Uso de la Técnica de Transfer Learning en Machine Learning para la Clasificación de Productos en el Banco Alimentario de La Plata

Agustín De Luca[1], Matías Irigoitia[1] , Gabriela Pérez[1,3] , Claudia Pons[1,2,4]

[1] Facultad de Informática, Universidad Nacional de La Plata,
[2] Comisión de Investigaciones Científicas CIC,
Bueno Aires, Argentina
[3] UNAJ, Universidad Nacional Arturo Jauretche
Florencio Varela, Buenos Aires, Argentina
[4] UAI, Universidad Abierta Interamericana
Ciudad de Buenos Aires
gabriela.perez@lifia.info.unlp.edu.ar, claudia.pons@lifia.info.unlp.edu.ar

Abstract. La presente investigación analiza la técnica de Transfer Learning en Machine Learning y realiza una comparación de distintos modelos pre-entrenados para facilitar los procesos de ingreso de productos al sistema del Banco Alimentario de La Plata, a partir del reconocimiento de imágenes. Con este fin se construyó un prototipo de aplicación que además de constituir una prueba de concepto de Transfer Learning, brinda una herramienta que facilita el accionar cotidiano del Banco Alimentario, contribuyendo a la meritoria tarea que éste lleva adelante.

Keywords: Inteligencia artificial, Machine learning, Transfer learning, TensorFlow, Keras, Redes Neuronales Convolucionales, Modelos pre-entrenados, COCO , MobileNet, Inception, RCNN, SSD..

1 INTRODUCCIÓN

Machine Learning o Aprendizaje Automático (ML por sus siglas en inglés) [1] es una rama de la Inteligencia Artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan a resolver problemas analizando datos y buscando patrones en ellos. Según este enfoque, los datos tienen un papel fundamental, ya que a partir de ellos se generará un programa que debe ser capaz de generalizar comportamientos e inferencias que permitirá trabajar con 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 utilizadas, principalmente gracias a la disponibilidad de grandes volúmenes de datos y al desarrollo de nuevas tecnologías que permiten aumentar la capacidad de procesamiento disponible. En pocos años se pasó de modelos con cientos de parámetros a otros que requieren millones. Entrenar estos modelos requiere cómputo, tiempo y una gran cantidad de datos, pero no siempre se encuentran disponibles. Por ello,

hay investigaciones que se enfocan en poder reutilizar estos modelos ya desarrollados y entrenados, en otros problemas similares. La técnica de Transfer Learning o Aprendizaje por Transferencia (TL por sus siglas en inglés) [2] [3] es un problema de investigación en ML que se enfoca en almacenar el conocimiento adquirido al resolver un problema, para aplicarlo a un problema diferente pero relacionado.

Con el objetivo de analizar la técnica de TL en ML, se desarrolló un caso de estudio: la clasificación de productos para el Banco Alimentario de La Plata (http://bancoalimentario.org.ar). Esta organización 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. Para ello, recibe y distribuye anualmente más de 500.000 toneladas de alimentos. Hoy en día dicho Banco Alimentario aplica procesos manuales o utilizando código de barras para registrar los ingresos de productos, en lo cual se observa una oportunidad para la aplicación de algoritmos de detección automática, en búsqueda de agilizar el proceso.

Este artículo está organizado de la siguiente forma. En la sección 2 se describe el Banco Alimentario. La sección 3 está dedicada a la técnica de Transfer Learning. En la sección 4 se explican las estrategias de entrenamiento de los modelos aplicando estas técnicas. En la sección 5 se compara el entrenamiento de distintos modelos pre-entrenados, para determinar cuál obtiene mejores resultados para el reconocimiento de productos para el Banco Alimentario. En la sección 6 se plantea una aplicación móvil para automatizar el proceso de ingreso de mercadería al banco utilizando ML. Finalmente, en la sección 7 se presentan las conclusiones y líneas de trabajo futuro.

2 EL BANCO ALIMENTARIO

El Banco Alimentario de La Plata es una sociedad civil sin fines de lucro fundada en el año 2000. Emplea un modelo que se basa en experiencias internacionales, el cual pretende poner en valor los 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, comerciales o de producción, no pueden ser comercializados.

La alimentación es un tema delicado, por lo que el Banco Alimentario cuenta con protocolos y normas estrictas en todos los procesos que realiza. Estos procesos se revisan mediante auditorías regulares junto con otros aspectos de gestión del Banco.

En este momento en Argentina existen 14 bancos operativos de alimentos. Estos integran una red para coordinar los esfuerzos, compartir experiencias, 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 hace que el Banco Alimentario de La Plata deba estar integrado y protocolizado de acuerdo con normas de buenas prácticas de manufactura de esta red internacional.

