiento.
Las órdenes deberán ser enviadas al sistema actual del Banco Alimentario de La Plata “Kolsen” para que las entradas sean cargadas allí.

A continuación, se explica a través de imágenes de ejemplo de la aplicación, cómo sería el proceso que deberían realizar los operarios para registrar una orden.

Como se ejemplifica en la figura 45, al ingresar a la aplicación *mobile*, se ven las órdenes ya registradas y un botón para agregar una orden.

Figura 45. Menú inicial de la aplicación mobile

(Elaboración propia, 2020)

La figura 46 representa a la ventana que se abre al hacer *tap* en el botón agregar orden. En esta ventana se pide seleccionar cuál es el proveedor o ingresar su nombre en caso de que no se encuentre. Debajo aparecerán, si es que hay, los productos ya agregados y los botones de agregar producto y de finalizar orden.

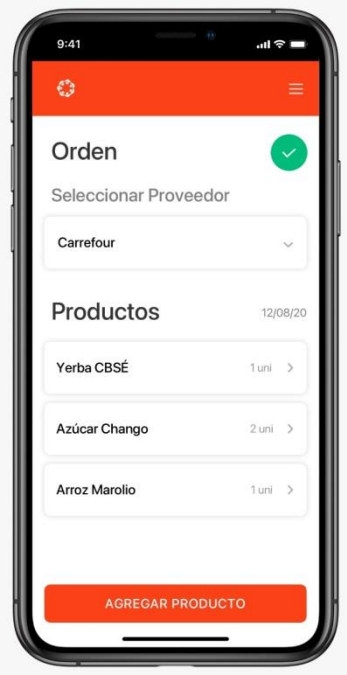

Figura 46. Ventana de tap
(Elaboración propia, 2020)

Al hacer *ta*p en el botón finalizar orden, se almacena la orden y es enviada al sistema Kolsen para que sea persistida.

Con pulsar *tap* en el botón agregar producto, se abre la cámara del móvil, para tomar una imagen al producto que se desea recuperar sus datos. Este paso está representado en la figura 47.

Figura 47. Incorporación del producto
(Elaboración propia, 2020)

Al capturar la imagen, es enviada al servidor *cloud* donde está alojado el sistema de reconocimiento de imágenes, donde es procesada y la respuesta es enviada de nuevo al móvil. Mientras que en la pantalla se informa que la imagen está siendo procesada, como se puede ver en la figura 48.

Figura 48. Aviso de imagen en proceso
(Elaboración propia, 2020)

Una vez que se recibe la respuesta, se despliega en una nueva pantalla los datos recuperados del producto, el código del producto, el nombre del producto y el peso unitario. Y hay dos campos más para completar manualmente ya que no se pueden recuperar a partir de la imagen: la cantidad y la fecha de vencimiento (ver figura 49).
Una vez completados los valores faltantes, el operario puede hacer *tap* en el botón siguiente para ingresar un nuevo producto o finalizar la orden (figura 46).

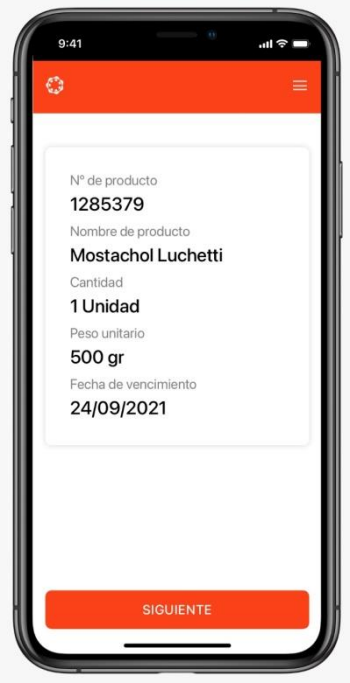

Figura 49. Información del producto
(Elaboración propia, 2020)

Al finalizar la orden, en una pantalla nueva se detalla la información de la misma (ver figura 50). A esta misma pantalla se puede acceder desde el menú principal donde se listan las órdenes existentes.

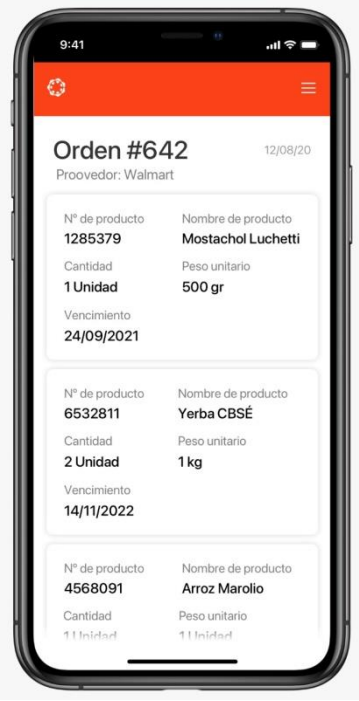

Figura 50. Información de orden
(Elaboración propia, 2020)

En el diagrama de la figura 51 se detalla la arquitectura propuesta para el sistema de reconocimiento de productos a través de imágenes.

6.3 Servicios en la Nube

Analizando el caso de estudio y entendiendo las problemáticas planteadas por el Banco Alimentario. A nivel arquitectura planteamos tener nuestro *framework* de *tensorflow* accesible como un servicio *cloud* en AWS o en alguna otra plataforma *cloud*, acompañado de un servicio REST para exponer estos servicios.

Del lado del cliente se cuenta con una aplicación *mobile*, encargada de capturar las fotografías y el resto de información de las órdenes, como por ejemplo el proveedor y la cantidad de unidades por producto.

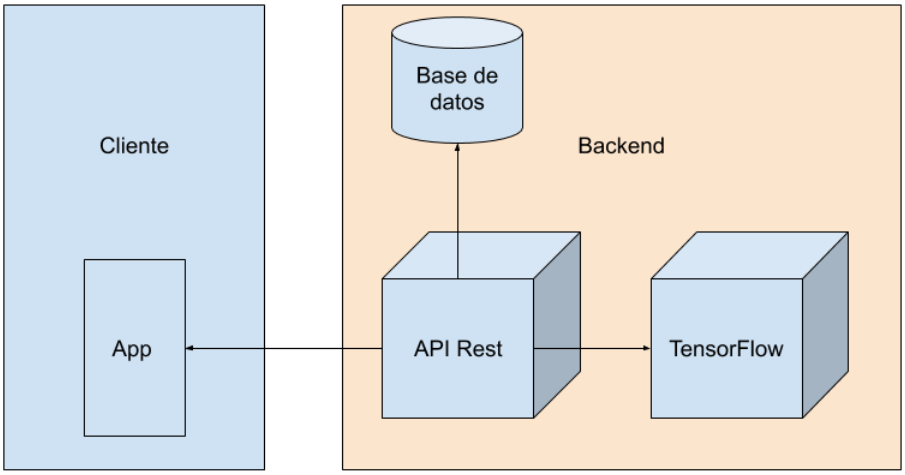

Figura 51. Diagrama de arquitectura
(Elaboración propia, 2020)

Del lado del *backend* se encuentra disponible la base de datos en la cual se persisten las órdenes detallando sus productos y proveedor.
También del lado del *backend* se encuentra el *framework* de *tensorflow*, con el cual se reconocen los productos de las imágenes que le son enviadas.
Además, como conector de todos estos elementos, se encuentra el servicio de API Rest, encargado de la comunicación con la aplicación *mobile*, recibiendo las imágenes y datos de las órdenes y devolviendo los datos una vez procesados. El servicio también se encarga de comunicarse con el *framework* de *tensorflow* para enviar las imágenes y recibir los datos recuperados de estas. Y se encarga de persistir en la base de datos las órdenes.

6.4 Dificultades

La principal dificultad que se nos presenta es la poca información que tenemos del sistema llamado Kolsen, utilizado por todos los Bancos Alimentarios del país. De este solo sabemos que es un sistema privativo, propiedad de la empresa Kolsen, que no existe la posibilidad de ser modificado, cuáles son los datos que se cargan por producto ingresado y una *screenshot* que nos proporcionó Elizalde (figura 52).

No sabemos si existe una forma en que podamos comunicar el sistema propuesto con Kolsen, como por ejemplo podría ser una API. Además, que no tenemos posibilidad de contactarnos con alguien de Kolsen para consultarle.
Además, hay que tener en cuenta que debería existir una persona, preferiblemente un ingeniero capacitado en inteligencia artificial, para dar mantenimiento, extender el dataset y entrenar el sistema para reconocer nuevos tipos de productos que el Banco Alimentario puede recibir.

Otra dificultad surge a partir de la pandemia Covid-19 que estamos atravesando y que nos imposibilito visitar el Banco Alimentario de La Plata para observar exactamente cómo sus operarios realizan el proceso de ingreso de la mercadería, ya que es posible que en las entrevistas, el director no nos haya contado, por ejemplo, de algún dato más que están registrando de los productos y que él no recuerde debido a que esa no es su tarea.
Otro problema que aparece es el de los datos que no pueden ser recuperados del producto a partir de las fotografías, como son la fecha de vencimiento, el proveedor y la cantidad. Esto hace que el proceso no pueda ser automático, y un operador deba completar los campos faltantes.
Además, en la entrevista nos contó que aproximadamente entre un 60% y un 70% de los productos recibidos están estandarizados, que siguen los patrones y pautas de la industria alimentaria. Es decir, hay un porcentaje considerable que no se pueden reconocer a través de imágenes, o al menos no se podrían recuperar todos sus datos, como son el caso de las frutas y verduras.

