UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DESARROLLO DE UN MOTOR DE RECONOCIMIENTO DE IMÁGENES INTELIGENTE PARA FOTOS DE PRUEBAS DE DESPACHO EN ÚLTIMA MILLA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

FRANCO ANTONIO MIRAUDA LIZANA

PROFESOR GUÍA:
Alfredo Vega Curihuinca

MIEMBROS DE LA COMISIÓN:
Francisco Rivera Serrano
Carlos Navarro Clavería

SANTIAGO DE CHILE
2023

DESARROLLO DE UN MOTOR DE RECONOCIMIENTO DE IMÁGENES INTELIGENTE PARA FOTOS DE PRUEBAS DE DESPACHO EN ÚLTIMA MILLA

El *Deep Learning*, corresponde a un conjunto de algoritmos muy utilizados en la actualidad, que han sufrido una mejora constante a través del tiempo, sobre todo, para resolver problemas ligados al procesamiento de lenguaje natural, como la predicción de texto; y también ligados al procesamiento de imágenes, como clasificación, detección y segmentación.

El desarrollo de esta memoria, esta enfocado al área de procesamiento de imágenes, específicamente, en el entrenamiento e implementación de modelos de clasificación multi-etiqueta, los cuales permiten identificar varias clases en una imagen, además, de modelos de detección de objetos, los cuales permiten identificar la ubicación y la clase del objeto.

En este trabajo, se propone un motor de reconocimiento de imágenes inteligente, el cual, permite evaluar la calidad de los despachos de última milla de la empresa Falabella, a través de la revisión de ciertos elementos en la fotografía, la cual se toma al momento de entregar un producto. Para lograr lo anterior, se realizó un *benchmark* entre herramientas basadas en Google Cloud Platform (GCP), y modelos entrenados en base a *transfer learning* utilizando lenguaje Python y la librería PyTorch.

Cabe destacar, que el motor de reconocimiento de imágenes se realizó de manera exitosa, dando mejores resultados, los modelos desarrollados *in house* en contraste con las herramientas de GCP. Lo anterior, principalmente debido al mayor control que se tuvo en el entrenamiento de los modelos *in house*.

La estructura del presente informe, comienza con la identificación y formulación del problema. Posteriormente, se hace mención al objetivo general y a los objetivos específicos. Luego, se explican las tareas de clasificación multi-etiqueta, detección de objetos, *Deep Learning*, algunas técnicas para evitar el sobre-ajuste durante el entrenamiento, las métricas utilizadas para evaluar los modelos y, finalmente, las redes implementadas para resolver el problema. A continuación, se describe la metodología desarrollada, se muestran los resultados técnicos y económicos, incluyendo el análisis de cada uno de ellos y, por último, se realiza una conclusión de todo el trabajo realizado.

Agradecimientos

Agradezco enormemente a Alfredo, Aldo y Franco por apoyarme y guiarme en todo el proceso de desarrollo del proyecto. Le doy las gracias también, a todos los que conforman y han sido parte del equipo de Advanced Analytics, por sus consejos, compañía y hermandad dentro de Falabella. Por último, agradezco a mi pareja y familia, por contenerme y darme ánimos en los momentos más difíciles.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Identificación y Formulación del Problema

El trabajo de memoria esta focalizado en la validación de entrega de productos en última milla, es decir, el trayecto final en el que se entrega directamente el producto al cliente, enmarcándose en el área de *Supply Chain* (conjunto de procesos que se llevan a cabo desde la fabricación del producto hasta el cliente final) de la empresa Falabella. Para validar la entrega, el despachador debe escanear un código QR obtenido por el cliente al momento de comprar. Dada la imposibilidad de realizar lo anterior para el 100 % de las entregas, debido a aspectos tecnológicos y culturales (falta de conocimiento para obtener código QR, entrega de producto en la recepción de un edificio, o bien, que el receptor no sea el cliente directo), existe un segundo proceso de validación en donde se obtiene información relevante de la persona que recibe el producto (nombre y RUT) y se saca una foto. Dicha prueba de despacho (PoD, por sus siglas en inglés) se guarda en una base de datos como respaldo ante cualquier problema de entrega y, también, para tener un seguimiento de la calidad de fotos que genera cada transportista.

Dado que las fotos son un respaldo, el negocio (torre de control de transporte) conformado por un equipo encargado de asegurar el cumplimiento del compromiso de entrega y experiencia de compra de los clientes; ha definido una serie de criterios para determinar cuándo una foto es válida, dentro de los cuales están: que aparezca el producto, que aparezca la etiqueta del producto, que no aparezca la cara del receptor, entre otras consideraciones. Hasta el momento, para verificar que esas fotos estén correctas, hay un equipo de personas que se encargan de escoger aleatoriamente fotos de la base de datos, las cuales son revisadas, con los criterios mencionados anteriormente, de manera manual. Ya que el numero de falsas entregas, corresponde al 17 % de la totalidad de los reclamos; el proceso, requiere de un alto costo en horas-hombre; la metodología para definir si una foto está bien o mal, no esta correctamente definida; y además, es poco eficiente (se logra etiquetar solo el 10 % del total de entregas), se propone elaborar un método inteligente y automático, mediante técnicas de inteligencia artificial para reconocimiento de imágenes y clasificación, con la finalidad de determinar si una foto es válida de acuerdo con los criterios del negocio. Es importante mencionar, que para ir verificando que el modelo propuesto funcione correctamente, no se debe eliminar totalmente el proceso manual. No obstante, aun así se reducirá considerablemente el costo de horas-hombre y será un proceso más eficiente. Lo expuesto anteriormente, se observa en la figura 1.1.

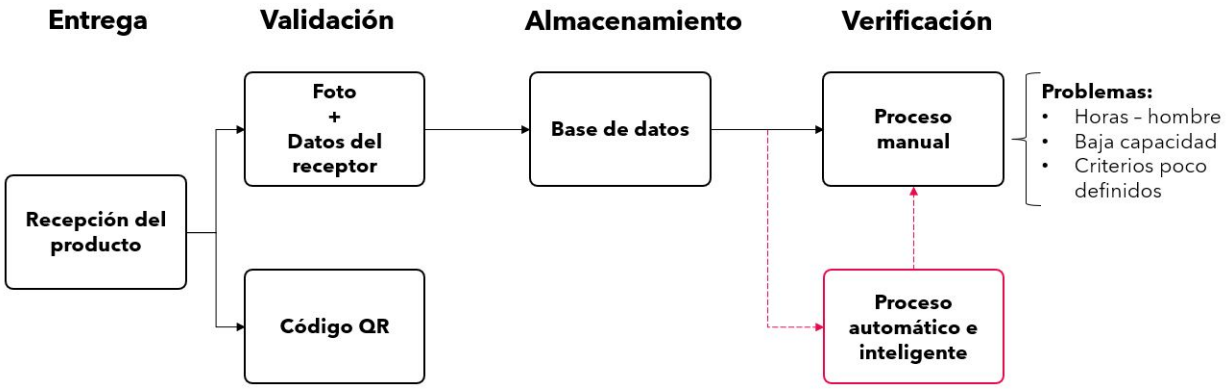

Figura 1.1: Proceso llevado a cabo desde la entrega del producto a la verificación de la base de datos. En rojo se muestra el flujo propuesto. Además se identifican los principales problemas del proceso manual.

La figura 1.2, muestra un resumen de lo que se espera del modelo inteligente. En general, se tiene una foto de entrada, que pasa por el modelo (representado mediante una caja negra) el cual genera como salida, los criterios identificados (con valor 1). Estos criterios, corresponden al producto (caja o paquete), el número de domicilio, la etiqueta del producto, la cara del receptor y un contexto, que hace referencia a qué porcentaje de toda la fotografía es ocupada por el producto (si el contexto se encuentra en un rango adecuado, entonces el valor es igual a 1). Finalmente, en base a los criterios, se calcula un "score" que corresponde a una nota con la cual se evalúa la fotografía. Entre mayor el "score", mejor es la fotografía.

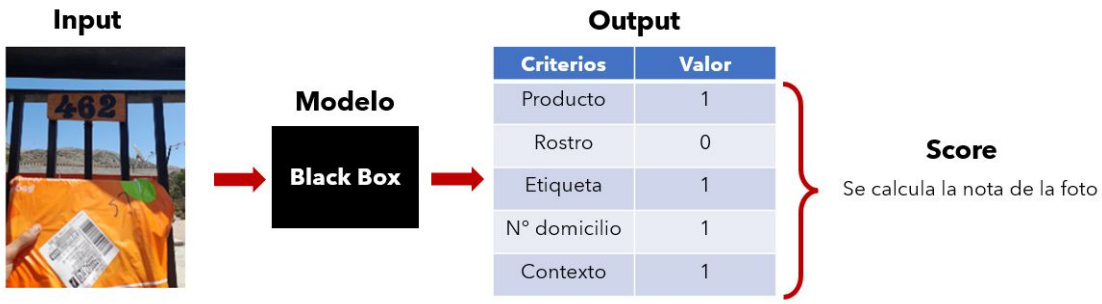

Figura 1.2: Diagrama que resume el procedimiento esperado por el modelo inteligente.

1.2. Objetivos de la memoria

El objetivo general del trabajo, es entregar una herramienta inteligente, que permita reducir las horas-hombre, que sea eficiente y con alta capacidad de etiquetado, el cual logre evaluar las fotografías en base a criterios específicos y de esa manera, que el negocio, mediante su implementación, pueda mejorar la base de datos y además, que le ayude a tener un seguimiento de la calidad de fotos que genera cada transportista.

Respecto al desarrollo del trabajo, y en cuanto a los objetivos específicos, se pretende como primera instancia, evaluar la herramienta Vision AI de Google Cloud Platform (GCP), debido a su facilidad de integración con los sistemas de Falabella y además con la finalidad de pilotear y validar con el negocio lo antes posible. Dentro de los productos de Vision AI se encuentra la Vision API, que permite detectar objetos mediante un modelo pre-entrenado, entregando las etiquetas correspondientes. Otro de los productos ofrecidos por GCP es el AutoML Vision, que mediante un proceso automático, encuentra el mejor modelo para clasificación en base a etiquetas definidas por el usuario. Luego de realizado el *baseline* mediante GCP, se buscará desarrollar un *benchmark* basándose en el estado del arte en clasificación de imágenes de etiquetas múltiples y detección de objetos, para poder comparar la herramienta de GCP con estos modelos generados manualmente y encontrar, finalmente, la mejor solución al problema.