La función del Banco Alimentario es conseguir donaciones, las cuales deben ser ingresadas al depósito para luego clasificarlas.

Luego, se debe distribuir, para destinar la mercadería a los comedores, según los productos con los que se cuenta y las necesidades de cada una de las instituciones. Para ello, el Banco cuenta con una base de datos de instituciones que brindan servicios alimentarios. Este registro permite conocer 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.

2.1 Problemáticas

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

El sistema que se usa actualmente, llamado Kolsen, es un sistema informático de aproximadamente 20 años. Este no cuenta con actualizaciones. No es posible resolver inconsistencias en la versión actual, ni permitir integraciones con otros sistemas, generando limitaciones de escalabilidad.

2.2 Solución propuesta

Se construyó un prototipo para detección y clasificación de productos alimenticios a partir de imágenes, detallando el proceso de desarrollo seguido. Además de constituir una prueba de concepto de Transfer Learning, este prototipo brinda una herramienta que facilitará el accionar cotidiano del Banco Alimentario de La Plata, contribuyendo a la meritoria tarea que éste lleva adelante.

3 TRANSFER LEARNING

El Transfer Learning o Aprendizaje por Transferencia es un área dentro del Aprendizaje Automático, que focaliza en la incorporación de conocimiento durante la resolución de un determinado problema y su posterior aplicación sobre otro diferente pero relacionado [2].

En ML tradicional, se genera un sistema (modelo) de aprendizaje donde cada tarea se aprende desde cero. En cambio, en TL se construye una base de conocimiento a partir del entrenamiento de tareas (origen), para que sea posible transferirla hacia otra tarea (destino) requiriendo menos cantidad y calidad de datos de entrenamiento [3].

Existen tres medidas comunes por las cuales la transferencia podría mejorar el aprendizaje. La primera es el rendimiento inicial que se puede lograr en la tarea destino, utilizando sólo el conocimiento transferido, antes de realizar cualquier aprendizaje adicional, en comparación con el rendimiento inicial de un agente ignorante. La segunda es la cantidad de tiempo que lleva aprender completamente la tarea destino dado el conocimiento transferido, en comparación con la cantidad de tiempo para aprenderlo desde cero. Y la tercera es el nivel de rendimiento final alcanzable en la tarea destino en comparación con el nivel final sin transferencia.

La aplicación de metodologías de TL incrementa el grado de generalización de los modelos de ML, permitiendo la utilización de conjuntos de datos de menor tamaño (decenas o

cientos de instancias) y mayor simplicidad (baja dimensionalidad). Además, posibilita el ahorro de tiempo de aprendizaje (aceleración) y disminuye los recursos de cómputo requeridos flexibilizando el requerimiento de hardware de alta prestaciones. Esto permite un rendimiento elevado de los modelos con un menor número de iteraciones para el entrenamiento. Por último, permite representaciones más robustas y de amplia aplicación [3].

3.1 Estrategias de Transfer Learning

Existen diferentes estrategias y técnicas de *TL* que pueden aplicarse en función del dominio, la tarea en cuestión y la disponibilidad de datos. Se pueden clasificar, según el tipo de algoritmos tradicionales de *Machine Learning* involucrados, en tres grupos: Inductivo, no supervisado y Transductivo.

Considerando la interrogante "¿qué transferir?" y las técnicas previamente mencionadas surgen cuatro enfoques, cuyo tratamiento permite evaluar los aspectos particulares de cada problemática y aplicar las metodologías más adecuadas. Estos enfoques son:

- **Transferencia de instancias:** se considera que ciertas partes de los datos del dominio de origen son útiles en el entrenamiento sobre el dominio objetivo y pueden reutilizarse.
- **Transferencia de representación de características:** la idea subyacente es el descubrimiento y aprendizaje de una "buena" representación de características del dominio destino a fin de reducir diferencias entre dominios y el error del modelo generado.
- **Transferencia de parámetros: se** asume que las tareas origen y destino comparten parámetros o distribuciones previas de hiperparámetros de los modelos. El conocimiento transferido es codificado en parámetros compartidos.
- **Transferencia de conocimiento relacional:** supone la existencia de relaciones entre los datos de ambos dominios, asumiendo que se trata de dominios relacionales. Esas relaciones entre dominios representan el conocimiento a transferir mediante la construcción de mapeos de conocimiento relacional.

Para el desarrollo del sistema para el Banco Alimentario se utilizaron dos de estas estrategias de TL: Transferencia de representación de características y la Transferencia de parámetros a través de la aplicación de modelos pre-entrenados.