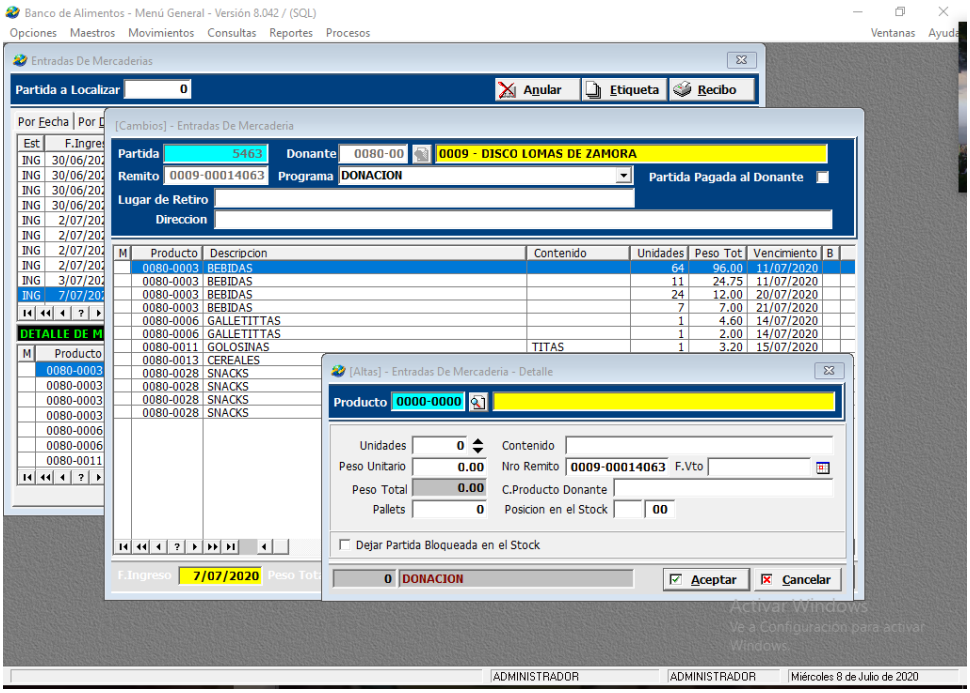

Figura 52. screenshot del sistema kolsen
(Elaboracion propia, 2020)

Conclusiones

A lo largo de este capítulo, y a partir de las entrevistas con su director, Pedro Elizalde, hemos desarrollado sobre qué es el Banco Alimentario de La Plata, cómo está formado, qué funciones cumple y cómo las lleva a cabo.

Se detalló con mayor profundidad cómo es actualmente el proceso de ingreso de mercadería y su persistencia en el sistema actual. A partir de esto analizamos cuáles son sus problemáticas.

Otro punto a destacar es el desarrollo de una propuesta, de cómo con ayuda de inteligencia artificial y reconocimiento a través de imágenes, se puede mejorar el proceso de ingreso de la mercadería al sistema actual del Banco Alimentario de La Plata.
Además, se explica y ejemplifica con imágenes, cómo sería el proceso de registrar el ingreso de mercadería con el sistema propuesto.

Por otro lado, se describen cuáles son las dificultades para llevar a cabo el sistema propuesto.

En el próximo capítulo hablaremos sobre las conclusiones a las que llegamos a partir de todo lo que hemos visto hasta aquí. Cuáles pueden ser trabajos futuros en base a este estudio y qué lecciones aprendimos durante esta investigación.

CAPÍTULO 7: CONCLUSIÓN FINAL

Hoy en día se puede producir, almacenar y enviar más datos que nunca antes en la historia, y la razón de ello se debe al aumento de la capacidad y accesibilidad de las tecnologías de la información, así como a la sustancial mejora de la calidad en los dispositivos de sensores, cámaras, micrófonos o cualquier otro que capture datos.

De hecho, se calcula que aproximadamente el 90% de los datos disponibles actualmente en el planeta se ha creado en los últimos dos años, todo ello siguiendo una tendencia fuertemente creciente.

En este escenario, se ponen en auge nuevas tecnologías, donde el principal objetivo es procesar datos y almacenarlos, dándoles otros significados o respuestas.
Es allí donde Machine Learning saca ventaja del gran volumen de datos y de variables, que le permite detectar patrones para resolver problemas, que antes eran técnicamente imposibles de resolver.

ML permite ofrecer una manera eficiente de capturar el conocimiento mediante la información contenida en los datos, y así de esta manera, mejorar de forma gradual el rendimiento de modelos predictivos y tomar decisiones basadas en dichos datos.
Es así que esas nuevas herramientas se han convertido en una tecnología con una amplia presencia, y actualmente se las encuentra en: filtros anti-spam para correo electrónico, conducción automática de vehículos o reconocimiento de voz, y el caso del enfoque de esta tesina es aplicado a la detección de productos en imágenes.

El objetivo principal de esta tesina ha sido el análisis de la técnica de *Transfer Learning* en Aprendizaje Automático a través de un caso de estudio, la Clasificación de Productos para el Banco Alimentario de La Plata.

En la aplicación tanto de ML y TL para el caso de estudio de Banco Alimentario se ha investigado en tecnologías actuales de vanguardia, herramientas de apoyo para el procesamiento de datos y sistemas de métricas para evaluar rendimientos.
Todo ello se ha logrado mediante la ejecución en etapas, como obtención de datos, validación de los datos, entrenamiento de las redes neuronales, obtención de resultados y comparación con otros modelos.

Se realizaron pruebas controlados de los modelos en las mismas condiciones, obteniendo resultados iníciales que parecen prometedores, sin perder de vista que necesitan realizarse mejoras como así también realizar mayor cantidad de pruebas para que los resultados sean concluyentes.

Esta condición ya ha sido explicada en detalle en el Capítulo III, donde se ha puesto a prueba tres modelos de distintas características, y RCNN ha obtenido casi un 95% en tasa de aserción, contra SSD inception y SSD mobileNet que no han llegado al 80%.
Si bien RCNN tardó más en detectar una imagen por tener una capa de detección de regiones sobre la imagen, esta dificultad no es grave para el enfoque de la tesina, ya que no afecta al objetivo de ella y por lo tanto no se asume como problema

También durante el desarrollo de esta tesina, y más precisamente en los capítulos de aplicación como Capitulo 4 y 5, se ha dejado reflejar que detrás de las tecnologías, existen arduos trabajos de mantenimiento para lograr que el sistema funcione de manera eficiente, esto conlleva costos tanto de tiempo, recolección de datos de cada producto y personal capacitado para llevar a cabo las tareas.

En los capítulos 2 y parte del 3 se ha explicado que se ha basado la investigación en recolectar información acerca de *transfer learning*, y los modelos pres entrenados disponibles.

Es sobre todo en este punto, que los tesistas han basado en tratar de encontrar similitudes en diferentes fuentes de información para validar la idea misma, ya que, en investigaciones académicas en inglés de solidez científica, muchas veces se ha tenido dificultades en la obtención de respuestas concretas.
Pero estos obstáculos han sido superados, ya que con tiempo y contrastando información desde diferentes fuentes se pudo seguir avanzando con el desarrollo para culminar los capítulos.

Los tesistas han tenido que realizar entrevistas e intercambio de comunicaciones por correo con el objetivo de recolectar información del Banco de alimento como posible caso de aplicación de estas tecnologías.
A nivel general y con un enfoque sistémico de cómo ha trabajado el Banco Alimentario, se ha obtenido un panorama extenso de ello, a pesar de la limitación que ha sido en parte la pandemia, que significó la imposibilidad de ir al Banco en persona y así entender mejor además de ver el sistema actual más allá de lo que detalla el representante, Señor Pedro Elizalde.

En conclusión, teniendo en cuenta los resultados obtenidos y las investigaciones realizadas, es posible afirmar que la aplicación de *machine learning* y *transfer learning* es factible de ser aplicada en el caso práctico propuesto, facilitando los procesos de entrada y salida de productos al sistema del Banco Alimentario. Aunque es de destacar que se necesita de una considerable inversión inicial para llevar a cabo la aplicación de este sistema y también que su mantenimiento es un proceso costoso.

7.1 Desarrollos Futuros

Si bien la propuesta presentada en esta tesina alcanza los objetivos acordes a la problemática a la que se buscaba dar solución en base a lo descrito en el Capítulo 2, el mismo puede tener varios puntos de extensión a futuro. Algunos de estos puntos son descriptos a continuación:

- Portal web:

Una aplicación en dispositivos móviles es práctica, accesible, no necesita hardware extra para llevar la tarea a cabo, pero es difícil para administrar grandes volúmenes de datos. Creemos que tener un portal web para la gestión y visualización más amplia de las órdenes y su detalle, y así poder editar y corregirlas, sería un gran avance para el sistema, y es un punto a tener en cuenta de cara al futuro.

- Servicio de reportes

Esto fue una idea que surgió durante las entrevistas con Pedro en su momento, incluir reportes diariamente usando mails, idea que ha sido de su agrado y así poder tener miles de resumen de las órdenes del día. para que las pueda leer de manera accesible, para todos los supervisores.

- Registro de entregas a comederos

Incorporar un registro de entregas a comederos en el sistema, integrado y accesible. Dando así seguimiento de las entregas.

- Sistema Analíticas

Es importante saber cómo está funcionando el Banco Alimentario y mostrar métricas, de productos, para ello se puede ingresar por períodos, clases de productos, vencimientos, etc. De esta manera se visualiza el estado actual del Banco y se facilita la toma de decisiones estratégicas.

- Exportar o importar los datos en otros formatos

Tener la posibilidad de exportar las gráficas y los datos de muestreo a planillas Excel, CSV, PDF. También podría permitir importar las planillas de Excel o CSV con un formato específico.

Estos son algunos trabajos futuros que se podrían realizar en diferentes líneas, ya sea mejorando el software como la metodología de carga de cada ingreso o egreso, tendiendo a que el pasante o investigador pueda contar con mayor realismo, y ayudarlo así a una forma rápida y eficiente en particular de la carga de productos.

BIBLIOGRAFÍA

Krizhevsky, Sutskever y Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks*. Neural Information Processing Systems*. 25 (2). Recuperado de: http://www.cs.toronto.edu/~kriz/imagenet_classification_with_deep_convolutional.pdf

He, K., Zhang X., Ren, S., y Sun., J. (2015). Deep Residual Learning for Image Recognition. Microsoft Research. Recuperado de: https://arxiv.org/pdf/1512.03385.pdf
Redes Neuronales de clases múltiples: sofmax. *Aprendizaje automático*. Recuperado de: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax?hl=es#:~:text=Softmax%20lleva%20esta%20idea%20al,probabilidades%20decimales%20deben%20sumar%201.0.