Capítulo 2

Marco Teórico y Estado del Arte

Para poder identificar los criterios en cada fotografía, se dividió el problema en dos subproblemas. Uno de los cuales se resuelve con un clasificador multi-etiqueta y, otro, con un detector de objetos. Lo anterior, es debido a que para calcular el contexto, se utilizó la capacidad del detector, para identificar y localizar el paquete mediante un *bounding box* (cuadro delimitador).

A continuación, se explicarán ciertos conceptos relevantes, para comprender, en qué consiste un clasificador multi-etiqueta y un modelo de detección de objetos.

2.1. Marco teórico

2.1.1. Clasificación multi-etiqueta y detección de objetos

La principal diferencia entre la clasificación de una sola etiqueta frente a la clasificación de etiquetas múltiples, es que en esta última, se tiene un conjunto de etiquetas, en la cual se indican los distintos objetos que aparecen en una imagen. Como se indica en [1], si es que se utiliza clasificación de una etiqueta para la figura 2.1, se podría identificar solo una de las clases que aparecen en la imagen, por ejemplo, podría ser el mar. En cambio, al utilizar clasificación de etiquetas múltiples, el resultado podría ser, mar, arboles, cielo, etc.

Figura 2.1: Imagen de ejemplo para diferenciar una etiqueta de muchas etiquetas. Extraído de [1]

Por otro lado, como se menciona en [2], la detección de objetos, no solo reconoce las distintas categorías de estos objetos, también predice la locación de cada objeto mediante un *bounding box*. Como se puede ver en la figura 2.2.

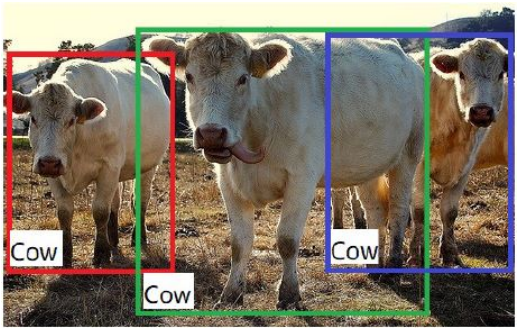

Figura 2.2: Imagen que muestra el objetivo del detector de objetos. El cual consiste en identificar las distintas clases (vaca en este caso) y la localización de cada una de ellas mediante *bounding boxes*. Extraído de [2]

2.1.2. Deep learning

Los métodos del estado del arte, que se utilizan para resolver problemas de clasificación y detección, son principalmente algoritmos de *Deep Learning*, el cual corresponde a un subconjunto de *Machine Learning* (ver imagen 2.3 en donde ML y DL son las siglas de *Machine Learning* y *Deep Learning*, respectivamente), en el que el mismo algoritmo es capaz de identificar y representar las características que encuentre necesarias para lograr el objetivo final, mientras que en *Machine Learning* es el humano el que define esas características.

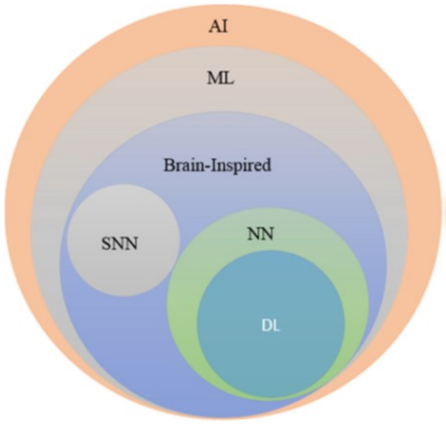

Figura 2.3: The taxonomy of AI. Extraído de [3]

Deep Learning consiste en el uso de algoritmos basados en redes neuronales profundas, es decir, varias capas (conjuntos de perceptrones) unidas entre ellas. Cabe destacar que un perceptrón es un elemento básico, que corresponde a una combinación lineal entre ciertas entradas con algunos pesos (valores que se deben encontrar mediante el entrenamiento y que permiten resolver el problema deseado de la mejor forma posible), pasada por una función de activación.

Una red neuronal muy utilizada en procesamiento de imágenes es la red neuronal convolucional (CNN por sus siglas en inglés). Como se menciona en [4], una CNN es un algoritmo de aprendizaje profundo que puede tomar una imagen de entrada, asignar importancia (pesos y sesgos aprendibles) a varios aspectos/objetos en la imagen y ser capaz de diferenciar uno de otro. Además, es capaz de capturar con éxito las dependencias espaciales y temporales en una imagen mediante la aplicación de filtros relevantes. En la figura 2.4 se puede ver una red neuronal convolucional que presenta capas convolucionales, de *pooling* y *fully-connected*. Las capas convolucionales corresponden a una ponderación de píxeles, en cambio, la operación de *pooling*, consiste en aplicar distintas relaciones matemáticas entre píxeles. Por último, la capa *fully-connected* permite disminuir la dimensionalidad de las otras capas para finalmente obtener la salida deseada.

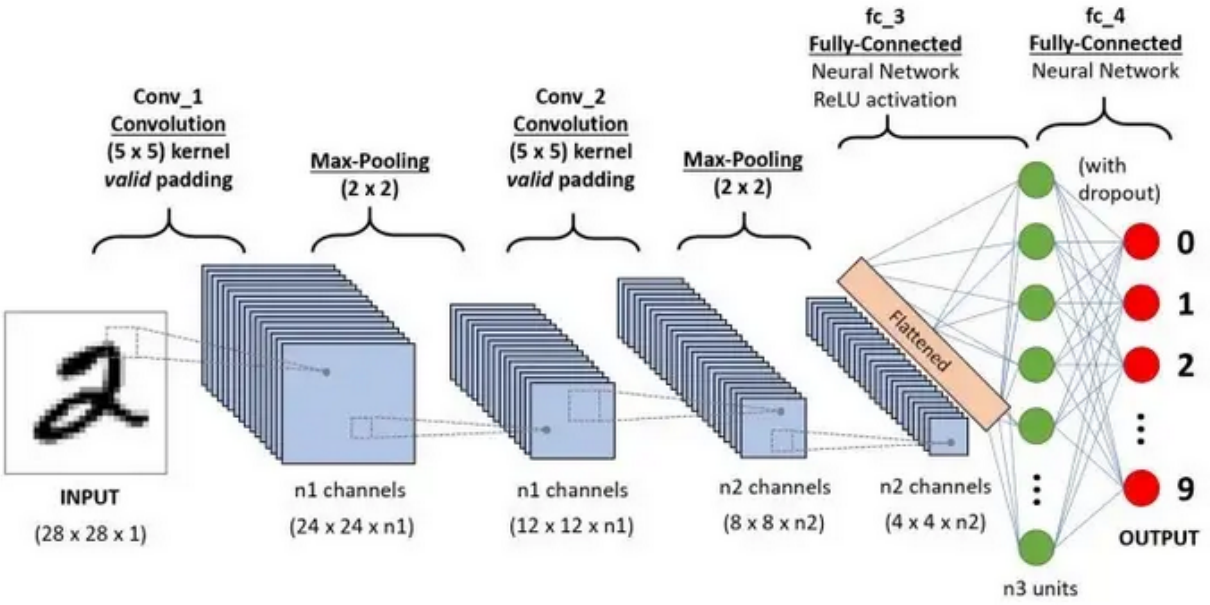

Figura 2.4: CNN sequence. Extraído de [4].

2.1.3. Transfer learning

Como se menciona en [5], la transferencia de aprendizaje o *transfer learning*, busca mejorar el rendimiento de un aprendiz, en el dominio de alguna actividad objetivo, traspasando el conocimiento obtenido, en otra actividad diferente pero relacionada. Un ejemplo de lo anterior, es la facilidad con la que una persona que sabe tocar el violín, puede aprender a tocar el piano, debido a que ambos, son instrumentos y, por ende, se basan en conocimientos parecidos.

En el caso de los modelos de *machine learning*, este método, se puede implementar mediante la compartición de pesos, es decir, aprovechar el aprendizaje adquirido por una red pre-entrenada, congelando varias de sus capas y afinando (entrenando) solo las últimas capas, o bien, se puede entrenar toda la red, pero inicializando los pesos con los valores obtenidos del modelo pre-entrenado, de manera de poder resolver un nuevo problema, pero relacionado con el anterior. Un ejemplo de esto, es realizar una transferencia de aprendizaje a un mode-

lo utilizado para clasificar perros y gatos, con el objetivo de lograr clasificar autos y camiones.

Los principales beneficios de implementar *transfer learning* es, por un lado, ahorrar en costos computacionales (se reduce el tiempo de entrenamiento) y, por otro lado, mejorar el desempeño del modelo.

2.1.4. Técnicas para evitar sobreajuste

El sobre-ajuste u *overfitting*, ocurre cuando un modelo no es capas de generalizar, es decir, cuando el modelo obtiene muy buenos resultados con la data de entrenamiento, pero genera un alto grado de error, al procesar la data nueva. Para evitar lo anterior, existen algunas técnicas como las que se mencionan a continuación.

2.1.4.1. Data augmentation

El *data augmentation*, se ha convertido en una parte fundamental para obtener modelos de *deep learning* exitosos, ya que es una forma eficaz de mejorar la suficiencia y diversidad de los datos de entrenamiento. Una de las formas más básicas de implementación de este procedimiento, consiste en aplicar transformaciones a las imágenes, como rotación, traslación, recortes, entre otras [6]. La tabla 2.1 muestra distintos métodos para manipular las imágenes.

Tabla 2.1: Tabla en donde se muestran distintos métodos para manipular imágenes y la descripción respectiva. Extraído de [6].

Métodos	**Descripción**
Voltear	Voltear imágen de manera vertical, horizontal o ambas.
Rotación	Rotar imágen en cierto ángulo.
Escalamiento	Incrementar o reducir el tamaño de la imágen.
Inyección de ruido	Agregar ruido a la imágen.
Espacio de color	Cambiar los canales de color de la imágen.
Contraste	Cambiar el contraste de la imágen.
Nitidez	Modificar la nitidez de la imágen.
Traslación	Mover imágen de manera vertical, horizontal o ambas.
Recorte	Recortar una sub-región de la imágen.