3.2 Redes neuronales pre-entrenadas

Un modelo es creado y entrenado con una gran cantidad de datos para resolver un problema.[4]. Cuando se necesita resolver un problema similar, en lugar de construir un modelo desde cero, se puede utilizar este modelo pre-entrenado, es decir, entrenado en otro problema, como punto de partida.

Se debe tener mucho cuidado al elegir el modelo pre-entrenado que se va a utilizar en cada caso. Si el problema es muy diferente, la predicción que se obtiene será muy inexacta. Existen muchas arquitecturas pre-entrenadas que están directamente disponibles para su uso en la biblioteca de Keras [5]. Algunas de ellas, se construyeron utilizando como datos de entrada el conjunto de imagenes conocido como ImageNet [6]. Este conjunto es lo suficientemente grande (1.2M de imágenes) para permitir crear un modelo generalizado.

Estos modelos clasifican correctamente las imágenes en 1000 categorías de objetos separadas. Estas categorías representan clases de objetos que se encuentran en la vida cotidiana, como especies de perros, gatos, diversos objetos domésticos, vehículos, etc. Estos modelos pre-entrenados demuestran una gran habilidad para generalizar a imágenes fuera del conjunto de datos ImageNet a través del Transfer Learning.

4 APLICANDO TRANSFER LEARNING

4.1 Factores que afectan el rendimiento

Como punto de partida, es importante analizar qué factores afectan el rendimiento al entrenar un modelo [7]:

- El procesamiento de cada imagen: cada imagen es una matriz de 3 dimensiones que debe ser recorrida en la convolución, a través del kernel. Esto depende fuertemente del tamaño de la imagen y la configuración de la convolución. Este proceso influye en el tiempo de procesamiento.
- El tiempo de entrenamiento: la capacidad de cómputo es un factor clave en el tiempo necesario para entrenar esa red.
- Tipo de software utilizado: La selección del software depende de la tecnología utilizada y de la capacidad de aprovechar el hardware disponible para entrenar.
- Número de clases definidas: el número de clases simboliza cada objeto individual que la red podrá reconocer.
- Tamaño del dataset: Si se construye un dataset de mayor tamaño, exigirá un tiempo de entrenamiento más prolongado.
- El uso de Image Augmentation: existen herramientas para rotar e invertir cada imagen del dataset, logrando un aumento considerable de éste, y como consecuencia, mejorar la calidad del aprendizaje.
- Configuraciones de los hiper parámetros: son parámetros de configuración que afectan al aprendizaje tales como batchsize, learning rate, etc.
- Calidad del etiquetado: cada imagen del dataset debe estar etiquetada de forma correcta

4.2 Transfer Learning y las estrategias de uso de modelos pre-entrenados

Cuando se utiliza un modelo pre-entrenado, se comienza eliminando el clasificador original. Luego se agrega uno que se adapte a los propósitos propios, y finalmente se realizan ajustes de acuerdo con una de las siguientes estrategias:

Estrategia 1. Se entrena a todo el modelo. En este caso, se utiliza la arquitectura del modelo pre-entrenado y se lo entrena con su conjunto de datos. El modelo aprende desde cero, por lo que se necesitará un gran conjunto de datos y mucha potencia computacional.

Estrategia 2. Se entrenan algunas capas y se dejan 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, mas dependientes del problema. Se debe balancear esta dicotomía eligiendo cuánto se quiere ajustar los pesos de la red. Las capas que se mantienen congeladas no cambian durante el entrenamiento. Por lo general, si se tiene un conjunto de datos pequeño y una gran cantidad de parámetros, se dejarán más capas congeladas. 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.

Estrategia 3. 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 1 presenta estas tres estrategias de manera esquemática.

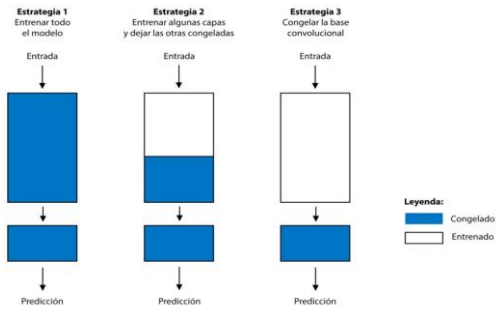

Figura 1. Estrategias de ajuste

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 convoluciones, es conveniente usar una tasa de aprendizaje pequeña, porque las tasas de aprendizaje altas 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 red de manera abrupta.

4.3 Proceso de Transfer Learning

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 al problema.