Shi Yiming. TensorFlow 1 Detection Model Zoo. Recuperado de: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md (FECHA ÚLTIMO INGRESO: 7/01/2021)
Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár (2015). Microsoft COCO: Common Objects in Context. Recuperado de: https://cocodataset.org/#home

Wei Liu, Dragomir Anguelov , Dumitru Erhan , Christian Szegedy , Scott Reed4, Cheng-Yang Fu, Alexander C. Berg. (2016). SSD: Single Shot MultiBox Detector. Recuperado de: https://arxiv.org/pdf/1512.02325.pdf

Sarkar,D (2018). A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning. Deep Learning on Steroids with the Power of Knowledge Transfer!.*Towards data science*. Recuperado de:https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Goodfellow I., BengioY.,y Courville A. (2015). Deep Learning, (Adaptive Computation and Machine Learning series), Recuperado de: https://www.deeplearningbook.org/

Cortes,C, Jackel, L. D. y Chiang, Wan-Ping (1994). “Limits on Learning Machine Accuracy Imposed by Data Quality”. *AT&T Bell Laboratories*. 239-246

SinnoJialin Pan and Qiang Yang (2010) “A Survey on Transfer Learning*”. IEEE Transactions on Knowledge and Data Engineering*, (22)10. 1345-1359

Torrey, L y Shavlik J, (2009). Transfer Learning. En Torrey, L y Shavlik J (Information Science Reference). *Machine Learning Applications and Trends: Algorithms, Methods, and*

Techniques. p.834. Universityof Wisconsin, USA. (FECHA DE ÚLTIMO INGRESO: 2/12/2020).

Atul Pandey (2018, 19 de Septiembre). “Depthwise Convolution and Depth-wise Separable Convolution”

Repositorio: TensorFlowObjectDetectionModel.
Recuperado de: https://github.com/tensorflow/models/tree/master/research/object_detection
(FECHA DE ÚLTIMO INGRESO: 2/12/2020)

Howard, Andrew & Zhu, Menglong& Chen, Bo &Kalenichenko, Dmitry& Wang, Weijun&Weyand, Tobias&Andreetto, Marco & Adam, Hartwig. (2017). “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. p 1-9 (FECHA DE ÚLTIMO INGRESO: 2/12/2020). Recuperado de: https://arxiv.org/pdf/1704.04861.pdf

Girshick, Ross (2015). “Fast R-CNN”. Microsoft Research. p. 1-9 Recuperado de: https://arxiv.org/pdf/1504.08083.pdf (FECHA DE ÚLTIMO INGRESO: 2/12/2020)

Wei Liu, Anguelov, D; Erhan D; Szegedy C; Reed S; Cheng-Yang Fu; Berg, C. (2016). “SSD: SingleShotMultiBox Detector”. Recuperado de: https://arxiv.org/pdf/1512.02325.pdf
(FECHA DE ÚLTIMO INGRESO: 2/12/2020)

Paul-Louis Pröve. (2017). “An Introduction to different Types of Convolutions in Deep Learning”. Toward data science. Recuperado de: https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d
(FECHA DE ÚLTIMO INGRESO: 2/12/2020)

Jonathan Hui (2018, 6 de Marzo). “mAP (mean Average Precision) for Object Detection”. Recuperado de:https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173
(FECHA DE ÚLTIMO INGRESO: 2/12/2020)

Atul Pandey (2018, 19 de Septiembre). “Depthwise Convolution and Depth-wise Separable Convolution”. Recuperado de: https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec
(FECHA DE ÚLTIMO INGRESO: 2/12/2020)

Model Zoo - Deep learning code and pretrained models for transfer learning, educational purposes, and more. Recuperado de: https://modelzoo.co/
(FECHA ÚLTIMO INGRESO: 25/06/2020)

Russell, S y Norvig P. (2009) *Artificial Intelligence: A Modern Approach.* Prentice-Hal.p. 709. USA

Kuhn M., y Johnson, K. (2013). *Applied Predictive Modeling*. Springer. p. 67, USA

Tensorflow Object Detection Api Tutorial Readthedocs. Installation. Recuperado de: https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html#general-remarks (FECHA DE ÚLTIMO INGRESO: 14/09/2020)

TensorFlow Image Recognition with Object Detection API: Tutorials. Recuperado de: https://missinglink.ai/guides/tensorflow/tensorflow-image-recognition-object-detection-api-two-quick-tutorials/ (FECHA DE ÚLTIMO INGRESO: 14/09/2020)

Setup TensorFlow for Object Detection on Ubuntu 16.04 (Part 1). Recuperado de: https://medium.com/@teyou21/setup-tensorflow-for-object-detection-on-ubuntu-16-04-e2485b52e32a
(FECHA ÚLTIMO INGRESO: 3/12/2020)

Training your Object Detection model on TensorFlow (Part 2). Recuperado de: https://medium.com/@teyou21/training-your-object-detection-model-on-tensorflow-part-2-e9e12714bdf
(FECHA ÚLTIMO INGRESO: 3/12/2020)

Ambika Choudhury (2019, Junio, 28) TensorFlow vs Keras: Which One Should You Choose. Recuperado de: https://analyticsindiamag.com/tensorflow-vs-keras-which-one-should-you-choose/#:~:text=Keras%20is%20a%20neural%20network,provides%20only%20high%2Dlevel%20APIs.&text=Keras%20is%20built%20in%20Python,more%20user%2Dfriendly%20than%20TensorFlow. (FECHA ÚLTIMO INGRESO: 3/12/2020)

14 Librerías de Python para ML. Recuperado de: https://www.iartificial.net/librerias-de-python-para-machine-learning/#Librerias_de_Python_para_Deep_Learning
(FECHA ÚLTIMO INGRESO: 3/12/2020)
Anaconda | The World's Most Popular Data Science Platform. Recuperado de: https://www.anaconda.com/
Krebs, Bruno (2017, Enero 31). Beating JSON performance with Protobuf. Recuperado de: https://auth0.com/blog/beating-json-performance-with-protobuf/
(FECHA ÚLTIMO INGRESO: 25/06/2020)

Gamauf, Thomas. (2018, Marzo 20). Tensorflow Records? What they are and how to use them. Recuperado de: https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564#:~:text=A%20TFRecord%20file%20stores%20your,train.
(FECHA ÚLTIMO INGRESO: 3/12/2020)
TowardDataScience.Kunkun Bai (2019, 11 Feb). A Comprehensive Introduction to different types of convolutions in deep learning. Towards intuitive understanding of convolutions through visualizations.

Szegedy, Wei Liu, Yangqing, Jia, Sermanet, Scott Reed, Anguelov, Erhan, Vanhoucke, Rabinovich (2014). “Going deeper with convolutions”. p 1-12. Cs.CV. Recuperado de: https://arxiv.org/pdf/1409.4842v1.pdf
(FECHA ÚLTIMO INGRESO: 12/12/2020)

Schlegel, D. R y Shapiro, S. C. (2013).“InferenceGraphs: A Roadmap”. *Second Annual Conference on Advances in Cognitive Systems*, University at Buffalo, USA. p.217-234

Freeze Tensorflow models and serve on web – CV-Tricks.com
Recuperado de: https://cv-tricks.com/how-to/freeze-tensorflow-models/
(FECHA ÚLTIMO INGRESO: 3/12/2020)
Aryan Mobiny (2018, Junio 9). How to Use TensorBoard? The two main advantages of TensorFlow
Recuperado de: https://itnext.io/how-to-use-tensorboard-5d82f8654496
(FECHA ÚLTIMO INGRESO: 3/12/2020)
Visualization with TensorBoard. Recuperado de: https://databricks.com/tensorflow/visualisation
(FECHA ÚLTIMO INGRESO: 3/12/2020)

Repositorio: LabelImg. Recuperado de: https://github.com/tzutalin/labelImg
(FECHA ÚLTIMO INGRESO: 14/09/2020)

Tensorflow (2018, Noviembre 15). Object detection time increases as the image resolution increase. Recuperado de: https://github.com/tensorflow/tensorflow/issues/23782
(FECHA ÚLTIMO INGRESO: 14/09/2020)

Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90. doi:10.1145/3065386

ANEXOS 1: Entrevista al Sr Director del Banco Alimentario, La Plata (Buenos Aires)

En este anexo se presentan las preguntas realizadas al Cdor. Pedro Elizalde en distintas entrevistas. Se ha transcripto el total de la entrevista para poder comprender mejor el contexto de las respuestas y los detalles relevantes para el objetivo de esta tesina.

❖¿Qué es el Banco Alimentario?

"Primero de todo nuestra forma jurídica es una sociedad civil sin fines de lucro, esto que quiere decir que somos una organización reconocida por persona jurídica. Trabajamos todo de acuerdo a lo que son las normativas vigentes, tenemos 20 años de existencia y nuestro modelo es un modelo que se basa en experiencias internacionales."

❖ ¿Qué es el modelo?

"El modelo es poner en valor alimentos, valor social, valor económico, valor alimentario. Alimentos que estando en perfectas condiciones, con respecto a todo lo que son sus valores nutricionales, cumplimiento con los códigos alimentarios por razones diversas que son muchas veces comerciales, estos alimentos son sacados del circuito de comercialización y también se da en otro aspecto que tiene que ver con la producción, por ejemplo, la producción agrícola, la producción agropecuaria, que muchas veces cosechas o determinados productos vegetales que no son llevados a mercado por condiciones que tienen que ver más bien con lo económico pero no con lo alimentario. Entonces nuestra acción está orientada justamente a recuperar esos alimentos y poner esos alimentos a disposición de los sectores vulnerables, de aquellos que por alguna razón no pueden acceder a una canasta básica de alimentos. Este sería en líneas generales el modelo del Banco Alimentario.
El nuestro está fundado en noviembre del año 2000, con lo cual la trayectoria son 20 años, tratamos de tener en todo lo que hacemos protocolos, reglas que nos permiten ya que estamos trabajando en un sector tan delicado como es la alimentación, que nos permitan ser muy estrictos en todos nuestros procesos. Nuestro equipo de trabajo está muy profesionalizado en lo que es la función que estamos haciendo y se cumplen estrictamente todas estas normas, todos estos protocolos, inclusive tenemos auditorías con respecto a cómo estamos cumplimentando todo lo que sería los aspectos de gestión nuestra. En argentina no hay un solo banco, en este momento hay 14 bancos operativos en distintos lugares y nos integramos en algo que llamamos una red. Sería una organización de segundo grado, por encima de los bancos está esta red con los fines de coordinar con fines de mejorar los desempeños, conseguir de alguna forma una mayor cantidad de donantes porque nosotros tenemos donantes de todo tipo y fundamentalmente de la gran industria alimentaria. Te pongo casos de empresas, a nosotros nos dona Danone, Coca