2.1.4.2. Early stopping

El *early stopping*, consiste en un método de regularización, en que se detiene el entrenamiento al encontrar el modelo que posee menor error de validación. En la figura 2.5 se muestra un gráfico en que se ve la curva de error de entrenamiento y de validación. Se puede ver que a medida que aumenta el número de épocas de entrenamiento, el modelo va encontrando funciones mucho más complejas que permiten que el error vaya decreciendo constantemente, no obstante, el modelo se va sobre-ajustando, haciendo que el error de validación comience a aumentar.

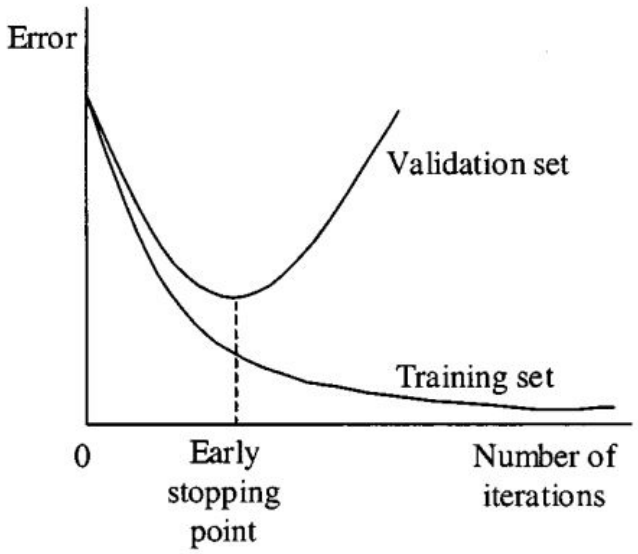

Figura 2.5: Gráfico en donde se ven las curvas de error, obtenidas con el conjunto de entrenamiento y validación, a través de las épocas. Además, se observa el punto en donde se detiene el entrenamiento para que el modelo no se sobre-ajuste. Extraído de [7].

2.1.5. Automated machine learning

Para comprender en qué se fundamenta la herramienta de AutoML de GCP, hay que entender qué es AutoML. De acuerdo a [8], AutoML o *automated machine learning* es una prometedora solución para construir sistemas de *Deep Learning* sin la necesidad de intervención de un ser humano, es decir, lo que se busca esencialmente es generar un procedimiento automático que consiste en 4 etapas principales: preparación de los datos, ingeniería de características (selección, extracción y construcción), una generación de modelos ya sea tradicionales, los cuales se optimizan buscando los mejores hiperparámetros, o modelos de redes neuronales profundas, los que se optimizan encontrando la mejor arquitectura; y finalmente, hay una estimación del modelo más adecuado según el problema que se desee resolver.

2.1.6. Métricas

2.1.6.1. Matriz de confusión

La matriz de confusión se representa en base a una tabla, la cual permite relacionar y visualizar, las predicciones realizadas por el modelo, con los valores reales de los datos, pudiendo obtener las métricas de *Accuracy*, *Precision*, *Recall*, entre otras. La figura 2.6 muestra una matriz de confusión, en donde cada cuadrante indica la relación entre el valor real (*True class*) con el valor predicho (*Hypothesized class*). De esa manera, se obtienen los verdaderos positivos (*True Positives* o TP), que son datos en que el modelo identifica correctamente a la clase positiva; los Falsos positivos (*False Positives* o FP), que son datos que el modelo indica que pertenecen a la clase positiva, pero en realidad son de la clase negativa; los falsos negativos (*False Negatives* o FN), que son datos que el modelo indica que pertenecen a la clase negativa pero en realidad son de la clase positiva; y finalmente, los verdaderos negativos (*True Negatives* o TN) que son datos en que el modelo identifica correctamente a la clase

negativa.

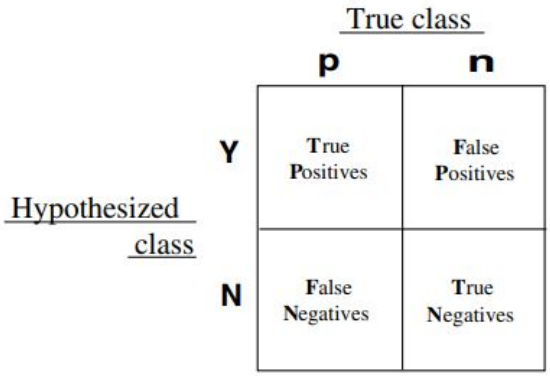

Figura 2.6: Matriz de confusión. Extraído de [9].

2.1.6.2. Accuracy

Esta métrica mide el porcentaje de acierto del modelo, frente a todos los casos, y se calcula como:

$$accuracy = \frac{TP + TN}{P + N} \tag{2.1}$$

Con $P = TP + FN$ y $N = FP + TN$.

2.1.6.3. Precision

Métrica que muestra qué porcentaje de las predicciones en que el modelo identifica a la clase positiva, pertenecen realmente a esa clase. Esta métrica, se calcula de la siguiente manera:

$$precision = \frac{TP}{TP + FP} \tag{2.2}$$

2.1.6.4. Recall o True positive rate

Métrica que muestra qué porcentaje de casos pertenecientes a la clase positiva, el modelo logra identificar correctamente. Esta métrica, se calcula como:

$$recall = \frac{TP}{P} \tag{2.3}$$

2.1.6.5. False positive rate

Métrica que muestra qué porcentaje de casos pertenecientes a la clase negativa, el modelo se equivoca. Esta métrica, se calcula como:

$$FPR = \frac{FP}{N} \tag{2.4}$$

2.1.6.6. F1-score

Métrica que corresponde a la media harmónica entre *precision* y *recall*. Se calcula de la siguiente forma:

$$F1 - score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{2.5}$$

2.1.6.7. Curvas ROC y PR

Las curvas ROC (*Receiver Operating Characteristic*) se obtienen graficando la tasa de verdaderos positivos (*recall*), con respecto a la tasa de falsos positivos (*false positive rate*) para distintos valores de umbrales. Esa curva en base a su forma y el área bajo la curva (AUC por sus siglas en ingles), permite comparar diferentes modelos. Entre más cercano a 1 esté el valor del área, entonces significa que el modelo es mejor.

Las curvas PR (*Precision Recall*), tal como su nombre indica, se obtienen graficando la métrica de *precision* con respecto al *recall* para distintos valores de umbrales. Esa curva, al igual que el caso anterior, en base a su forma, y el área bajo la curva o en este caso el *Average Precision* (AP), permite medir el rendimiento de un modelo. Entre más cercano a 1 esté el valor del AP, mejor será el modelo. A diferencia de la curva ROC, este gráfico se utiliza cuando se tienen clases desbalanceadas, debido a que no se considera el número de verdaderos negativos.

Es importante mencionar que encontrando el máximo F1-score en la curva PR y por otro lado, maximizando la diferencia entre la tasa de verdaderos positivos con respecto a la tasa de falsos positivos, en la curva ROC, es posible obtener el mejor umbral para ambos casos. Con eso, se puede definir los umbrales de predicción del modelo.

2.1.7. Intersection over union (IoU)

Corresponde a un valor que se utiliza para la detección de objetos, el cual permite cuantificar el grado de sobre-posición entre dos *bounding boxes* y se obtiene dividiendo la intersección entre el *bounding box* verdadero con el *bounding box* predicho, por la unión de esos *bounding boxes*.

En base a este valor, es posible calcular ciertas métricas como por ejemplo el *Average Precision* (AP). En este caso, la siguiente notación $Ap@0.5 : 0.05 : 0.95$ significa promediar los distintos valores de AP para cada IoU, desde 0.5 hasta 0.95 aumentando en pasos de 0.05. Al igual que el AP, es posible obtener en base al valor de IoU, el *Average Recall* (AR). Entender lo anterior será útil para analizar las tablas 4.2 y 4.3.

Existen variaciones para las dos métricas anteriores como es considerar el tamaño del objeto, en donde *small* significa que el objeto cubre un área menor a 32^2 píxeles; *medium* que cubre un área entre 32^2 y 96^2; y *large* que cubre un área mayor a 96^2. También se considera el número de detecciones por imagen: 1, 10 o 100.

2.2. Estado del Arte

2.2.1. Residual Network (ResNet)

ResNet es una CNN que fue desarrollada por Kaiming He en el año 2015 (para más detalles ver paper [10]), con el objetivo de obtener redes muy profundas que no sufrieran del llamado desvanecimiento del gradiente [3]. El desvanecimiento del gradiente puede tener como efecto, que los pesos no se sigan actualizando. Para poder mitigar lo anterior, se desarrolló el bloque de aprendizaje residual (ver figura 2.7), el cual posee conexiones directas, en donde la entrada de una capa esta conformada por un término adicional correspondiente a la salida de una capa anterior. Esto permite, que la red pueda aprender la función identidad si es que así lo quisiera.

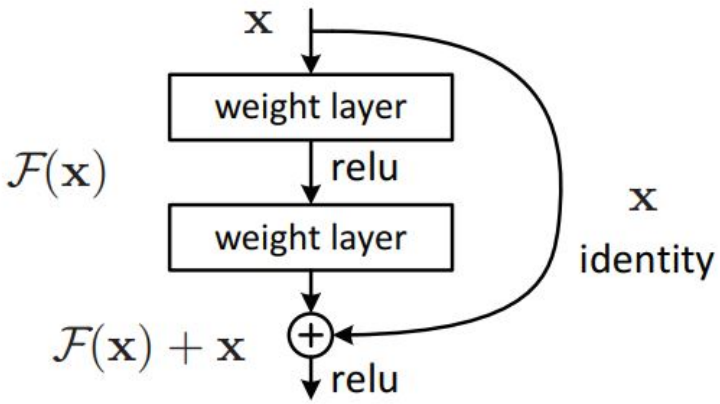

Figura 2.7: Bloque residual. Extraído de [10].