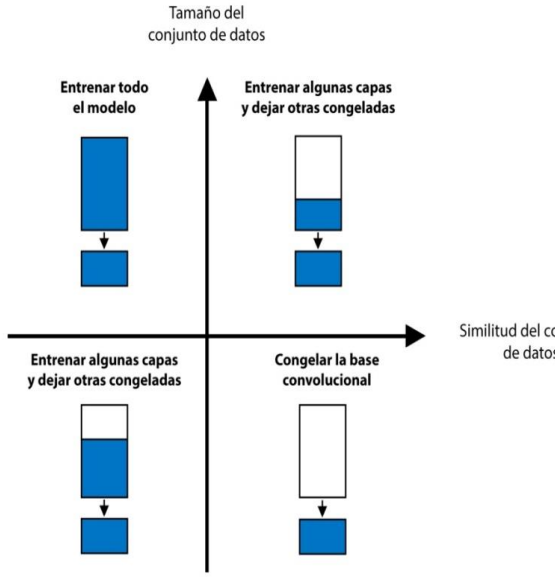

Figura 2. Mapa de decisión para ajustar modelos pre-entrenados

Clasificación del problema de acuerdo con la Matriz de similitud de tamaño. La Figura 2 muestra un mapa que guía en la eleccion del modelo pre-entrenado. Este mapa clasifica el problema 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 [6] 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.

Refinamiento del modelo. Aquí se puede usar el mapa 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 2 proporciona un resumen visual de cada uno de los cuatro cuadrantes de ese mapa:

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. 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. 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 del cuadrante 2, esta es la peor situación. En este caso se debe optar por la Estrategia 2. Se dificulta encontrar un equilibrio entre la cantidad de capas para entrenar y para congelar. 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 este caso, se deberá optar por la Estrategia 3. Sólo 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.

5 PROCESO DE TRANSFER LEARNING PARA EL BANCO ALIMENTARIO

En esta sección se describe el proceso de Transfer Learning llevado adelante para construir la solución de reconocimiento de imágenes de productos alimenticios para el Banco de La Plata.

El primer paso consistió en la selección de los potenciales modelos pre-entrenados. De la amplia gama de modelos disponibles, se eligió uno que se adapta al problema. Luego se clasificó el problema de acuerdo con el mapa de similitud de tamaño. Se ubicó el problema en el cuadrante 4 dado que el tamaño del conjunto de datos disponibles es pequeño y su similitud con COCO [9], el conjunto de datos en el que se formaron los modelos pre-entrenados es alta. Finalmente se procedió al ajuste del modelo con los nuevos datos. Por tratarse de un problema en el cuadrante 4 se optó por la Estrategia 3 (congelar la base convolucional) donde se está utilizando el modelo pre-entrenado como un mecanismo de extracción de características fijas. Las capas del modelo pre-entrenado se congelan, se elimina la última capa (capa de salida), para luego usar las características resultantes para entrenar a un nuevo clasificador.

5.1 Descripción de los modelos pre-entrenados utilizados en este proyecto

Se analizó el *TensorFlow Model Zoo* [8] que proporciona una colección de modelos de detección previamente entrenados utilizando conjuntos de datos open source, tales como el conjunto de datos Common Object in COntext (COCO) dataset [9]. COCO contiene 91 categorías de objetos comunes. 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 entrenar modelos capaces de localizar objetos de forma precisa.

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.

Existen decenas de modelos disponibles de los cuales se han seleccionado tres modelos para este desarrollo, uno de tipo Faster Region-based Convolutional Neural Network abreviado como Faster R-CNN y dos de Single Shot Multibox Detection (SSD), que son SSD_mobileNet y SSD_Incepcion. A continuación, se describen los dos tipos de modelos evaluados.

Faster Region-based Convolutional Neural Network (Faster R-CNN). Una red Faster R-CNN [10] toma como entrada una imagen completa y un conjunto de propuestas de objetos, como se ilustra en la fig.3. La red primero procesa la imagen completa con varias capas convolucionales y *maxpooling*, 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.

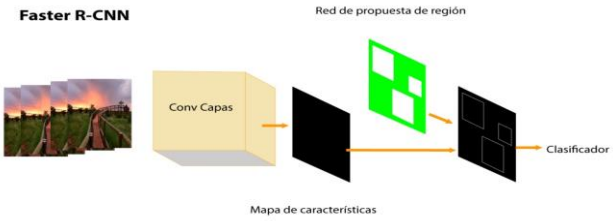

Figura 3. División de capas Faster RCNN

Cada vector de características alimenta a una secuencia de capas completamente conectadas 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 un cuadro delimitador refinadas para una de las K clases [10].