Cola, Mc Donals, Molinos, osea tenemos los grandes donantes a partir de la industria y todo de esta manera está concentrado a través de esta red de bancos. A su vez, formamos parte de una red mundial, la Global FoodBanking, que quiere decir, que todo lo que es el accionar nuestro, está de una manera o de otra integrado y protocolizado de acuerdo a normas de buenas prácticas de manufactura, normas que tienen que ver con lo legal en los alimentos, este sería el trabajo nuestro, conseguir esas donaciones, estas donaciones serán la entrada a nuestro sistema que lo que nosotros tenemos que hacer es tener una logística, una capacidad de entrega de esta mercadería con una muy buena dirigencia, porque muchas veces son mercaderías que están con una ventana de vencimiento muy corta que hace que nosotros tengamos que ser muy ágiles para entregarlas, que quiere decir, de los donantes ingresar la mercadería, tenemos un depósito, en ese depósito clasificamos y lo que hacemos es de un banco de datos de comedores que son instituciones que dan servicio alimentario, nosotros vamos entregando esa mercadería de acuerdo a lo que es la composición de los servicios que dan cada uno de estos comedores, algunos están dando almuerzo, otros están dando cenas, otros son solamente merenderos. Entonces lo que tenemos es un registro que está muy detallado de cada uno de estos comedores, no solo el servicio que dan sino la población a la que atienden. Lo que hacemos es enlazar la mercadería que disponemos con las necesidades de los comedores, en ningún caso nosotros trabajamos con los pedidos de los comedores, porque lo que tenemos disponible viene de alguna manera aleatoriamente de acuerdo a lo que nos puede donar la industria. Para poner un ejemplo, podemos llegar a tener en un momento una cierta cantidad de yogures que nos dona Danone, esos yogures los tenemos que colocar y a lo mejor no estamos entregando en ese momento fideos, porque no los tenemos, entonces no actuamos como un supermercado que uno elige. Lo nuestro es tener un claro conocimiento de la mercaderia que tenemos en stock disponible a partir del ingreso y enlazarla a esa mercadería de acuerdo a estrategias de vencimiento, a estrategias de necesidad de los comedores, con nuestros comedores. Esto sería el modelo global.”

❖ **¿Cómo es el proceso de ingreso de los alimentos?**

“Como ejemplo vamos a poner un donante en particular que podemos tener y que los podemos manejar. Danone puede tener partidas que sabe que no va a comercializar de agua saborizada, puede tener yogures, puede tener distintos artículos. Ellos coordinan previamente con nosotros cuál es la mercadería que nos va a donar, agua saborizadas. Entonces a un determinado momento se concreta el tema logístico de la llegada de la mercadería al Banco, agua saborizada que puede ser de distintos tipos de sabores, distintos tipos de formatos, todo aquello que hace a la distinción de los artículos que forman parte de lo que sería la alimentación. Coordinado la fecha en que nos llega, esto nos va a llegar en un camión a nosotros.“

❖ **¿Cómo llega la mercadería?**

“Comúnmente por los volúmenes que manejamos llegan pallets y cada uno de los pallets, dependiendo el tipo de artículo, trae más o menos kilos, más o menos artículos, hay distintos tipos de paletización, y eso es lo que se descarga del camión. Al momento del ingreso, que se hace manualmente, se tiene que registrar el pallet, los artículos que

contiene, fecha de vencimiento, todos aquellos indicadores que a nosotros nos permite estimar la mercadería, saber que stock contamos de cada mercadería, quién es el donante, y a partir de ese momento queda ingresada en nuestro sistema informático y queda ingresada físicamente en el depósito."

❖ **¿Cómo es el proceso de salida de los alimentos?**

"Tenemos variedad de proveedores, variedad de artículos, entonces todo esto queda registrado para que nosotros podamos tener una estrategia de ir entregando a los comedores, que también están dentro de un sistema informático la cantidad de comedores que tenemos, y entregarles de acuerdo a la cantidad de gente que a ellos asisten. Entonces la salida sería primero el encuentro entre lo que nosotros tenemos de oferta dentro de nuestros almacenes y el encuentro de las necesidades que tienen cada uno de los comedores. A partir de eso, automáticamente vamos generando lo que nosotros llamamos un pedido, que sería comedor por comedor, de acuerdo a la mercaderia que tenemos y a lo que ellos necesitan, un detalle donde decimos van 200 kg. de fideos, van aguas saborizadas por tal cantidad. Todo eso queda documentado en lo que después se va a transformar en una especie de remito. Eso lo que nos permite a nosotros es con anticipación ir preparando ya el pedido físico, o sea generamos a través del sistema lo que sería lo que va a tener cada comedor y después físicamente con eso vamos armando con una hoja, comedor por comedor se va armando lo que sería el pedido físico, o los artículos físicamente se embalan y al día siguiente son retirados por cada uno de los comedores."

❖ **¿Cómo resuelven el tema de la fecha de vencimiento?**

"Tomemos el ejemplo del yogurt. Todos sabemos que el yogurt tiene una fecha de vencimiento y a nosotros normalmente lo que nos llega, porque muchas veces recuperamos en los supermercados también, son los productos de corto vencimiento, que el supermercado sabe que no va a vender. Entonces directamente antes de mandarlo a destrucción, eso nos los dona a nosotros, caso Walmart, caso la cadena Jumbo, que trabajamos de esa manera, qué quiere decir, que a lo mejor estamos trayendo de una de las tiendas de Walmart o de Jumbo yogures que va a vencer en 48 horas. Ahí automáticamente los ingresamos y automáticamente estamos haciendo por la corta ventana de vencimiento la distinción de los comedores que por ejemplo, en muchos casos, nos centramos en merenderos, copa de leche. O sea el vencimiento es el factor determinante para hacer rápidamente el encuentro entre lo que tenemos y lo que tenemos que entregar."

❖**¿Cómo se manejan con las frutas y verduras?**

"Acá nosotros hacemos retiros de frutas y verduras en el mercado regional de La Plata y también vamos al cordónhortícola que está en la zona de Olmos. Ahí lo que nosotros hacemos es, tenemos acuerdos con productores hortícolas, fundamentalmente es tomate y otros tipos de verduras, y vamos a retirar. Lo que retiramos, automáticamente, como los yogures tienen que ser una entrega muy rápida. Si bien nosotros tenemos cámaras frías, por ejemplo vegetal los podemos mantener bastante tiempo. Lo que hacemos es que las

entregas sean muy rápidas, siempre de acuerdo a la necesidad que tiene cada comedor, y tratar de ser equitativo, porque no tenemos la cantidad para abastecer al 100% los requerimientos que tienen cada uno de los comedores. Hay comedores que tiene 100 personas, otros más de 100, entonces lo que tratamos de buscar es una equidad de que el stock que tenemos llevarlo, pero todo lo que tenga que ver con vencimiento de productos frescos como la fruta y las verduras requieren que el encuentro entre la llegada del producto, el ingreso al sistema nuestro y la entrega sea lo más rápido posible.
El sistema es todo único, lo que hace es tener la funcionalidad de incorporar los registros de entrega, tener incorporada la base de comedores y la funcionalidad es sencillamente la de hacer el encuentro entre necesidades y lo que tenemos."

❖ **Cuando te réferis a sistema, ¿Te réferis a sistema informático?**

"Si, sistema informático, es un sistema informático que está diseñado para todos los bancos que formamos parte de la red, porque aparte de ahí salen estadísticas sale información que hace de cada uno de los bancos y yo les comente que tenemos auditorías, el sistema brinda también información para la auditoría."

❖**Acerca de las auditorias, ¿Son esporádicas?, ¿En qué parámetros se basan?**

Primero de todo, les conté que hay normas, hay protocolos, hay buenas prácticas de manufacturas que están todas redactadas. Y lo que tienen que tener es que lo que se tiene escrito se tiene que cumplir en la realidad. Y la auditoría lo que busca es chequear eso.
En estos momentos que estamos trabajando en una situación muy límite, muy acotada por la pandemia, estamos con un depósito que no está cumpliendo las prácticas que tendría que tener la auditoría con a cuanto el almacenamiento por la falta de personal.
Nosotros trabajamos con personal estable y con voluntarios, que son de las mismas organizaciones, y en estos momentos por el aislamiento buscamos de que no trabajen voluntarios, con lo cual hoy día con las normas, los bancos que formamos la red estamos un poco fuera de las normas en cuanto el almacenamiento de mercadería, porque la guardamos en distintos lugares de acuerdo a que sean alimentos más perecederos, que sean alimentos que no pueden estar muy en contacto con otros productos, y hoy en dia esas normas están siendo violadas por la situación."

❖ **Acerca del personal, ¿Cuánta gente trabaja, qué tipo de personal es?**

"Nosotros tenemos un personal estable que está integrado por un director ejecutivo, que es un futuro ingeniero industrial, después tenemos un área de administración con una ingeniera industrial, después tenemos un área de logística otra ingeniera industrial, después tenemos un área que llamamos el área social que es la que se encarga de todo lo que tenga que ver con el contacto con las organizaciones sociales, las incorpora, las va siguiendo, también hacemos visitas y se encarga de toda la relación con respecto a los comedores. Después tenemos un área de comunicación donde hay una licenciada en comunicación social, después tenemos un jefe de depósito que es un muchacho técnico en eso y después tenemos un chofer encargado de uno de los vehículos del Banco y un ayudante. Este es el personal estable del Banco, pero nosotros movemos por mes más o

menos 50 a 60 mil kilos de mercadería, tenemos un depósito con una dimensión de 20 metros por 60 metros, con lo cual son unos cuantos metros cuadrados que tenemos. Tenemos una planta de procesamiento de vegetales que ahí hacemos puré de tomate, hacemos corte de hortalizas para hacer bandejas de sopa, que son mix de vegetales, y todo esto no lo podemos manejar con esta gente, entonces cómo se trata de una sociedad civil con alto compromiso social y solidario, los mismos comedores y hay gente que son de las facultades, tenemos convenio con la Universidad de La Plata, con la facultad de económicas, entonces tenemos mucha gente que viene a colaborar con nosotros y esos serian los voluntarios y lo que le dan al modelo mucha potencia, porque trabajamos también en un ambiente de alto compromiso social."

❖ **¿Ustedes chequean producto a producto la fecha de vencimiento de los pallets o confían en que la empresa le diga este pallet tiene esta fecha de vencimiento?**

"Volviendo al tema del ingreso, nosotros chequeamos que cada uno de los productos tenga una fecha de vencimiento determinada, porque la empresa nos puede mandar en los remitos distintos productos con distintas fechas de vencimiento. Lo que nosotros tenemos que ser muy estrictos es, si viene unapaleta, en el pallet viene todo con la misma fecha, eso se hace así. Sacar mercadería, controlar la fecha de vencimiento, eso sería una parte de registro, hoy día eso lo estamos haciendo manualmente nosotros. Ahí es donde yo creo que si ustedes pueden, a ver hay productos que vienen ya con código, la mayoría. Si nosotros pudiéramos capturar en ese momento que se está bajando la mercadería e ingresarlo al sistema, el artículo, el proveedor, la fecha de vencimiento. Eso sería para nosotros algo muy interesante y que nos daría mucha más eficiencia."