2.2.2. ResNeXt

ResNeXt es una CNN desarrollada en 2016 y que, a diferencia de la ResNet, posee una nueva dimensión llamada "cardinalidad" o "cardinality". De acuerdo a [11], la "cardinalidad" corresponde al tamaño del conjunto de transformaciones, lo que permite crear redes más profundas y anchas, mejorando el *accuracy* en clasificación y obteniendo mejores resultados que la ResNet. La figura 2.8, compara un bloque de la ResNet y un bloque de la ResNeXt, en donde se puede ver claramente que la complejidad se mantiene similar, pero obteniendo mejores resultados con la estructura de la derecha.

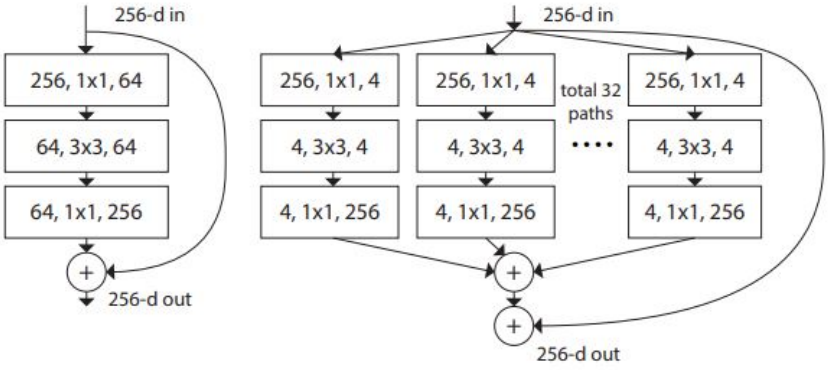

Figura 2.8: Comparación entre un bloque de ResNet (izquierda) y un bloque de ResNeXt con cardinalidad 32 (derecha). Extraído de [11].

2.2.3. Faster R-CNN

Corresponde a un detector de dos etapas (ver figura 2.9). Su primera etapa consiste en una red de propuesta de región (RPN por sus siglas en ingles), la cual propone candidatos de *bounding boxes*, y la segunda etapa, la red extrae características usando una capa de región de interés o *Region of Interest pooling layer*, para cada candidato de *bounding box* y, luego, se pasa por una capa densa para realizar la clasificación y la regresión de *bounding box* [12].

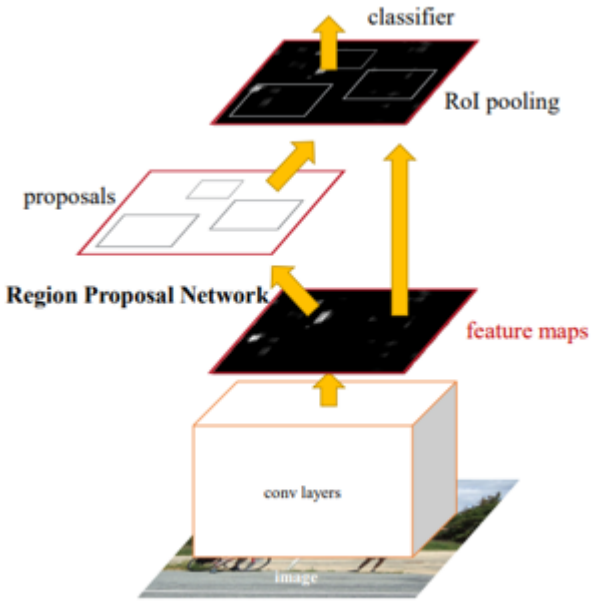

Figura 2.9: Estructura de una red Faster R-CNN. Extraído de [13].

Capítulo 3

Desarrollo del Motor de Reconocimiento de Imágenes

3.1. Determinación de los criterios

La primera etapa consistió en identificar, en conjunto con el negocio, los criterios más relevantes para determinar si una fotografía cumple con los requisitos de validación de una entrega. De esa manera, se concluyó, que los criterios que se adaptan mejor a los requerimientos son:

- Identificación del paquete.
- Identificación de la etiqueta del paquete.
- Identificación del número de domicilio en la fachada de un edificio o casa.
- Que en la fotografía no aparezca la cara del receptor del producto, ya que es data sensible que puede generar consecuencias legales.
- Determinar la relación entre el área que ocupa el paquete, respecto al área total de la fotografía (criterio denominado "contexto"). Si es que el valor de la relación es insignificante o por el contrario, de gran magnitud, entonces la fotografía pierde la capacidad para validar la entrega de un producto.

La figura 3.1 muestra cada uno de los criterios mencionados, en una fotografía de ejemplo.

Figura 3.1: Fotografía en donde se enmarca con un cuadro rojo el criterio de "numero de domicilio", en un cuadro blanco (se descartó la cara por ser data sensible) se identifica el criterio de "rostro", el cuadro azul enmarca al "paquete" y el cuadro negro enmarca la "etiqueta del producto". Finalmente se puede deducir que el valor de "contexto" es cercano a $0,5$.

3.2. Función de score

Una vez identificado los criterios, se desarrolló una función de "score", que permite generar una nota para cada fotografía, y de ese modo, poder evaluarlas. La ecuación 3.1, muestra la función implementada.

$$score(B_i) = \sum_{i=1}^{5} w_i \cdot B_i \tag{3.1}$$

Donde w_i indica el peso para cada criterio y B_i corresponde a un valor binario (0 o 1), en donde si es que la confianza del modelo para un criterio, es mayor o igual a cierto umbral predefinido, entonces B_i es 1 y, en caso contrario, es 0. Lo anterior, se observa en la ecuación 3.2, con P_i la confianza del modelo para el criterio i y μ_i el umbral definido para ese criterio.

$$B_i(P_i) = \begin{cases} 1 & si \quad P_i >= \mu_i \\ 0 & si \quad P_i < \mu_i \end{cases} \tag{3.2}$$

Para el caso del "contexto", el valor binario asociado, depende del rango en el que se encuentre (como se puede ver en la ecuación 3.3) en donde μ_{inf} corresponde a un límite inferior y μ_{sup} es un límite superior.

$$B(ctx) = \begin{cases} 1 & si \quad \mu_{inf} <= ctx <= \mu_{sup} \\ 0 & si \quad ctx < \mu_{inf} \quad \vee \quad ctx > \mu_{sup} \end{cases} \tag{3.3}$$

Es importante notar, que para obtener el "contexto", se calcula la unión de todos los *bounding boxes*, identificados en la fotografía, y se le resta la suma total de las áreas entre todas las combinaciones de *bounding boxes*.

Los pesos w_i, son regulables (se pueden escoger dependiendo de las necesidades del negocio) y determinan la importancia del criterio i, por lo que si un criterio se considera más relevante como, por ejemplo, que aparezca el paquete en la fotografía, entonces el valor del peso asociado a ese criterio, es mayor. Además, se debe cumplir que $\sum_{i=1}^{5} w_i = 1,1$. Con lo anterior, se tiene que el valor de "score" va de 0 a $1,1$, en donde el valor base es 1, pero existe un beneficio de $0,1$ cuando aparece el número de domicilio en la fotografía. Esto, ya que en ciertas situaciones como, por ejemplo, cuando se realiza una entrega en el departamento de un edificio, al transportista se le dificulta obtener fotografías con ese criterio, debido a que el número de domicilio en esos casos, se encuentran en la fachada del edificio.

3.3. Revisión de data

Es importante mencionar, que si bien la data entregada por el negocio, ya estaba etiquetada de manera binaria (1 si aparece la clase y 0 si no aparece), una cantidad considerable de fotografías, no estaba bien etiquetada, o bien, presentaban ciertas ambigüedades como, por ejemplo, la identificación de un rostro, cuando en verdad no se lograba distinguir las facciones, entre otros aspectos. Además, existía un desbalance importante entre las clases. Debido a lo anterior, se seleccionó solo un subconjunto de datos, tomando en consideración clases balanceadas, y se tuvo que hacer una segunda revisión de la data, para cerciorarse de entrenar los modelos con los datos adecuados.

3.4. Herramienta Vision AI

Para poder desarrollar el *baseline*, se descompuso el problema en 3 sub-problemas. El primero, corresponde a la implementación de un detector de objetos, para determinar cuándo hay un producto en la imagen y, además, poder calcular el contexto. El segundo, es usar un detector de caras y, el tercero, corresponde a la implementación de un clasificador multi-etiqueta para determinar cuándo está presente la etiqueta del producto y/o el número de domicilio.

Dado que para elaborar el primer modelo, se tuvo que utilizar productos de Google Cloud Platform, los sub-problemas anteriores, se adecuaron a esta plataforma. De esa manera, mediante la Vision API, se puede implementar tanto un detector de objetos como de caras. Luego, con el uso de AutoML Vision, se pudo entrenar un modelo de clasificación multi-etiquetas para detectar la etiqueta del producto y el número de domicilio. Ya que AutoML corresponde a un proceso automático de entrenamiento, se tuvo que utilizar una base de datos con imágenes etiquetadas, las cuales fueron facilitadas por el negocio. Es importante destacar, que estas imágenes no presentan como etiqueta la ubicación de los objetos a detectar, es decir, solo presentan un etiquetado binario para identificar si está o no el elemento en la fotografía. La figura 3.2, muestra un ejemplo del etiquetado manual.

Paquete	Etiqueta	Cara	N° Domicilio
1	1	0	1

Figura 3.2: Fotografía con etiquetado binario para cada criterio.

En la figura 3.3 se muestra un diagrama de la metodología llevada a cabo para obtener el *baseline*. Se puede ver, que la primera etapa consistió en el entrenamiento del modelo de clasificación multi-etiquetas mediante AutoML, en donde se consideró fotografías pertenecientes a todo el espectro de combinaciones de las distintas clases (etiqueta del producto y nº de domicilio), es decir, fotografías con ambas clases; fotografías con solo una de las clases y fotografías sin esas clases, con el objetivo de que el modelo pueda generalizar correctamente. La segunda etapa, fue la implementación por un lado del modelo entrenado y, por otro lado, de la herramienta de Vision API, la cual consiste en modelos pre-entrenado listos para su ejecución. Finalmente, para evaluar el modelo, se obtuvieron las métricas de *accuracy*, *precision*, *recall* y F1-score. Es importante mencionar, que dado que la herramienta Vision API solo entrega predicciones por sobre un umbral de 0,5, no fue posible obtener curvas ROC y de PR representativas.