Single-Shot Multibox Detection (SSD). SSD [11] ofrece un nuevo enfoque de detección de objetos como alternativa a los enfoques basados en regiones anteriores (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 mayor que el enfoque Faster R-CNN.

La arquitectura SSD se ilustra en figura 4. Consiste en una red neuronal convolucional base seguida de capas convolucionales multibox. La red base y las capas multibox predicen la presencia de instancias de clase de objeto y las ubicaciones del cuadro delimitador [11].

Hay seis capas de predicción que disminuyen el tamaño del mapa de características. Esto permite que el SSD maneje la detección de objetos en múltiples escalas.

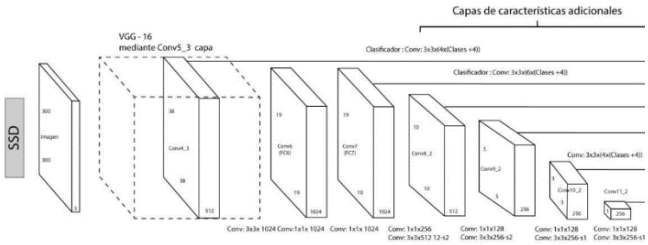

Figura 4. Single Shot Multibox Detection SSD

5.2 Construcción del entorno de prueba

A continuación, se procederá a detallar las evaluaciones realizadas sobre los tres modelos descriptos anteriormente.
El hardware utilizado en este trabajo de investigación fue:
Sistema operativo Windows 10, Intel core i7 7700k 3.9ghz con una arquitectura de cuatro núcleos con hyperthreading.
El software utilizado fue TensorFlow versión 1.13, el repositorio IA de Tensorflow Object Detection [14]. Para la implementación de código Python se utilizó el entorno de trabajo Anaconda.

5.3 Definición del Dataset

En este trabajo, cada dato es una imagen de un producto alimenticio asociado a una metadata o *label* como se refleja en la figura 5 que ha sido creado en el proceso de creación de dataset y *labeling* [17].

Figura 5. Dataset. Imágenes con sus metadatos.

El dataset se divide en un conjunto de datos para entrenamiento y otro conjunto de datos para validación. El conjunto de datos de entrenamiento es el que se usa para entrenar un algoritmo 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 aprendió el algoritmo.

Usualmente el conjunto de datos completo se fracciona en dos subconjuntos, uno de un 80~90% del conjunto original para el entrenamiento en sí y el 10~20% restante para su validación.

5.4 Data augmentation.

Data augmentation [18] consiste en aplicar cambios geométricos a todas las imágenes del dataset como recortes, rotaciones, etc, con el fin de aumentar el tamaño inicial del dataset. Esto, no solo es útil para el caso de contar con datos limitados sino también para mejorar la generalización de los modelos debido a que aumenta la diversidad del conjunto final.

Esta técnica fue aplicada al dataset del Banco Alimentario para facilitar el proceso de creación del dataset, como una alternativa complementaria a la captura y etiquetado manual de imágenes.

5.5 Descripción de las pruebas realizadas

El objetivo principal de las pruebas fue evaluar los modelos seleccionados para elegir el más apropiado según ciertas métricas que se describirán más adelante. Como se menciona anteriormente hará la comparación de tres modelos pre-entrenados en pruebas controladas de forma objetiva.

Se realizaron pruebas de entrenamiento, midiendo la capacidad de aprender de cada modelo utilizando el mismo dataset. Para ello se contó con 97 imágenes totales, de las cuales se separaron 77 imágenes para entrenamiento y 17 para validación, y se entrenó durante un periodo de 160 minutos. Se utilizaron técnicas de *data augmentation* para incrementar el tamaño del dataset inicial en cada etapa.

5.6 Métricas para evaluar

Durante las pruebas y en el transcurso del entrenamiento de cada modelo, se monitorearon dos métricas fundamentales para el objetivo propuesto.

– Función de Pérdida (o *Loss function*): Esta medición fue realizada en cada iteración sobre el dataset de entrenamiento utilizando la herramienta *TensorBoard*.

– Precisión (o *Accuracy*): este valor fue medido cada 5 minutos mientras se ejecuta el algoritmo de validación sobre del dataset de validación.

Se observaron y se registraron las métricas durante todo el proceso que duró 160 minutos.

5.7 Función de pérdida

La función de pérdida se utiliza para evaluar qué tan bien responde un algoritmo.
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 pérdida es una función logarítmica decreciente que debe tender a cero ya que de otra manera significa que se produjo un problema en el entrenamiento.