❖ **Para cargar los productos al sistema, ¿Usan pistolas lectoras de código de barra o se cargan manualmente?**

"No, la mayoría de las cosas se cargan a mano."

❖ **Pero ¿Utilizan el código de barra?**

"El código de barra se está utilizando muy poco, casi todo está cargado a mano. Osea, supongamos que llega el camión de Danone, vienen tantos palets, se miran los palets. Hay una planilla en donde se va anotando y después todo lo cargamos al sistema. Vas anotando el artículo, el peso que tiene, la fecha de vencimiento, tipo de producto, etc."

❖ **Te cuento un poco lo que estamos haciendo nosotros, estamos trabajando con inteligencia artificial, tenemos un algoritmo de reconocimiento de productos para reconocerlos a través de una foto. Esto puede ser llevado a un dispositivo que capte imágenes, lo lleve a un servicio y reconozca que producto es.**

O sea que directamente vos a través de una foto estarías teniendo la capacidad de reconocer el producto y ¿Reconocer otras variables del producto también?

❖ **En principio solamente identificar el producto. Por eso una de las preguntas también es ¿Los pallets son homogéneos o hay pallets que son heterogéneos?**

No, todo lo que viene paletizado son productos homogéneos, es el mismo artículo que está guardado por lo que llamamos pallets.

❖ **Lo que se podría hacer es una vez identificado qué producto es, ya sea un ejemplo, pallets de bolsas de harina o simplemente bolsas de harina individuales. Esos dos tipos de productos se podrían contabilizar directamente. Obviamente el peso, ustedes el tema peso, ¿Tienen balanzas?**

No, a ver, hay algunos que se pesan pero la mayoría de los pallets ya vienen con el peso determinado. Por ejemplo, un pallet de galletitas tiene tantas unidades de paquetes de galletitas y multiplicas eso por su peso y te da. Y ya viene determinado el peso.
¿Ustedes no tienen ninguna forma de determinar características del producto?

❖ **Eso a través de la foto es prácticamente imposible porque nosotros detectamos patrones en los productos. Nosotros lo hicimos con polentas y tipos de fideos, entonces nosotros detectamos Matarazzo, la forma de la marca Matarazzo, el color rojo, contra Lucchetti que es azul, la marca Lucchetti y eso. Y en el tema de fecha de vencimiento vienen en distintos lugares con distintos formatos, entonces fecha de vencimientos es prácticamente imposible. Eso tiene que ser puesto de alguna manera manual en alguna etapa del proceso de ingreso.**

¿El peso si lo pueden capturar, si es de 500 gramos, si es de 1000 gramos?

❖ **Si es individual, o sea si me das un producto que es de una polenta es envasada, o sea si esa polenta con esa marca, con esa forma siempre es de 500 gramos, obtenemos el peso. Pero si es un pallet que un dia viene con 400 polentas y otro dia viene con 500 polentas, necesitamos que alguien ponga la cantidad.**

¿Qué posibilidades tiene esto que me hablabas, fideos Lucchetti, fideos Matarazzo, de ir reconociendo un universo de productos mucho más grande? Acá estamos llegando me parece a una interrelación de artículos muy grandes y vos todo lo tenes que modelizar a través de las fotos que sacas para poder recién interpretar que tipo de producto es y qué marca es. Osea vos tenes que ir uno a uno para poder llevar a tener de alguna forma o de otra un compendio de productos o artículos que nosotros manejamos muchísimos.

❖ **Si, necesitamos fotos de cada producto que ustedes manejan y en base a eso se van agregando. De cada producto se necesitan alrededor de 15 fotos y se van agregando de manera esporádica. No tiene que ser totalmente de una sola vez, se puede ir agregando.**

Y esa foto lo que estaría reconociendo marca, producto y llegado el caso ¿Podría reconocer el peso?

❖ **Si es un producto envasado, por ejemplo los fideos Luchetti de un mismo modelo que viene de 500 gramos y de un kilo, si lo podemos reconocer.**

Entiendo, y todo ese universo de fotos para poder detectar el universo de artículos que tenemos nosotros ¿Dónde queda residente?

❖ **Bueno eso se puede manejar de varias maneras. Lo puede manejar alguien en una computadora, en un disco duro, que se va almacenando o se puede subir a un servidor cloud como backup. Ese conjunto de imágenes se llama dataset, conjunto de datos, eso va creciendo. No tiene que arrancar directamente con 5000 imágenes. Comenzamos con 50 productos, después agregas 50 mas, después agregas 100, después agregas 400. Es un desarrollo y es prueba y error. Cuando un producto funciona mal, hay que ver que foto están mal hechas y corregirlo. Pero es muy potente, porque si ustedes pueden identificar qué porcentaje de todo su ingreso es estandarizado, la verdad que tal vez con esto podría ser un principio de un proceso automatizado.**

"Si, yo creo que de lo que nosotros manejamos hay entre un 60% y 70% que es estandarizado. Sigue patrones, sigue pautas de la industria alimentaria, por lo cual es estandarizado. Cuando entramos al mundo, que hoy me preguntaron, de fruta, de verduras y esas cosas, ahí ya entras en otra cuestión. Pero bueno si hay una herramienta que permita que el 60% de nuestra operación sea reconocida mucho más directa de lo que hacemos ahora, estamos en un punto muchísimo mejor."

❖ **¿Qué datos se registran de cada producto?**

"Son varios los datos, ahí es donde creo que tenemos la gran dificultad. La captura de datos apuntá primero quien es el proveedor, ósea puede ser un supermercado, puede ser un particular, puede ser una gran empresa, este es el dato que de alguna forma nos permite a nosotros tener bien identificado la procedencia del producto. El segundo dato que tenemos, que tipo de artículo es, ahí lo tenemos codificado a los artículos que son. Ustedes me habían dicho que habían trabajado en lo que eran fideos, habían visto dos tipos de fideos. Bueno, fideos.
Tercer dato que tomamos, el peso que tiene el artículo, yo si quieren les mando un registro, pero a ver, acá esta mi gran duda. Vamos a la captura de datos y al método que ustedes utilizan, yo no lo conozco, pero la consistencia de datos para nosotros es fundamental, o sea que tiene que tener una validación firme de lo que estamos ingresando al sistema y me queda un poco la duda, si bien no conozco cómo es, ustedes me dicen que a través de fotos ustedes consiguen llegar a una determinada identificación de caracteres o de variables, ¿es así no?"

❖ **Si, más que nada reconoce productos por la forma del producto, los colores, los patrones. Después les puedo mostrar una imagen de cómo funciona, pero sí más que nada así, no es 100% exacto, un buen número es arriba del 90, 95%.**

"Ahí ya tenemos un problema, porque a partir de un dato que tenga un 5% de error nos genera a nosotros una incongruencia, o sea ustedes saben que la captura del dato es donde explota el sistema todas las posibilidades y si tenemos un dato que tiene una probabilidad de error ya estaríamos en problemas. Si bien la captura de datos hoy en dia es humana, que hay siempre porcentaje de error, pero de alguna forma o de otra ya está bastante más acotada.
De los patrones que ustedes toman, para nosotros es muy importante poder identificar un proveedor, no sécomo el sistema de ustedes, o la captura que ustedes prevén puede llegar a capturar de que es un supermercado, que supermercado es. Ahí creo que entramos a un mundo de dificultades."

❖ **Salvo que no haya dos proveedores que entreguen el mismo producto.**

"Puede haber dos proveedores que entreguen el mismo producto. Ahí es donde tenemos los problemas, yo creo que la dificultad grande está en que es un método en el cual deja abierta muchas posibilidades a lo aleatorio."

❖ **¿Ustedes la mayoría de las cosas las trabajan con pallets?**

"Trabajamos con pallets, una gran mayoría con pallets, pero no quiero cerrarle la puerta a lo que puede ser un recupero que hacemos en un supermercado donde, por ejemplo, Walmart nos entrega artículos variados y quizás no son pallets. Pallets tenemos una gran mayoría, podría ser un camino los pallets, osea tomar que la captura de ustedes apunta a única y exclusivamente a los pallets."

❖ **¿Podrías decirnos un porcentaje de pallets que manejan sobre el total?**

Depende sobre el total que digas. Si es sobre el total de kilos es el mayoritario, pero si vamos al total de artículo que tenemos, pallets no es el mayoritario. Hay dos mundos, un mundo es kilos en total que almacenamos y otro mundo tiene que ver con la variedad de artículos que vienen, en la variedad ya pallets no es mayoritario.

❖ **En el registro ¿Almacenan cantidades de cada producto?**

"Cada producto forma un registro. Suponte, productos serían, vamos a decir fideos, vos podes tener fideos tirabuzón, podes tener fideos moñito, ahí se empieza a abrir todo. Entonces fideos, tendría que estar el tipo de fideo que es, arrancamos por el proveedor, supongamos que viene todo del mismo proveedor. Tenemos un proveedor determinado que puede ser Molinos, Molinos nos manda a nosotros fideos, esos fideos pueden ser fideo tipo moñito, es un registro ya, eso va a armar dentro de nuestro sistema un archivo determinado, que ese archivo va a tener ese tipo de fideo y tiene que tener el peso que tiene y la fecha de vencimiento que tiene. Y después podemos tener otra variedad de fideos que puede ser tirabuzón que también los manda Molinos, el peso que puede ser 500 gramos, puede ser 1000 gramos depende de la variedad que sea y también su fecha de vencimiento."

❖ **Estos fideos son fideos conocidos, ¿Son de la marca Molinos que los podes encontrar en el supermercado?**

"Si, los podes encontrar en supermercado y puede haber algunos que son de otra variedad, Marolio por ejemplo."

❖ **Mi pregunta iba más si existe la posibilidad de que Molinos les mande una bolsa grande de fideos sin envasar.**

"Pasa también, por ejemplo recibo galletitas, Granix nos manda galletitas Granix en bolsas que esas directamente tenemos que pesarlas."

❖ **Porque no vienen envasadas en el producto del consumidor final.**

"No vienen envasadas, vienen en un gran bolsón que puede pasar X cantidad de kilos, y a esas lo que nosotros tenemos que hacer directamente almacenarla por ese peso, después la vamos clasificando y ordenando en tandas de tantos kilos. Pero es un bolsón de productos."