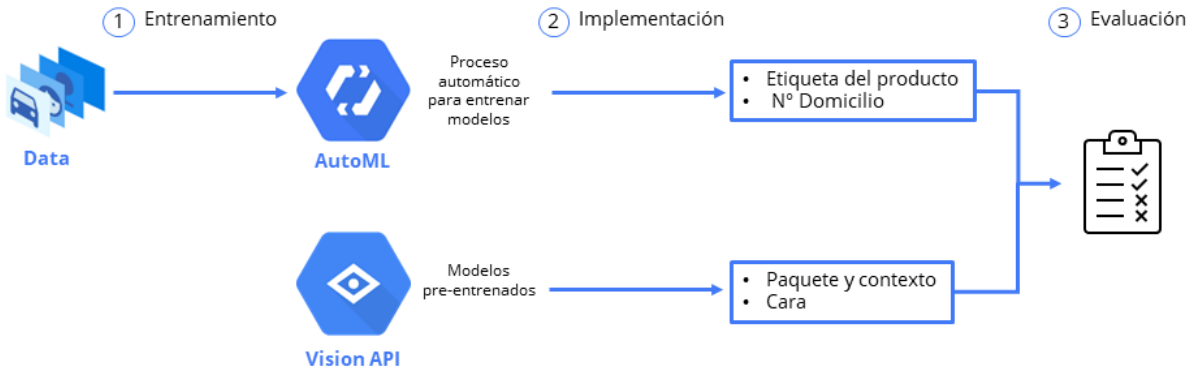

Figura 3.3: Diagrama de la metodología llevada a cabo para obtener el modelo en base a la herramienta GCP.

3.5. Modelo in-house

Para el desarrollo del modelo *in-house*, se decidió dividir el problema en dos sub-problemas. El primero, corresponde a la implementación de un detector de objetos, que al igual que en

el *baseline*, su función es identificar el paquete en la imagen y, además, calcular el contexto, de acuerdo al *bounding box* respectivo. El segundo, corresponde a la implementación de un clasificador multi-etiqueta para determinar cuándo esta presente la etiqueta del producto y/o el número de domicilio y/o la cara del receptor.

En la figura 3.4, se muestra las etapas llevadas a cabo para desarrollar el modelo. El primer paso, consistió en la elaboración del código para implementar cada sub-problema, el cual fue escrito utilizando lenguaje Python y la librería PyTorch; librería de *machine learning* de código abierto, muy utilizada para procesamiento de imágenes. El segundo paso, fue entrenar el modelo en base al método de *transfer learning* y utilizando la data entregada por el negocio. Finalmente, se escribió un código que une cada modelo, de manera de trabajar cada predicción en conjunto y poder obtener el *score* de la fotografía.

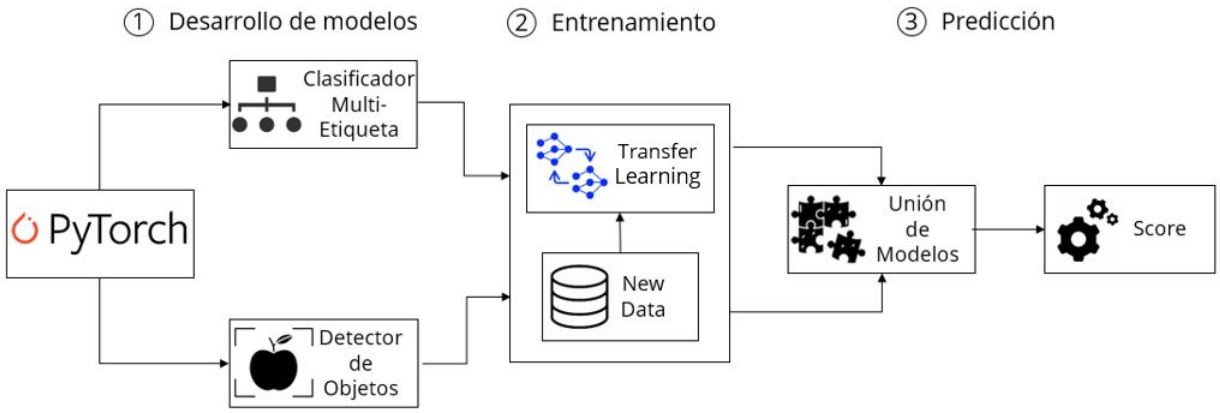

Figura 3.4: Diagrama de la metodología llevada a cabo para obtener el modelo *in house*.

3.5.1. Manipulación de data

En este caso, para poder entrenar el clasificador multi-etiqueta, se utilizó la data binaria, en donde se consideró fotografías pertenecientes a todo el espectro de combinaciones de las distintas clases (etiqueta del producto, nº de domicilio y cara del receptor). Por otro lado, para poder entrenar el detector de objetos, se tuvo que realizar un *workshop*, en donde se le enseñó al negocio a cómo etiquetar mediante *bounding boxes* y, para ello, se utilizó la herramienta Make Sense la cual es de libre acceso; no requiere de instalación y además es segura en cuanto a los datos.

3.5.2. Clasificador multi-etiquetas

El desarrollo del clasificador multi-etiqueta, se basó en la implementación de *transfer learning*, en dos redes del estado del arte para clasificación: ResNet-101 y ResNeXt-101, con el objetivo de realizar una comparación entre estos dos modelos. El entrenamiento se realizó, teniendo en cuenta el método de *early stopping*, para evitar sobre-ajuste y, además, se implementaron dos configuraciones distintas de *data augmentation*. En base a lo anterior, se elaboraron varios experimentos, que posteriormente fueron comparados tomando en conside-

ración: las métricas de *accuracy*, *precision*, *recall*, *F1-score* y curvas ROC y PR.

Los experimentos realizados se observan en la tabla 3.1. Es importante mencionar, que además de cambiar el *backbone*, se trabajó con dos datas distintas (data 1 y data 2). Lo anterior, ya que los resultados del modelo para la data 1, muestran un bajo rendimiento del modelo al identificar el nº de domicilio, en fotografías en que solo aparece la fachada de la casa. Esto se intentó solucionar, aumentando la cantidad de fotografías en donde no hubiera nº de domicilio. Además de lo anterior, se trabajó con dos transformaciones distintas. La transformación 1 considera voltear la imagen de manera horizontal y vertical, aleatoriamente con probabilidad de 0,5. Por otro lado, la transformación 2, considera, además de voltear la imagen verticalmente (al igual que en el caso anterior), agregar rotaciones aleatorias, de 20 grados, y traslación con una relación de 0,05 en cada lado.

Tabla 3.1: Experimentos llevados a cabo y sus cambios respectivos.

Modelo	**Data**	***Backbone***	**Transformación**
1	1	ResNet-101	1
2	1	ResNet-101	2
3	1	ResNeXt-101	2
4	2	ResNeXt-101	2

3.5.3. Detector de objetos

Basándose en el código que aparece en [14], se logró desarrollar el detector de objetos, mediante *transfer learning*, aplicado sobre el modelo Faster R-CNN. Dado los buenos resultados obtenidos con esa configuración de hiper-parámetros, no se siguió iterando con nuevos experimentos, debido a los altos costos de entrenamiento. No obstante, al igual que el clasificador multi-etiquetas, se utilizó *data augmentation*, que en este caso, consistió en voltear la imagen de manera horizontal y vertical con probabilidad de 0,5 cada una. Además, se tomó en cuenta el método de *early stopping*, para evitar sobre-ajuste.

Para poder evaluar los resultados del modelo, se utilizó la métrica IoU en el conjunto de validación y, además, se implementaron las métricas de *accuracy*, *precision*, *recall*, *F1-score* y curvas ROC y PR.

3.6. Benchmark

Una vez desarrollado el *baseline*, y obtenido los modelos *in-house*, se hizo una comparación considerando las métricas de *accuracy*, *precision*, *recall* y *F1-score*. Además, se calcularon los costos económicos en la implementación de cada modelo, teniendo en cuenta las horas-hombre y la capacidad de etiquetado.

El desarrollo del cálculo económico en base al etiquetado manual, se obtuvo mediante información entregada por el negocio, en donde se tiene como dato, que con una cantidad

de 13 personas, se logra etiquetar solo el 10 % de las imágenes utilizando 3[h] de la jornada laboral a un costo de $1696 USD. Respecto a lo anterior, se hizo una proyección, del número de personas necesarias para etiquetar el 100 % de las fotografías. Por otro lado, para obtener el análisis de los modelos, se hizo una medición del tiempo de ejecución al procesar una cierta cantidad de fotos y, además, el costo por servicios, como las herramientas de GCP en el caso del *baseline*, y el uso de máquinas con *Graphics Processing Unit* (GPU por sus siglas en ingles) o el uso de máquinas con gran capacidad de memoria RAM, para optimizar las predicciones realizadas por el modelo *in house*.

3.7. Selección de umbrales

Para poder obtener los umbrales, se identificó el umbral que maximiza el *F1-score* para cada criterio y, además, se consideró el umbral que maximiza la diferencia entre la tasa de verdaderos positivos con respecto a la tasa de falsos positivos. En conjunto con lo anterior, se fue analizando las curvas ROC y PR en cada caso.

Capítulo 4

Resultados y Análisis

4.1. Modelo In house

4.1.1. Clasificador multi-etiquetas

A continuación se muestran los resultados obtenidos para los 4 experimentos realizados con el objetivo de determinar el mejor clasificador.

4.1.1.1. Curvas ROC

Las figuras 4.1, 4.2 y 4.3 muestran las curvas ROC de cada modelo, para los criterios de "etiqueta", "nº de domicilio" y "rostro", respectivamente. Junto a lo anterior, es posible ver el valor del área bajo la curva en cada gráfico.