No existe una función de pérdida única para todos los algoritmos en el aprendizaje automático, ello significa que cada modelo responde y se comporta diferente en base a su arquitectura. No se puede afirmar o establecer un valor deseado en la función de pérdida, pero sí se puede esperar un patrón de algoritmo decreciente tendiendo a 0 a lo largo del entrenamiento durante los 160 minutos.

5.8 Precisión de la clasificación

La precisión de clasificación se compone de dos métricas: accuracy y recall.

La tasa de aserción (Accuracy en inglés) representa la fracción de elementos de una clase dentro del total de elementos asignados a esa clase (es decir, la cantidad de verdaderos positivos, sobre la suma de verdaderos positivos y los falsos positivos)

$$P(c) = \frac{VP}{VP + FP}$$

Complementariamente el Recall representa la fracción de elementos asignados a una clase dentro del total de elementos de esa clase (es decir, la cantidad de verdaderos positivos, sobre la suma de verdaderos positivos y los falsos negativos).

$$R(c) = \frac{VP}{VP + FN}$$

Y finalmente dado que ni accuracy ni recall pueden determinar individualmente el verdadero rendimiento del clasificador, se calcula la media armónica entre ambas, denominada Score, como se muestra a continuación

$$FS(c) = 2\left(\frac{P(c)R(c)}{P(c) + R(c)}\right)$$

No se puede establecer un valor deseado, 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.

5.9 Resultados obtenidos

Evaluación de Faster-RCNN.

El entrenamiento se detiene manualmente a los 160 minutos con una pérdida de 0.027 en ese punto. La Figura 6 muestra la función de pérdida durante el entrenamiento. Para Faster-RCNN, este valor se logra a los 4388 pasos. Por otra parte, se logra 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 de las 17 imágenes.

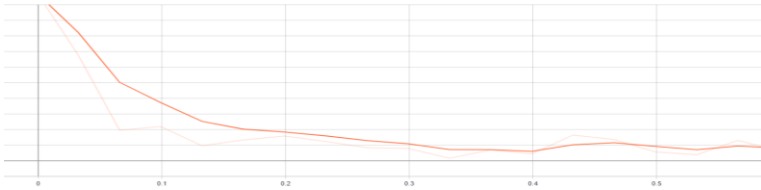

Figura 6. función de Pérdida de Fast-RCNN, TensorBoard

Evaluación de SSD MobileNet.

Como se indicó, el entrenamiento se detiene manualmente a los 160 minutos con una pérdida de 2.264 a ese punto, La Figura 7 muestra la función de pérdida durante el entrenamiento. Para SSD MobileNet, este valor se logra a los 479 pasos. Por otro lado, se logra 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 de las 17 imágenes.

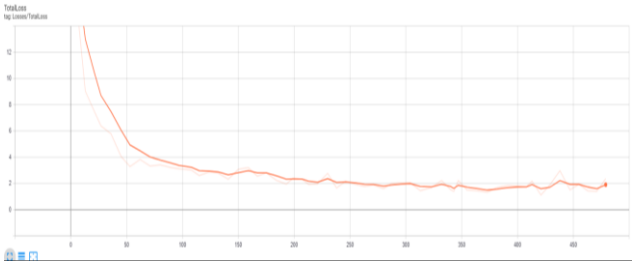

Figura 71. función de Pérdida de SSD MobileNet, TensorBoard

Evaluación de 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 8 muestra la función de pérdida de entrenamiento. Para SSD Inception, este valor se logra a los 401 pasos. Por otra parte, se logra 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 de las 17 imágenes.

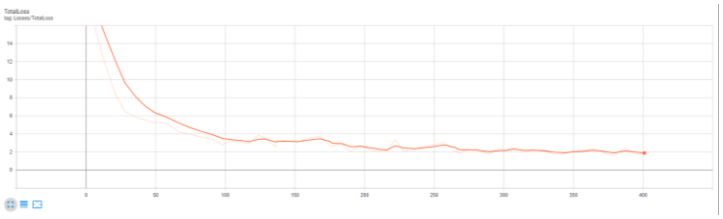

Figura 8. Función de Pérdida de SSD Inception, TensorBoard

5.10 Resultados

En este trabajo se logró comparar rigurosamente los entrenamientos de los tres modelos pre-entrenados seleccionados, mostrando sus prestaciones. A partir de esta comparación se pudo decidir cuál es el que adquiere conocimiento de forma más rápida y logra mejor rendimiento. Así, se pudo elegir el más apropiado para seguir adelante con el desarrollo de reconocimiento de alimentos para el Banco Alimentario.