❖ **Concretamente, poniéndome en lugar del operario, me llega el palet, lo descargan del camión, desarmo el pallet, agarro un producto, lo cargó y lo guardo, o ¿Cómo es el proceso concretamente?**

"El proceso concreto es como vos bien decis, viene el pallet, ese pallet te llega y ahí se toma un producto y se ingresa dentro de nuestro sistema. Como normalmente son más de un pallet, con solo que abras uno solo de esos pallets, los otros son similares. Es decir, con un solo artículo que saques del pallet, ya podes ingresar 3, 4, 5, 10 pallets, lo que fuera a nuestro sistema."

❖ **¿Ingresan el peso o la cantidad de productos?**

"No, tenes que ingresar todo. Vos tenes que cada pallets pesa tantos kilos, entonces lo que nosotros ingresamos es a través de un artículo, si son tres pallet, tres pallets por lo que pesa cada pallet y sabemos que es, vamos a poner que son alfajores, al ingresar tres pallets por lo que pesa cada pallet de alfajores y todo tendrían la misma fecha de vencimiento en el caso de que ves de que cada pallet es homogéneo."

❖ **Osea ustedes no desarman los pallets en su totalidad, nunca.**

"No, nosotros tomamos de un pallet una muestra y si siempre y cuando cada palet corresponda a esa misma población."

En el complejo ¿Tienen acceso a internet ?

"Si, nosotros trabajamos todo con internet. Todo nuestro respaldo lo tenemos en la nube. Y estamos siempre conectados con distintos bancos a través de internet."

❖ **¿Han intentado anteriormente con soluciones informáticas?, además del sistema que tienen, ¿Han probado con otras ideas?**

"No, este sistema es un sistema estándar como les dije anteriormente, está desarrollado para los bancos que conformamos la red y no hemos probado con otros aplicativos."

❖ **El sistema que usan fue desarrollado por alguien de La Plata, es un software famoso, se puede encontrar en google o es especial a medidas de ustedes?**

"A medida de los bancos, se llama Kolsen, que es el nombre del desarrollador, lo desarrollo para el banco de Buenos Aires."

❖ **Hablaste de distintos tipos de pallets, ¿A qué se refiere?**

"Normalmente están estandarizado, lo que puede pasar es que en los pallets puede haber distintos pesos. Hay pallets de 800 kilos, hay pallets que pesan menos. Depende del tipo de artículo que lleve cada pallet y la posibilidad de apilar que tenga cada artículo. No es lo mismo apilar latas de aceite que apilar alfajores."

❖ **Por lo que nos venía contando, ¿Para ustedes es bastante rápido procesar el ingreso de los productos en caso de venir en pallets?**

"No estoy tan seguro, no te puedo dar una certeza. A mí lo que me parece importante en todo esto que estamos hablando es el método de captura que ustedes previendo, que vos bien decis, ya tener un margen de error en sí mismo y creo que la identificación de productos está muy en desarrollo dentro de lo que ustedes están haciendo.
Vamos concretamente a una aplicación específica, supongamos que esto funciona, que se desarrolla todo bien. En la práctica, como nosotros identificamos, con que método, con qué sistema, con que herramienta identificamos la mercadería que llega?

❖ **Se utiliza una librería que se llama TensorFlow, que se utiliza con un lenguaje de programación Python, esto está localizado en la nube, un servicio conectado a una aplicación que interactúe con la aplicación de ustedes, sin tener que modificar la aplicación.**

¿No hay una interfaz? Sería algo directamente que interactuaba sin interface, ingresaría directamente.

❖ **Debería haber una interface, para que el operario pueda saber que el producto se procesó correctamente o no. Y en caso de procesarlo de manera incorrecta tal vez utilizar el sistema normal, manual.**

La idea también es capturar las imágenes, todas las imágenes sirven, hayan sidas acertadas o no, para después luego poder analizarlas y ver el por qué y poder llegar a conclusiones, si las imágenes, tal vez, había bastante oscuridad, había un operario adelante de la imagen entonces el producto no se veía bien, puede ser varios factores al tomar la imagen para detectar un producto.

¿Con que se captura la imagen física? Cuando llega el pallet ¿Cómo haría el operario para capturar la imagen esa del pallet y poder determinar el input?

❖ **Con una aplicación en un celular, sería lo más accesible. Tal vez el celular en un trípode en un lugar fijo, que la aplicación está siempre activa y cuando el operario pulsa un botón que tome la foto, que la procese y luego que le diga si el producto lo reconoció bien, le muestre los datos. Y algunos campos van a poder completarse, por ejemplo el tipo del producto, la variedad, el peso individual.**

¿La fecha de vencimiento también?

❖ **Esa parte lamentablemente tiene que ser manual. Hasta ayer pensábamos que ustedes desarmaban el pallet y procesan los productos uno por uno, para certificarse que tenga bien la fecha de vencimiento, más que nada por temas jurídicos. Osea asegurarse que corresponda la fecha de vencimiento de cada producto que almacenan. Pero con esto que nos decis que solamente toman un producto de muestra y en base a ese producto de muestra todo lo demás pasa directamente al depósito.**

"Eso es así, sí porque ya todo lo que tenga que ver con lo jurídico está garantizado por el proveedor. Ellos tienen que seguir el código alimentario y lo que está paletizado tienen que responder todos a la misma fecha de vencimiento, osea nosotros manejamos eso.
Y esto del trípode, ¿Necesariamente tiene que estar el celular en un trípode?

❖ **No, es simplemente para garantizar la calidad de la imagen.**

❖ **Entonces de la muestra tendrias el producto identificado, osea la clase y el producto, osea si es fideo y si es un Matarazzo spaghetti de 500 gramos, el SKU el individual digamos. Eso lo tendrías, lo que falta es cuántos de esos estás ingresando y a quéfecha de vencimiento. Eso podrían ser dos inputs que digan cuántos de estas estas ingresando con esta carga, si estas ingresando 500 spaghetti Matarazzo de 500 gramos a una fecha de vencimiento particular.**

Y ese ingreso ¿Se puede hacer vía el celular mismo?

❖ **Lo podrías hacer vía el celular mismo. Y el proceso sería tomar la foto, la procesa, te la devuelve, te dice los datos del producto. Vos ahídecís es correcto si, seguís a la siguiente pantalla, rellenas los campos que faltan, como dijimos recién, cuántos ingresas y la fecha de vencimiento.**
También, otra cosa que pensamos que queríamos hablar con vos, no séqué tan descabellada es la idea de poder utilizar una fecha de vencimiento por defecto. En caso de los fideos tienen mucha fecha de vencimiento, entonces ponerle una fecha de vencimiento por defecto que sea 3 meses, entonces muchas veces no necesitas poner la fecha de vencimiento y que te dé la opción de ponerla específica.

"Es un tema delicado, porque a nosotros el tema fecha de vencimiento es una cuestión que cuidamos muchísimo y poner un default puede hacer que el operador lo deje así, no lo controle y después estamos entregando productos que están vencidos y entramos en un mundo de alta complejidad legal."

❖**Si entiendo, Entonces la fecha de vencimiento debería ingresarse por producto reconocido y la cantidad. También tratar de cargarlo al sistema de ustedes, me imagino que el sistema de ustedes tiene una interfaz para interactuar con otros sistemas de afuera, eso deberíamos verlo en todo caso, es más que nada para que ustedes puedan seguir utilizando su sistema.**

"No, es que nosotros no podemos cambiar de sistema, eso está claro. Lo nuestro es un camino universal y estandarizado y responde a patrones que de alguna manera tenemos un protocolo y toda la información sale de ese sistema, etc. Ese sistema no se tocaría.
Lo que tenemos que pensar es lo siguiente, volvamos al tema porque para ir agilizando y avanzando. Capturamos la imagen, de alguna manera esa imagen tiene que ser validada con todo el mapeo que vos tenes de imágenes ¿Cierto?

❖ **Cierto, antes de poder incluir un nuevo producto al sistema deberíamos conseguir las fotos de ese producto, alrededor de 15 fotos, agregarlas al sistema, procesarlas, que esto tarda 1 o 2 días y después ese producto sería accesible.**
El sistema lleva un soporte detrás, alguien que lo vaya manteniendo y va creciendo y cada día va reconociendo más productos.

Buenos sigamos, tenemos un mapeo de X cantidad de productos

"Vamos al proceso operativo, la persona que está a cargo del depósito nuestro, llega Presto Pronta, captura a través del celular, identifica qué es, identifica el peso, 750 gramos en este caso, si bien que hay polentas de 500 y 1000 gramos, y me va a identificar que son 750 gramos. Este proceso que yo capturo con el celular, ¿Cómo sería la secuencia operativa?, Yo tomé la foto, esto va a consultarse al banco que tiene de foto que tiene la polenta que está en la nube."

❖ **Si este servicio estaría en la nube.**

¿Qué tiempo de respuesta hay ahí?

❖ **Tiene que subir la foto y esto tarda menos de 5 segundos en reconocer el producto.**

Osea que volvería a tener en el celular el ok de que es una polenta Presto Pronto en 5 segundos, algo así.

❖ **Entre 3 y 7 segundos, no más. Hay que subir la imagen dependiendo la conexión de internet, que es como enviar una imagen por Whatsapp, sería un segundo dos segundos en el peor de los casos, el cálculo y la respuesta, como mucho menos de 10 segundos.**

Bien, de a partir de ahí es donde nosotros decimos que probablemente se puede ingresar los datos que tienen que ver con la fecha de vencimiento, quién es el proveedor, porque acá no obtengo el proveedor.

❖ **Claro, no y tampoco tenés manera, porque como dijimos antes distintos proveedores pueden proveer el mismo producto, ese es el tema.**

Claro, y tomando los grandes proveedores, llámese Molinos, llámese Danone, llámese todas esas grandes empresas, ¿Podemos identificar los logos de cada uno de ellos para poder ya desde la captura de la imagen identificar proveedor y artículo?

❖ **En principio sí, pero la pregunta sería más bien ¿Hay posibilidad que un proveedor que no sea Arcor te entregue un producto Arcor?**

No, en general no, pero yo puedo tener un supermercado que me esté donando productos Arcor.

❖ **Entonces, nosotros acá sabemos la marca del producto, nosotros sabemos que Canalé puede ser un proveedor y que Arcor puede ser un proveedor y por default podemos darle esos proveedores como empresas grandes y después que el operario, en caso de no serlo, que ingrese el proveedor que él crea que es el real.**

O sea que vos trabajas que la marca es la mandatoria del proveedor.