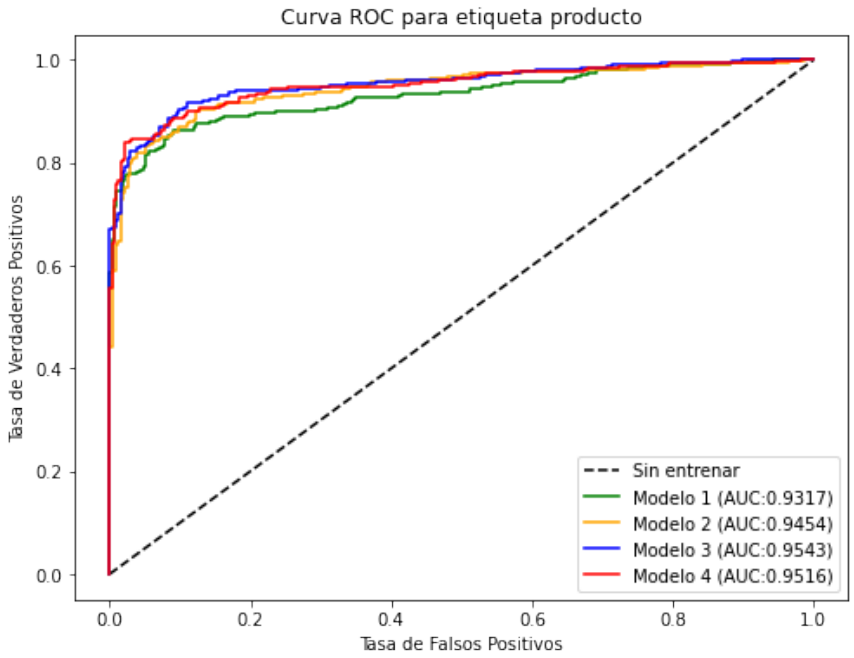

Figura 4.1: Curva ROC de la clase "etiqueta del producto" para los 4 experimentos.

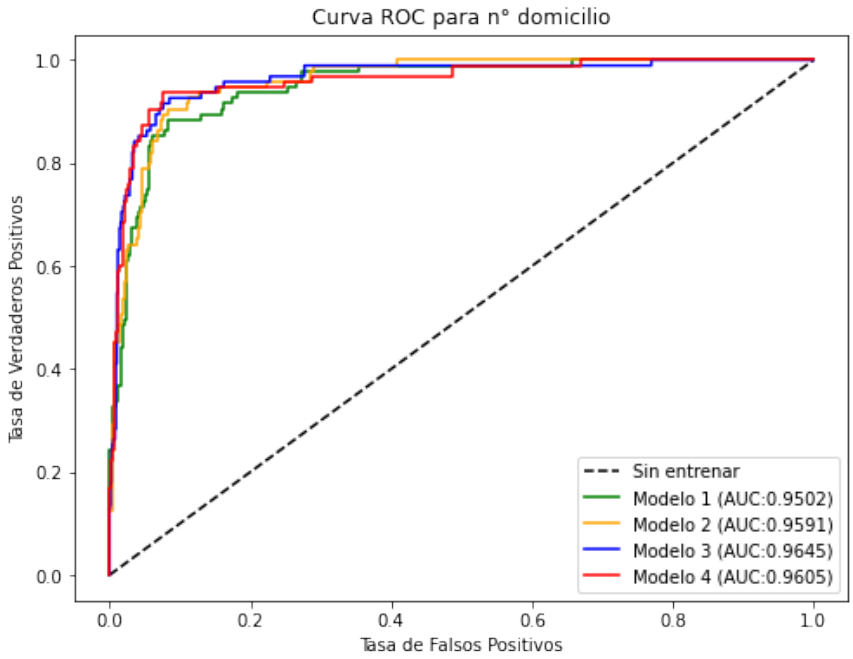

Figura 4.2: Curva ROC de la clase "nº de domicilio" para los 4 experimentos.

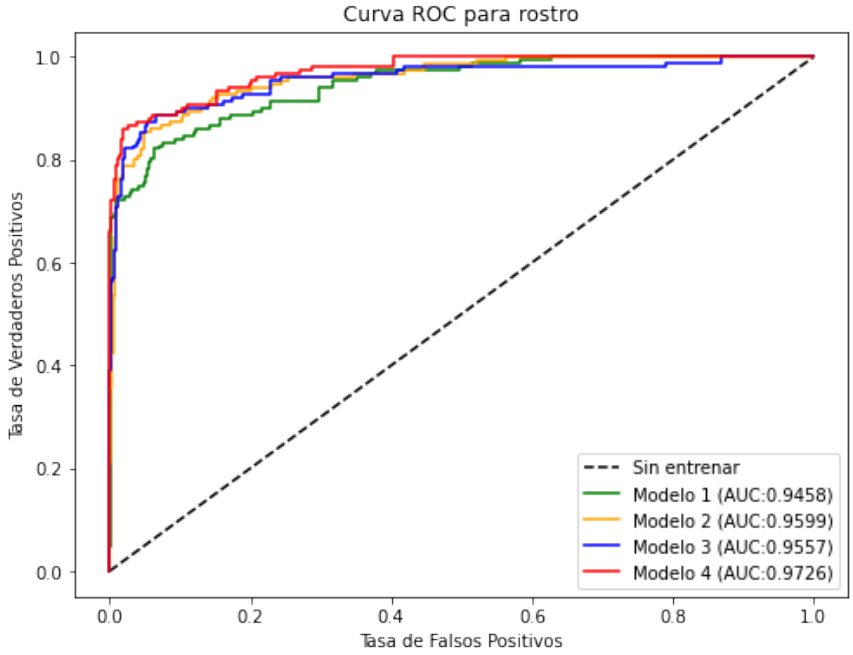

Figura 4.3: Curva ROC de la clase "rostro" para los 4 experimentos.

4.1.2. Curvas PR

Las figuras 4.4, 4.5 y 4.6 muestran las curvas PR de cada modelo, para los criterios de "etiqueta", "nº de domicilio" y "rostro", respectivamente. Junto a lo anterior, es posible ver el valor del área bajo la curva en cada gráfico.

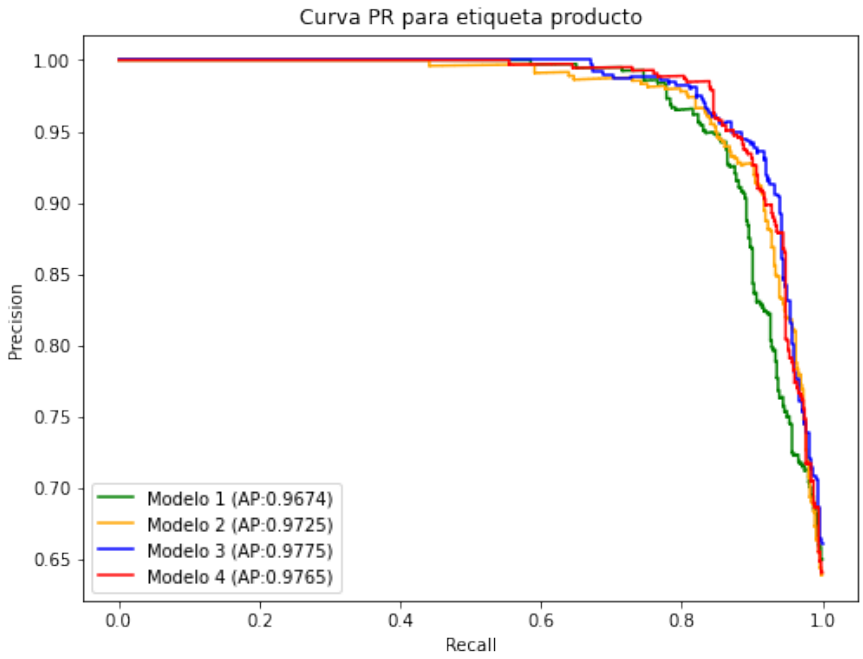

Figura 4.4: Curva PR de la clase “etiqueta del producto” para los 4 experimentos.

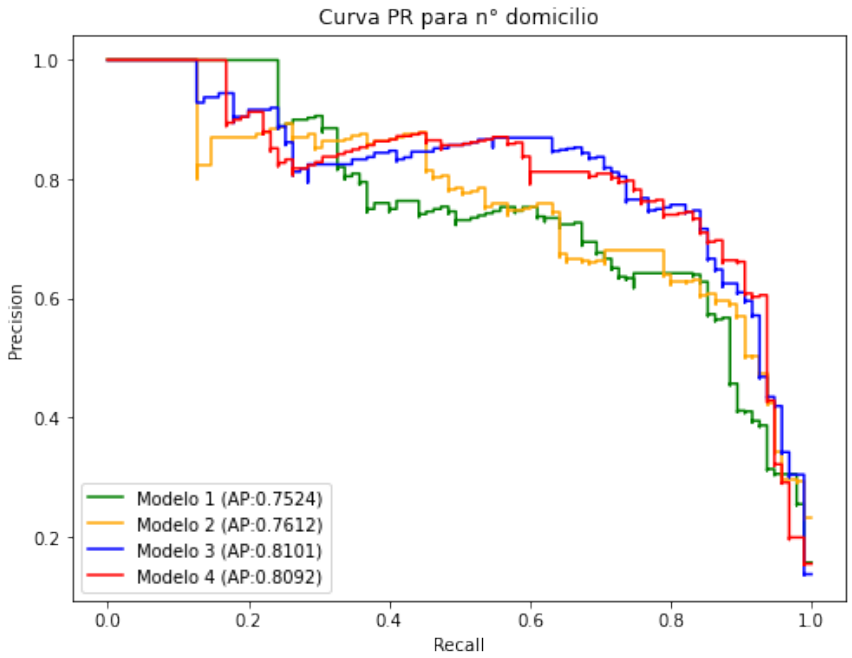

Figura 4.5: Curva PR de la clase “nº de domicilio” para los 4 experimentos.