Los experimentos sugirieron que el enfoque de TL en tareas de detección y clasificación de objetos ofrece un resultado excelente aun con un número moderado de muestras. De las pruebas realizadas con los tres modelos, se puede concluir que Faster-RCNN es el modelo que brindó mayor precisión con un *accuracy* del 94.7% como muestra la Tabla 1. También funciona aceptablemente en la detección de objetos y mostró rápida convergencia durante la fase de entrenamiento.

En cuanto a SDD, produjo un mayor número de falsos negativos en los experimentos.

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

Tabla1. Resultados de la evaluación.

6 Propuesta para el Banco Alimentario

Como solución a las problemáticas existentes en el proceso de ingreso de mercadería al sistema actual del Banco Alimentario, se pensó en un sistema coexistente al Kolsen, que facilite el cargado de los productos ingresantes, utilizando inteligencia artificial para el reconocimiento de éstos a través de imágenes. El sistema también ayudará en las tareas de generar el remito para el donante y en la validación de los datos cargados.

El prototipo implementado se compone de recursos almacenados en la nube (*cloud*) con un algoritmo de reconocimiento de imágenes, el cual recibe imágenes de productos, los reconoce y retorna información referida a éstos, como el código de producto, el nombre y su peso unitario. Actualmente está entrenado para reconocer aproximadamente 15 imágenes de los productos que habitualmente el Banco Alimentario recibe. Se planea continuar entrenando al algoritmo con las fotografías de todos los productos, progresivamente hasta llegar a su totalidad.

6.1 Interfaces de usuario

Se diseñó una aplicación para *smartphones*, donde los operarios pueden 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 Kolsen, para que las entradas sean cargadas allí.

A continuación se explica el proceso que deberían realizar los operarios para registrar una orden.

Como se ejemplifica en la figura 9a, al ingresar a la aplicación *mobile*, se ven las órdenes ya registradas y un botón para agregar una orden.

Figura 9a. Menú inicial de la aplicación

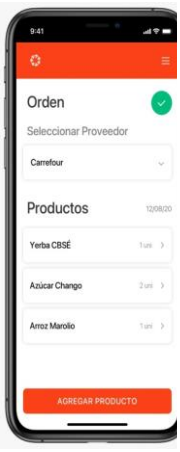

Figura 9b Ventana de tap

Figura 9c. Incorporación del producto

La figura 9b muestra la ventana que permite agregar una orden. En esta ventana se pide seleccionar cuál es el proveedor o ingresar su nombre en caso de que no se encuentre. Debajo aparecen, si es que hay, los productos ya agregados y los botones de agregar producto y de finalizar orden. Al finalizar orden, se almacena se envía al sistema Kolsen para ser persistida. Al agregar un producto, se abre la cámara del móvil, que permitirá tomar una imagen del producto. Este paso está representado en la figura 9c.

Luego, la imagen es enviada al servidor donde es procesada y la respuesta obtenida es enviada al móvil. Mientras tanto, en la pantalla se informa que la imagen está siendo procesada.

Una vez que se recibe la respuesta, se despliega en una nueva pantalla los datos recuperados del producto, su código, nombre y

el peso unitario. Además, se deben completar otros campos de forma manual, ya que esos datos no se pueden recuperar a partir de la imagen: la cantidad y la fecha de vencimiento (ver figura 10a).

Al finalizar la orden, se muestra la información completa en una pantalla (ver figura10b).

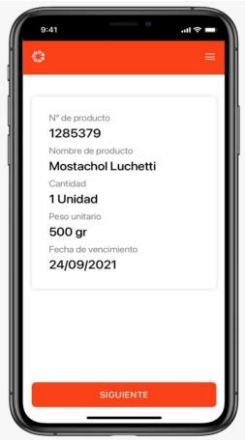

Figura 10a
Información del producto

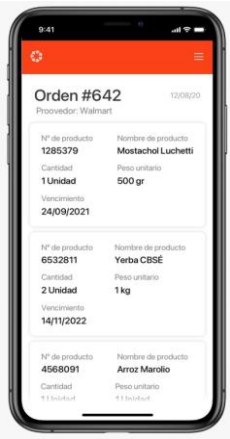

Figura 10b
Información de orden

6.2 Servicios en la Nube

Analizando y entendiendo las problemáticas planteadas por el Banco Alimentario, se plantea tener disponible el modelo *tensorflow* como un servicio *cloud*, acompañado de un servicio REST para exponer estos servicios. En el diagrama de la figura 11 se detalla la arquitectura del sistema.