❖ **Y si, pasa que el proveedor, vos sabes en este caso de Arcor ¿Cuál sería el proveedor? ¿No es Arcor?**

Puedo tener productos Arcor que no me los está donando Arcor. Porque me lo puede estar donando Arcor, por ejemplo yo trabajo con toda la cadena Jumbo que son los Disco, yo trabajo con Walmart y ellos en góndola tienen productos de Arcor, tienen productos de Molino. Y ahí de una manera el donante no es ni Molinos ni Arcor, es Jumbo y yo después les tengo que responder y dar el remito, porque nosotros tenemos que tener un remito por cada uno de los proveedores por la mercadería que nos entregó, es a Jumbo y no es a Arcor. Ahí yo entro en un mundo de no discriminación de quién es el donante.

❖ **Entonces creo que deberíamos ver por el lado de ingresar el proveedor a mano al igual que la cantidad y al igual que la fecha de vencimiento como ustedes vienen haciendo hoy en día.**

"Les preguntaba esto porque en el mundo nuestro actual es todo manual, es una planilla y ahí ingresa y va todo de una, por eso te preguntaba los tiempos, te preguntaba todas estas cuestiones que hacen la eficiencia."

❖ **Perfecto, ya que estamos ahí no me quiero ir de ahí, porque creo que teníamos una pregunta que no nos la respondió mucho, sobre el ingreso del producto, pero realmente usted cuánto cree que tarda un producto en ese proceso de que bajas el pallet y anotas en la planilla, ¿Es lento?, ¿Es rápido? o ¿Cómo lo ves vos?**

"No, a pesar de ser manual está muy automatizado, tiene una respuesta bastante rápida. Todo depende porque todo puesto en la situación ideal es rapidísimo, puesto en una situación en la que en un momento operativo te están llegando dos camiones juntos como nos suele pasar, ahí ya todo se difiere, toda esta integración de registro a un momento posterior de que se fueran los dos camiones. Y ahí si yo tomo el tiempo desde que llegó el camión hasta que completó el registro, a lo mejor pasó una hora."

❖ **¿Por Entrega?**

"Por entrega, en el camión pueden venir varios pallets de distinta mercadería. Vos estabas poniendo el ejemplo de Arcor, llega Arcor y puede estar trayendo pallets de latas de tomates, pallets de dulce, palets de todos los productos que tiene Arcor. Posteriormente viene otro camión trayendo de Danone, y ahí ya tenemos que bajar ese camión y después todo lo que es tanto de Arcor como de Danone, recién ahí se completa la ficha que es nuestro input al sistema. Entonces si tomas los tiempos entre que llego la mercaderia hasta que la cargas pueden pasar dos horas. Una hora descargando un camión más otra hora descargando el otro camión hasta que se va completando esa planilla. Pero ese registro si lo pudiéramos hacer enseguida se hace muy rápido."

❖ **¿Hoy en día lo están haciendo en una computadora de escritorio?**

"Hoy día hacemos una planilla manual que después nos queda también como forma de control, de verificación y esa después se carga dentro del sistema a través de una computadora personal."

❖ **¿Al final del día o de la semana?**

"No, depende. Puede ser en el transcurso de que ya se registró todo en la planilla ahí se vuelca, puede ser al final del dia. Eso depende de las condiciones en las que estemos

trabajando, como les decía antes nosotros no tenemos un equipo tan grande de gente y el movimiento nuestro es aleatorio, nosotros no trabajamos nada sobre un ordenamiento lógico, porque dependemos de los donantes."

Todo este tema, ¿Qué criterio de rigurosidad para validación, que haya congruencia en los datos habría en el celular? Ahí no le veo mucha posibilidad de seguridad, es lo que ingresa el operador y se acabó.

❖**Primero una consulta, el operador carga los datos, llena una planilla y después un supervisor lo corrobora?**

"En líneas generales, llena el registro y este registro nosotros lo cargamos y después hacemos una validación. Se hace una inspección y ahí estaría una validación un poco a lo que les comentaba. Por supuesto tenemos discrepancias, tenemos errores. pero es un método que tiene bastante consistencia."

❖ **Lo que se podría agregar en ese caso para ayudar también sería la manera de poder ver las órdenes que vos cargaste al sistema y poder modificarlas. Y otra cosa es que te mande un correo electrónico al final del día, cuáles fueron las órdenes creadas, así todos los miembros del Banco de alimento tienen acceso a la planilla directamente y pueden iniciar la validación y pueden detectar algún ingreso no válido.**

"Exactamente, eso sería importante."

❖ **Igualmente, son todas ideas que pueden ser agregadas con el desarrollo, pero nosotros como hasta el alcance de la tesis es más que nada, conocerlos a ustedes la viabilidad de esto si se puede aplicar y más que nada lo importante es nuestros algoritmos de reconocimiento de imágenes que es lo que hemos estado trabajando en eso. Pero si, todo esto lo que es el tema de la validación y el correo electrónico se puede ir agregando, es todo parte del desarrollo.**

"El tema de la validación para nosotros es importante porque puede haber mercadería faltante o sobrante y después entramos a un mundo donde aparecen inconsistencias que pueden ser costosas."

❖**Si, por eso el sistema tiene que brindar esas opciones de poder corregirlas porque somos humanos y lamentablemente los errores van a estar y no se pueden evitar por más sistemas que tengas**

"Exacto, y de una mala captura de datos todo lo que se procesa después es incorrecto."

❖ **Por eso también cuando tomás una imagen y no reconoces un producto, esa imagen no es que se desperdicia, no es que se tira, se guarda para después el desarrollador encargado pueda decidir si la imagen fue un error del sistema o si fue un error del operario al tomar la imagen.**

¿Qué tiene que tener un desarrollador para estar encargado de esto? ¿Que maneje todas estas tecnologías?

❖ **No necesariamente un desarrollador, sino alguien capacitado que pueda llevar adelante la extensión de poder agregar productos semana a semana.**

"Claro, te entiendo, yo para ir viendo todo esto en la realidad nuestra. La realidad nuestra es un proceso muy operativo, es un proceso muy logístico donde no hay lugar donde podamos tener gente específica para mantener un sistema.
A mi la idea me parece muy buena, los que les tengo que decir claramente es que veo que está todo muy embrionario, me imagino que todo esto va a ir avanzando con el tiempo. Podría ser una primer aproximación, pero que nosotros lo incorporamos como un proceso cierto tengo mis serias dudas, cierto y para tenerlo operativo."

❖ **¿Cómo cargan en el sistema? ¿Tienen una página web o una aplicación de escritorio?**

"No, es un sistema tipo Batch, un sistema offline el nuestro. Después si lo que tenemos interconectado es todo lo que puede ser la copia de seguridad, todas esas cosas, pero es un sistema autónomo que tiene la computadora central conectada, se trabaja en red y a partir de eso se hacen los distintos inputs."

ANEXOS 2: PYTHON

Desde su lanzamiento en 1991 el lenguaje de programación Python ha tenido un fuerte crecimiento y en la actualidad, el auge de la inteligencia artificial determinó que sea uno de los más usados para el desarrollo de este tipo de proyectos.
Existen varias razones por las cuales Python es uno de los lenguajes favoritos de los desarrolladores de ML y una de ellas es que cumple con la característica de ser multiparadigma.
Python es un lenguaje de programación interpretado que busca desarrollar una sintaxis que priorice la legibilidad del código y es multiparadigma porque soporta diferentes orientaciones, tal como orientación a objetos, la programación imperativa y funcional.

También, con algunas extensiones disponibles, se pueden soportar paradigmas de programación adicionales y ésto favorece la utilización de diversos estilos.
Esta libertad es una de las características que ha entusiasmado a muchos programadores a desarrollar en Python.

Otra de la singularidad de Python por lo que es muy valorada para desarrollar ML es la legibilidad del código desarrollado, que también es sencillo, elegante y busca ser consistente. Esto permite que la estructura del lenguaje se asemeje a la forma de pensar de los seres humanos y a lo que se conoce como lenguaje matemático, permitiendo que éste sea leído como un pseudocódigo.

La facilidad de su implementación es otro de los motivos que ha llevado a que este lenguaje tenga un fuerte crecimiento en la actualidad.
Adicionalmente a ésto, Python cuenta con iteraciones rápidas de datos que favorecen la concentración de éstos y en el desarrollo de los algoritmos.
Otra particularidad que ofrece este lenguaje y que es muy valorada por los programadores es que cumpliría el rol de funcionar como un lenguaje puente entre el mundo científico y el mundo empresarial.

Al conectar ambos ecosistemas, facilita la creación de códigos entendibles de rápido aprendizaje como los que son necesarios en proyectos de ML.
Las librerías de Python son amplias. Existen miles de librerías de data science y matemáticas, pero el sistema de empaquetamiento de este lenguaje permite construir librerías nuevas sobre las ya existentes para que éstas sean más amplias y potentes.
Por último, es importante destacar que la capacidad de combinar librerías como NumPy y SciPy, permite que Python sea uno de los lenguajes con mejor rendimiento para realizar proyectos de ML

Librerías de Python para Deep Learning

El aprendizaje profundo (en Inglés, Deep Learning - DL) es un subcampo del ML que se ocupa de algoritmos animados por la estructura y función del cerebro llamadas redes neuronales artificiales. Por así decirlo, copia el funcionamiento de nuestras mentes. Aunque el DL se engloba dentro del ML, últimamente, el DL es el mayor responsable de la reciente popularidad del ML.

Los algoritmos de DL son como la organización de los sistemas sensoriales donde cada neurona se asociaba entre sí y transmitía información. El DL es parte de una familia más amplia de métodos de ML basados en representaciones de datos de aprendizaje, a diferencia de los algoritmos específicos de tareas. El aprendizaje puede ser supervisado, semi-supervisado o sin supervisión.

TensorFlow

TensorFlow es una librería de código abierto para DL y ML. Es una librería de python, desarrollada por Google, para realizar cálculos numéricos mediante diagramas de flujo de datos. Esto puede chocar un poco al principio, porque en vez de codificar un programa, modificaremos un grafo. Los nodos de este grafo serán operaciones matemáticas y las aristas representan los tensores (matrices de datos multidimensionales). Con esta computación basada en grafos, TensorFlow puede usarse para DL y otras aplicaciones de cálculo científico.

El grafo que representa la red neuronal profunda y sus datos, se podrá ejecutar en una o varias CPU o GPU en un PC, en un servidor o en un móvil.