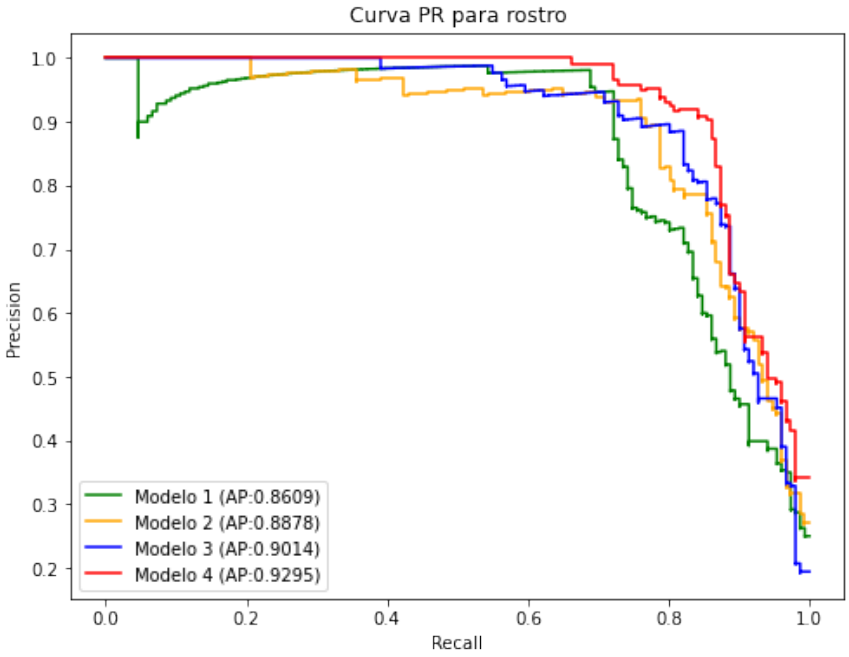

Figura 4.6: Curva PR de la clase "rostro" para los 4 experimentos.

4.1.2.1. Tabla de métricas

La tabla 4.1, muestra las métricas de *accuracy*, *precision*, *recall* y *F1-score* para cada clase y considerando los 4 modelos y un umbral de 0,5. Además, se marcaron los valores más altos con color verde.

Tabla 4.1: Tabla comparativa entre los 4 experimentos realizados, con respecto a las métricas de *accuracy*, *precision*, *recall* y *F1-score* para un umbral de 0,5.

Modelo	Clase	*Accuracy*	*Precision*	*Recall*	*F1*
1	Etiqueta	0.8463	**0.9839**	0.7711	0.8646
2	Etiqueta	0.8750	0.9607	0.8378	0.8951
3	Etiqueta	**0.8945**	0.9495	**0.8811**	**0.9140**
4	Etiqueta	0.8899	0.9791	0.8450	0.9072
1	Nº domicilio	0.8589	0.4292	0.8947	0.5802
2	Nº domicilio	0.8910	0.5000	0.9053	0.6442
3	Nº domicilio	0.8979	0.5176	**0.9263**	0.6642
4	Nº domicilio	**0.9255**	**0.6056**	0.9053	**0.7257**
1	Rostro	0.9450	**0.9478**	0.7219	0.8195
2	Rostro	0.9151	0.7104	0.8609	0.7784
3	Rostro	0.9381	0.8129	0.8344	0.8235
4	Rostro	**0.9541**	0.8725	**0.8609**	**0.8667**

4.1.2.2. Análisis

Se observa por un lado, que el modelo 3 es levemente mejor en etiqueta y el domicilio (ver AUC y AP en figuras 4.1, 4.2, 4.4 y 4.5 respectivamente). Por otro lado, el modelo 4 es mejor para identificar rostros (ver AUC y AP en figuras 4.3 y 4.6, respectivamente), lo que se justifica, ya que en ambos casos se esta implementando la red ResNeXt, que es mejor para clasificación debido a su mayor capacidad, debido a la cardinalidad y, además, se esta aplicando una transformación mas fuerte (transformación 2) lo que aumenta la diversidad de los datos de entrenamiento. Es importante notar, que el aumento de fotografías sin nº de domicilio, trajo consecuencias en los resultados, ya que por un lado, implicó mejoras en el criterio de "cara" pero, por otro lado, implicó una leve disminución de los valores de AUC y AP para las clases "etiqueta" y "nº de domicilio". No obstante, si se analizan las curvas ROC y PR, se observa que en ciertos umbrales el modelo 4 rinde mejor que el modelo 3 para esas clases. Sumado a esto, se puede ver que, considerando los resultados de la clase "nº de domicilio" para un umbral de 0,5, efectivamente se mejoraron las métricas de *accuracy*, *precision* y *F1-score* (ver tabla 4.1). De lo anterior, se desprende que al incorporar más data de entrenamiento para el modelo, es importante fijarse en la calidad, la diversidad y el desbalance que se puede generar en los datos, ya que los resultados no siempre serán mejores.

4.1.3. Detector de objetos

A continuación se muestran dos tablas referentes a las métricas de AP y AR para el detector de objetos, en conjunto con las curvas ROC y PR.

4.1.3.1. Tablas de métricas

La tabla 4.2 muestra el valor de la métrica de AP, considerando el cálculo del IoU para un umbral de 0,5 hasta 0,95 en pasos de 0,05; para un umbral de 0,5 y para un umbral de 0,75. Además, se considera el área dependiendo del tamaño del objeto y el número de detecciones igual a 100. Por otro lado la tabla 4.3, muestra el valor de la métrica de AR, considerando aspectos similares al caso anterior.

Tabla 4.2: Tabla que muestra el *average precision* obtenido en el conjunto de validación para el detector de objetos.

Métrica	IoU	Area	maxDets	Valor
AP	0.50:0.95	all	100	0.732
	0.50	all	100	0.918
	0.75	all	100	0.814
	0.50:0.95	small	100	0.125
	0.50:0.95	medium	100	0.622
	0.50:0.95	large	100	0.808

Tabla 4.3: Tabla que muestra el *average recall* obtenido en el conjunto de validación para el detector de objetos.

Métrica	IoU	Area	maxDets	Valor
AR	0.50:0.95	all	1	0.698
	0.50:0.95	all	10	0.787
	0.50:0.95	all	100	0.788
	0.50:0.95	small	100	0.310
	0.50:0.95	medium	100	0.715
	0.50:0.95	large	100	0.849

4.1.3.2. Curva ROC y PR

Las figuras 4.7 y 4.8, muestran las curvas ROC y PR, además de los valores del área bajo la curva en cada caso, para la clase "paquete".

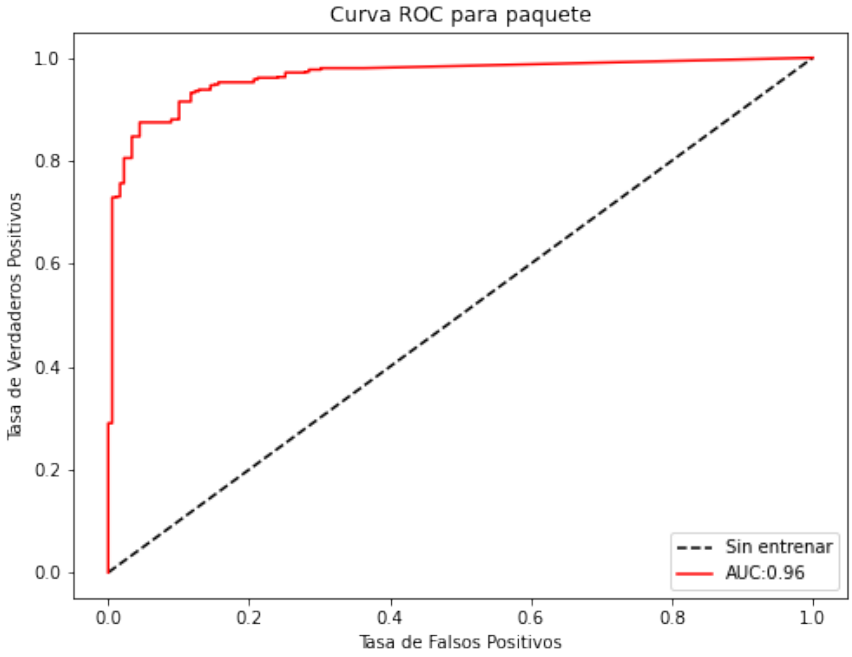

Figura 4.7: Curva ROC de la clase "paquete" obtenida con el detector de objetos.

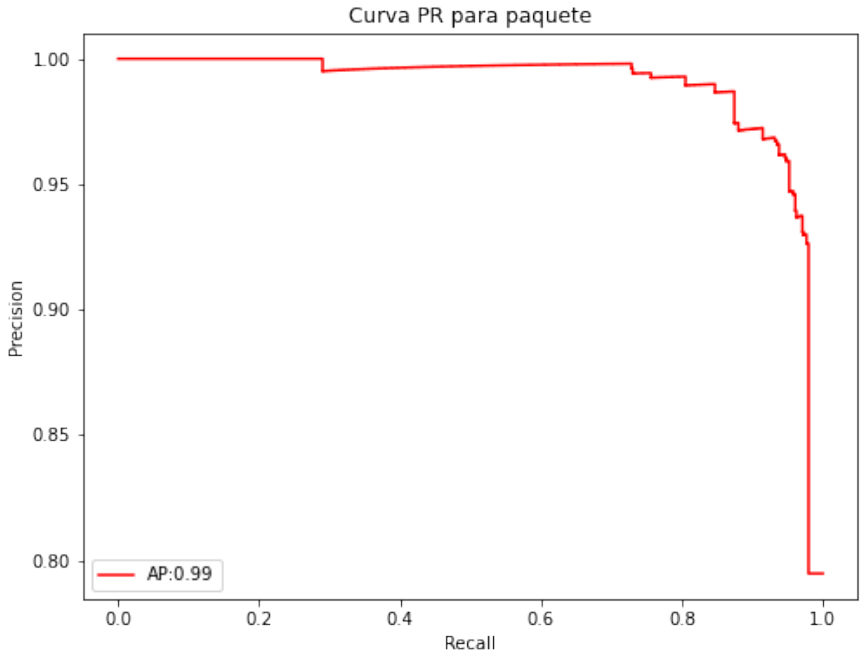

Figura 4.8: Curva PR de la clase "paquete" obtenida con el detector de objetos.

4.1.3.3. Análisis

En el caso del detector de objetos, se observa que los valores de AP y AR (ver tablas 4.2 y 4.3, respectivamente) en la mayoría de los casos, son superiores a 0,5 y cercanos a 1 en otros. Lo anterior, indica un buen rendimiento del modelo al enmarcar mediante los *bounding boxes* cada objeto. No obstante, se ve que para el caso de objetos con tamaños pequeños, el detector posee resultados inferiores a 0,5. Lo anterior, se explica ya que la distribución de fotografías de la base de datos, que presentan objetos pequeños en comparación con objetos medianos o grandes, es menor. Por otro lado, respecto a la forma de las curvas ROC y PR, además de las métricas AUC y AP, se observa que el detector de objetos, se acerca bastante a un modelo perfecto de clasificación (ver figuras 4.7 y 4.8).