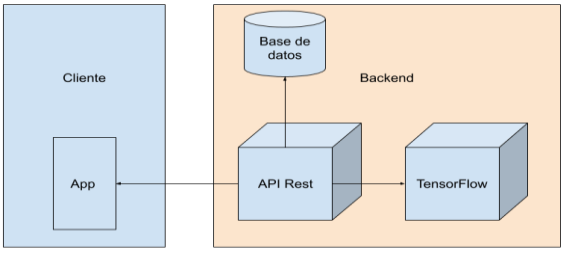

Figura 11. Diagrama de arquitectura

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.

Del lado del *backend* se encuentra disponible la base de datos en la cual se persisten las órdenes detallando sus productos y proveedor. Además se encuentra el *modelo tensorflow*, con el cual se reconocen los productos a partir de las imágenes enviadas.

Además, como conector de 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 resultados obtenidos. El servicio también se encarga de comunicarse con el *modelo tensorflow* para enviar las imágenes y recibir los datos recuperados de estas. Por último, se encarga de persistir las órdenes en la base de datos.

7 CONCLUSIONES

El Aprendizaje por Transferencia es un problema de investigación en Aprendizaje Automático que se enfoca en almacenar el conocimiento adquirido al resolver un problema, para que pueda ser aplicado a un problema diferente pero relacionado.

El objetivo principal de esta investigación es el análisis de la técnica de Aprendizaje por Transferencia a través de un caso de estudio: la clasificación de productos para el Banco Alimentario de La Plata.

Para cumplir este objetivo, se realizó un estudio de las tecnologías asociadas a Transfer Learning, de las herramientas de apoyo para el procesamiento de datos y de los sistemas de métricas para evaluar rendimientos.

Se realizaron pruebas controladas de varios modelos en las mismas condiciones. Concretamente, se han puesto a prueba tres modelos pre-entrenados de distintas características: Faster-RCNN, SSD Inception y SSD MobileNet.

El Faster-RCNN ha obtenido casi un 95% en accuracy, mientras que SSD inception y SSD mobileNet que no llegaron al 80%.

Si bien Faster-RCNN tardó más en detectar una imagen por tener una capa de detección de regiones sobre la imagen, esta dificultad no se consideró relevante para el objetivo de la aplicación planteada.

En conclusión, teniendo en cuenta los resultados obtenidos y las investigaciones realizadas, es posible afirmar que la aplicación de Aprendizaje por Transferencia es conveniente para resolver el problema propuesto, ya que facilita los procesos de entrada de productos al sistema.

7.1 Desarrollos Futuros

Si bien la propuesta presentada alcanza para cumplir con los objetivos acordes a la problemática planteada, existen varios puntos de extensión a futuro, tales como la construcción de un Portal web, un Servicio de reportes, un Registro de entregas a comederos y un Sistema para permitir Analíticas.

8 BIBLIOGRAFÍA

[1] Goodfellow I., BengioY.,y Courville A. (2015). Deep Learning, (Adaptive Computation and Machine Learning series),

[2] Sinno Jialin Pan and Qiang Yang (2010) “A Survey on Transfer Learning”. IEEE Transactions on Knowledge and Data Engineering, (22)10. 1345-1359

[3] Torrey, L y Shavlik J, (2009). Transfer Learning. En Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. p.834. Universityof Wisconsin.

[4] Model Zoo - Deep learning code and pretrained models for transfer learning, educational purposes, and more. Recuperado de: https://modelzoo.co/(fecha de ingreso: 25/06/2020)

[5] Keras (https://keras.io/about/)

[6] 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

[7] 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.

[8] 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 de ingreso: 7/01/2021)

[9] 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

[10] Girshick, Ross (2015). “Fast R-CNN”. Microsoft Research. p. 1-9 Recuperado de: https://arxiv.org/pdf/1504.08083.pdf

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

[12] 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.

[13] 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 ingreso: 14/09/2020)

[14] Repositorio: TensorFlowObjectDetectionModel. Recuperado.de: de acceso https://github.com/tensorflow/models/tree/master/research/object_detection(fecha : 2/12/2020)

[15] Anaconda | The World's Most Popular Data Science Platform. Recuperado de: https://www.anaconda.com/

[16] Librerías de Python para ML. Recuperado de: https://www.iartificial.net/librerias-de-python-para-machine-learning/#Librerias_de_Python_para_Deep_Learning (fecha de acceso: 3/12/2020)

[17] Repositorio: LabelImg. Recuperado de: https://github.com/tzutalin/labelImg(fecha acceso: 14/09/2020)

[18] Shorten, Connor; Khoshgoftaar, Taghi M. (2019). "A survey on Image Data Augmentation for Deep Learning". Mathematics and Computers in Simulation. springer. 6: 60. doi:10.1186/s40537-019-0197-0.