Keras

Con Keras es muy fácil experimentar con DL y obtener resultados rápidamente
Keras es una interfaz de alto nivel para trabajar con redes neuronales. La interfaz de Keras es mucho más fácil de usar que la de TensorFlow. Esta facilidad de uso es su principal característica. Con Keras es muy fácil comprobar si nuestras ideas tendrán buenos resultados rápidamente. Keras utiliza otras librerías de DL (TensorFlow, CNTK o Theano) de forma transparente para hacer su trabajo.

Consideraciones entre Keras y TensorFlow

Básicamente, TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático.
Además,es un ecosistema completo y flexible de herramientas, bibliotecas y otros recursos que proporcionan flujos de trabajo con API de alto nivel.
El marco ofrece varios niveles de conceptos para que elija el que se necesita para construir e implementar modelos de aprendizaje automático. Por ejemplo, si el programador necesita realizar grandes tareas de aprendizaje automático, puede usar la API de estrategia de distribución para realizar configuraciones de hardware distribuido.
En cambio, si precisa una tubería de aprendizaje automático de producción completa, simplemente puede usar TensorFlow Extended (TFX).

Las características más destacadas del TensorFlow Extended se describen a continuación:

- Fácil construcción de modelos: TensorFlow ofrece múltiples niveles de abstracción para construir y entrenar modelos.

- Producción robusta de ML en cualquier lugar: TensorFlow permite entrenar e implementar el modelo fácilmente, sin importar el idioma o la plataforma que utilice.

- Potente para la investigación:TensorFlow brinda flexibilidad y control con características como la API funcional de Keras y la API de subclases de modelos para la creación de topologías complejas.

La API de detección de objetos TensorFlow es un marco de código abierto construido sobre TensorFlow que ayuda a construir, entrenar e implementar modelos de detección de objetos.

La API detecta objetos utilizando los extractores de características ResNet-50 y ResNet-101 entrenados en el conjunto de datos de detección de especies de iNaturalist durante 4 millones de iteraciones.

Keras, por otro lado, es una biblioteca de redes neuronales de alto nivel que se ejecuta en la parte superior de TensorFlow, CNTK y Theano.
El uso de Keras en el aprendizaje profundo permite una creación de prototipos fácil y rápida, además de funcionar sin problemas en la CPU y la GPU.
Este marco está escrito en código Python que es fácil de depurar y permite la extensibilidad.

Las principales ventajas de Keras se describen a continuación:

- Fácil de usar: Keras tiene una interfaz simple y consistente optimizada para casos de uso común que proporciona comentarios claros y procesables para los errores del usuario.

Se ofrece una API consistente y simple que ayuda a minimizar la cantidad de acciones del usuario requeridas para casos de uso comunes, también proporciona comentarios claros y procesables sobre el error del usuario.

- De Arquitectura Modular y Componible: los modelos Keras se fabrican conectando bloques de construcción configurables, con pocas restricciones.
- Fácil de ampliar: con la ayuda de Keras, puede escribir fácilmente bloques de construcción personalizados para nuevas ideas e investigaciones.

Diferencias de los Frameworks.

Mientras que Keras es una biblioteca de red neuronal, TensorFlow es una biblioteca de código abierto para varias tareas diversas en el aprendizaje automático. Además, TensorFlow proporciona API de alto nivel y de bajo nivel, mientras que Keras sólo proporciona API de alto nivel. En términos de flexibilidad, la ejecución entusiasta de Tensorflow permite la iteración inmediata junto con la depuración intuitiva.
También Keras ofrece API de alto nivel simples y consistentes y sigue las mejores prácticas para reducir la carga cognitiva para los usuarios.
Ambos marcos proporcionan API de alto nivel para construir y entrenar modelos con facilidad. Además, Keras está construido en Python, lo que lo hace mucho más fácil de usar que TensorFlow.

La API de detección de objetos de TensorFlow es una herramienta poderosa que facilita la construcción, entrenamiento e implementación de modelos de detección de objetos. En la mayoría de los casos, entrenar a una red convolucional desde cero requiere mucho tiempo y grandes conjuntos de datos.

Este problema se puede resolver utilizando la ventaja de la transferencia de aprendizaje con un modelo pre-entrenado utilizando la API TensorFlow. TensorFlow presenta dos variaciones diferentes y ambas se pueden instalar, dependiendo del interés del usuario sobre la plataforma de ejecución elegida.

TensorFlow se puede ejecutar en CPU o GPU, y las variaciones son conocidas como TensorFlow CPU y TensorFlow GPU.

La documentación disponible para el usuario es organizada según la plataforma elegida para la instalación en su computadora, pudiendo instalar una de ellas o ambas.
Es aconsejable que el usuario instale las aplicaciones en entornos diferentes, si desea instalar ambas. La dificultad que se plantea cuando el usuario instala ambas en el mismo entorno es

que se pueden presentar fallas o la incertidumbre del usuario de cual es la variante usada en la ejecución del código.
A los fines de que el usuario no se encuentre en una situación de conflictos de paquetes de diferentes aplicaciones, y que además pueda instalar las variantes deTensorFlow ya nombradas, es aconsejable el uso de entornos virtuales de algún tipo.

En este trabajo han utilizado Anaconda, existiendo otros administradores de entorno virtual como virtual en v, etc.
Para el usuario, una ventaja interesante es que puede instalar ambas aplicaciones ya que el framework logrará detectar el hardware, luego secuencialmente comprobará si es posible utilizar GPU y en caso negativo utilizará CPU.
En este trabajo, los tesistas han preferido utilizar CPU TensorFlow porque han decidido no agregar condicionantes adicionales a nivel hardware como sería instalar además GPU TensorFlow.
Esta flexibilidad en la instalación es muy valorada por los usuarios al decidir utilizar TensorFlow.

ANEXOS 3: Herramientas Generales

Anaconda como gestor de entornos y dependencias:

Anaconda es una distribución de python para Cálculo Numérico, Análisis de Datos y ML. Contiene las librerías más usadas por los científicos de datos. Además hace muy fácil la instalación de otras librerías que sea necesario utilizar.

Anaconda también permite crear varios entornos de trabajo si el usuario trabaja en varios proyectos. Esto puede ser útil, por ejemplo, si uno de los proyectos necesita python 3 y el otro python 2.
También es de utilidad si se trabaja en un proyecto que necesita librerías específicas o que tengan una versión específica.
Es recomendable que, de no trabajar con aplicaciones antiguas, se utilice la distribución de Anaconda con Python 3.

Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente.
Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.
Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o frontend como Cogito o StGIT.Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena.
Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux.

En cuanto a derechos de autor Git es un software libre distribuible bajo los términos de la versión 2 de la Licencia Pública General de GNU.
Uno de los usos de git es el de trackear y gestionar los cambios en el proyecto de una manera organizada.Es por ello que es muy útil para hacer cambios temporales utilizando ramas para cada prueba que se haga.Esta ventaja es valorada cuando se tienen que hacer muchos cambios dentro del framework y de otra manera se perdería fácilmente el control de lo que se configura.

Jupyter

Durante el proceso, en la exigencia de ejecutar grandes cadenas de comandos para ejecutar un proceso es muy difícil recordarlas y para cada acción encima del framework todo se maneja a través de comandos en consola.
En vez de crear scripts que puedan ayudar para evitar repetir una y otra vez las ejecuciones, el mundo python provee una herramienta un poco más interactiva y amigable la cual puedes crear nuestra ejecución de manera más interactiva, pudiendo así ver los resultados y errores de manera más rápida. Esa herramienta es Jupyter.

Tensorboard

TensorBoard es un conjunto de aplicaciones web para inspeccionar y comprender sus ejecuciones y gráficos de TensorFlow. Actualmente, TensorBoard admite cinco visualizaciones: *escalares, imágenes, audio, histogramas y gráficos*. Los cálculos que usará en TensorFlow para cosas como entrenar una red neuronal profunda masiva, pueden ser bastante complejos y confusos, TensorBoard hará que esto sea mucho más fácil de entender, depurar y optimizar sus programas de TensorFlow.

Las dos principales ventajas de TensorFlow sobre muchas otras bibliotecas disponibles son la flexibilidad y la visualización. Imagínese si puede visualizar lo que está sucediendo en el código (en este caso, el código representa el gráfico computacional que creamos para un modelo), sería muy conveniente comprender y observar profundamente el funcionamiento interno del gráfico. No solo eso, también ayuda a arreglar las cosas que no funcionan como deberían. TensorFlow proporciona una forma de hacer precisamente eso con TensorBoard.

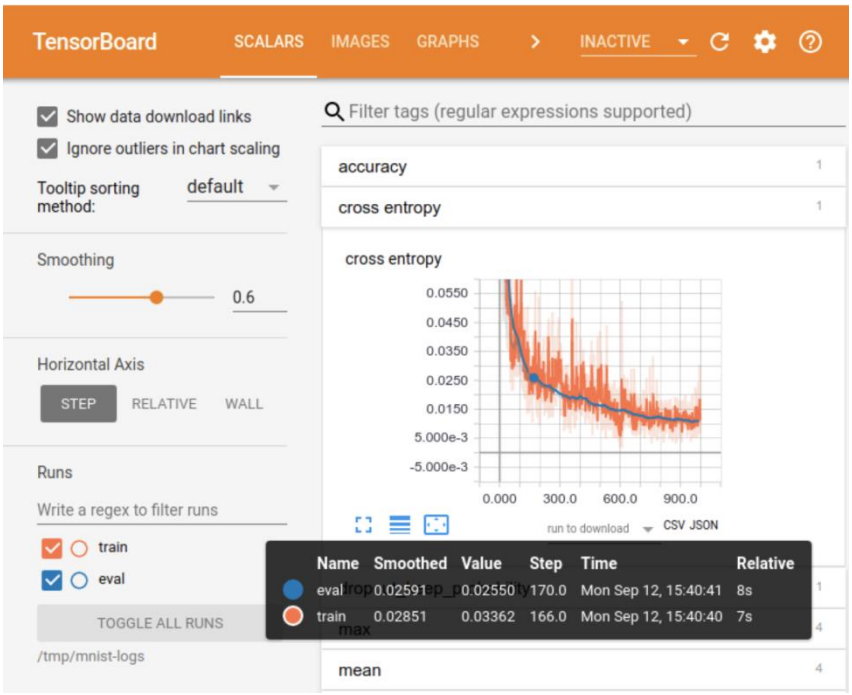

Figura 53. Tensorboard apariencia
(Elaboración propia,2020)