4.2. Benchmark

En esta sección se muestra una comparación entre las métricas y los aspectos económicos, del *baseline* con respecto al modelo *in house*.

4.2.1. Métricas

La tabla 4.4, muestra las métricas de *accuracy*, *precision*, *recall* y *F1-score* para cada modelo y considerando los 2 modelos respectivos. Además, se marcaron los valores más altos con color verde.

Tabla 4.4: Tabla comparativa entre el *baseline* y el modelo desarrollado *in house*, con respecto a las métricas de *accuracy*, *precision*, *recall* y *F1-score* para un umbral de 0,5.

Modelo	Clase	*Accuracy*	*Precision*	*Recall*	*F1*
Baseline	Paquete	0.6778	**0.9725**	0.6118	0.7511
	Etiqueta	0.8372	0.8889	0.8505	0.8692
	N° domicilio	0.8372	0.3886	0.8632	0.5359
	Rostro	0.9209	**0.9881**	0.5497	0.7064
In house	Paquete	**0.9278**	0.9495	**0.9592**	**0.9543**
	Etiqueta	**0.8945**	**0.9495**	**0.8811**	**0.9140**
	N° domicilio	**0.8979**	**0.5176**	**0.9263**	**0.6642**
	Rostro	**0.9541**	0.8725	**0.8609**	**0.8667**

De la tabla 4.4, se observa que para un umbral de 0,5, las métricas obtenidas con el modelo *in house*, en la mayoría de los casos, son mejores que los resultados obtenidos con el *baseline*. Lo anterior, debido a que por un lado, se realizó un detector de objetos, que reconociera un tipo de paquete especifico requerido por el negocio, en vez de utilizar un detector de objetos como el de Vision API el cual estaba entrenado en un *dataset* con distintos tipos de objetos, muchos de los cuales no eran de interés. Algo parecido ocurre con la identificación de los rostros, en el cual también se consideró fotografías en que las personas aparecen con cierta parte de la cara tapadas por las mascarillas, por rejas, entre otros casos. También se desprende que al tener un mayor control sobre la estructura del modelo y, además, del entrenamiento, fue posible obtener un mejor modelo en comparación con un modelo entrenado en base a AutoML.

4.2.2. Aspectos económicos

La tabla 4.5, muestra la cantidad de personas o máquinas requeridas (en el caso de los modelos desarrollados), la jornada laboral utilizada (considerando 9 horas el 100 %), la capacidad de etiquetado y el costo manual en dolares, requerido para realizar la validación de las fotografías. En cuanto a los procesos, el "Manual 1" hace referencia al proceso previo a la implementación del modelo; el "Manual 2" corresponde a una proyección de los costos, si es que se quisiera validar el 100 % de las fotos diarias recibidas de manera manual. Finalmente, el *baseline* y *in house*, hace mención al modelo desarrollado con GCP, y el modelo desarrollado internamente, respectivamente.

Tabla 4.5: Tabla comparativa del proceso manual, el *baseline* y el modelo desarrollado *in house*, con respecto al costo de horas-hombre, capacidad de etiquetado y costo monetario.

Proceso	Cant.	Jornada [%]	Etiquetado [%]	Costo semanal [USD]
Manual 1	13	33.33	10	$ 1696
Manual 2	44	100	100	$ 17220.92
Baseline	5	100	100	$ 5112
In house	**1**	**15.68**	**100**	**$ 5.08**

En cuanto a los aspectos económicos, se observa por un lado que con el proceso manual llevado a cabo actualmente solo es posible etiquetar el 10 % del total de fotografías, lo que explica la gran limitante en el proceso. Además, se ve que el valor de etiquetar el 100 % de las fotos solo con un proceso manual, requiere de al rededor de 44 personas trabajando la jornada laboral completa lo cual tendría un costo total mensual de $ 17220,92 USD. Por otro lado, se observa que al implementar el *baseline*, esto se reduce a $ 5112 USD, lo cual sigue siendo un valor alto. No obstante, con el modelo desarrollado *in house*, los costos se reducen considerablemente a $ 5,08 USD y, además, se puede ver que para lograr aquello, solo se necesitaría de una sola maquina que trabaje el 15,68 % de la jornada laboral. Lo que con los otros métodos es imposible de lograr.

4.3. Selección de umbrales

En esta sección se muestran los mejores umbrales obtenidos para cada criterio, en cuanto a la relación entre los verdaderos positivos y los falsos positivos, y la relación de las métricas de *precision* y *recall*.

Tabla 4.6: Tabla que muestra los umbrales que maximizan, la diferencia entre la tasa de verdaderos positivos y tasa de falsos positivos, en el caso de la curva ROC, y el *F1-score* para el caso de la curva PR.

Criterios	ROC Th.	PR Th.
Paquete	0.9046	0.4376
Etiqueta producto	0.1831	0.1831
N° domicilio	0.7112	0.9672
Rostro	0.6491	0.6491

Analizando la tabla 4.6 y tomando en consideración las curvas ROC y de PR para cada criterio, se decidió utilizar los siguientes umbrales:

- Paquete: 0,5000
- Etiqueta: 0,1831
- N° Domicilio: 0,8392
- Rostro: 0,6491

Debido a los buenos resultados obtenidos con el detector de objetos, considerando un umbral de 0,5, se decidió dejar ese valor para el paquete. Por otro lado, en el caso del número de domicilio, se decidió utilizar el valor promedio.

Es importante mencionar, que esos valores, pueden ir cambiando a medida de que se vaya generando una validación periódica y dependiendo del rendimiento del modelo. Lo anterior, se propone como trabajo futuro en la siguiente sección.

Capítulo 5

Conclusiones y trabajo futuro

Se puede ver en primer lugar, que se logró implementar una solución efectiva para el negocio, reduciendo considerablemente los costos y aumentando la capacidad de etiquetado al 100 %. Esto tiene un alto impacto para el área de *Supply Chain* de Falabella, ya que con el modelo realizado, se podrá monitorear de manera óptima la validación de las entregas de los productos, lo que, eventualmente, se traducirá en una mejora de la base de datos.

En segundo lugar, se observa que el desarrollo del *baseline* mediante la plataforma de Google, permitió tener un punto de partida para la implementación de nuevos modelos, los cuales, finalmente, resultaron en un mejor rendimiento, debido al mayor control sobre la estructura de la red y el entrenamiento.

Es importante mencionar, que para lograr una implementación sostenible del modelo desarrollado a través del tiempo, se debe hacer una validación periódica de las métricas, con el propósito de verificar que el modelo esté rindiendo de la mejor manera. Considerando, si es que fuera necesario, la modificación de los umbrales y/o el re-entrenamiento del modelo.

Finalmente, dado que los modelos realizados son perfectibles, se recomienda como trabajo futuro: realizar mayor cantidad de experimentos modificando los hiperparámetros; reentrenar el modelo de manera periódica con mayor cantidad de data; definir nuevos umbrales de acuerdo a la validación periódica; y estudiar la aplicación de otro tipo de redes del estado del arte. En el caso de la clasificación de etiquetas múltiples, se podrían utilizar estructuras con *transformer-decoder* (para más detalles ver [15]), las cuales han tenido muy buenos resultados en la actualidad. Dentro de las redes que implementan esas estructuras, se encuentran la *Query2Label* y el *ML-Decoder* (ver [16] y [17] respectivamente). Por otro lado, para el detector de objetos, se podría utilizar una red como la *DINO* (ver [18]), la cual también presenta una estructura de *transformer-decoder* y, además, propone una forma de eliminar el ruido en el entrenamiento, logrando mejores resultados.

Bibliografía

[1] R. Devkar y S. Shiravale, "A Survey on Multi-label Classification for Images," *International Journal of Computer Applications*, vol. 162, no. 8, p. 39–42, Mar. 2017, doi:10.5120/ijca2017913398.

[2] X. Wu, D. Sahoo, y S. C. H. Hoi, "Recent Advances in Deep Learning for Object Detection," Aug. 2019, doi:10.48550/arXiv.1908.03673.

[3] M. Z. Alom *et al.*, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics*, vol. 8, no. 3, p. 292, Mar. 2019, doi:10.3390/electronics8030292.

[4] S. Saha, "*A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.*", https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. (accessed May 6, 2022).

[5] F. Zhuang *et al.*, "A Comprehensive Survey on Transfer Learning," June 2020, doi:10.48550/arXiv.1911.02685.

[6] S. Yang, W. Xiao, M. Zhang, S. Guo, J. Zhao, y F. Shen, "Image Data Augmentation for Deep Learning: A Survey," Apr. 2022, doi:10.48550/arXiv.2204.08610.

[7] R. Gençay y M. Qi, "Pricing and Hedging Derivative Securities with Neural Networks: Bayesian Regularization, Early Stopping, and Bagging," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 726–734, Aug. 2001, doi:10.1109/72.935086.

[8] X. He, K. Zhao, y X. Chu, "AutoML: A Survey of the State-of-the-Art," Apr. 2021, doi:10.48550/arXiv.1908.00709.

[9] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, June 2006, doi:https://doi.org/10.1016/j.patrec.2005.10.010.

[10] K. He, X. Zhang, S. Ren, y J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, doi:10.48550/arXiv.1512.03385.

[11] S. Xie, R. Girshick, P. Dollar, Z. Tu, y Kaiming He, "Aggregated Residual Transformations for Deep Neural Networks," Apr. 2017, doi:10.48550/arXiv.1611.05431.

[12] L. Jiao *et al.*, "A Survey of Deep Learning-based Object Detection," Oct. 2019, doi:10.48550/arXiv.1907.09408.

[13] S. Ren, K. He, R. Girshick, y J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jan. 2016, doi:10.48550/arXiv.1506.01497.

[14] "*Torchvision Object Detection Finetuning Tutorial.*", https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html. (accessed July 10, 2022).

[15] A. Vaswani *et al.*, "Attention Is All You Need," Dec. 2017, doi:10.48550/arXiv.1706.03762.

[16] S. Liu, L. Zhang, X. Yang, H. Su, y J. Zhu, "Query2Label: A Simple Transformer Way to Multi-Label Classification," Jul. 2021, doi:10.48550/arXiv.2107.10834.

[17] T. Ridnik, G. Sharir, A. Ben-Cohen, E. Ben-Baruch, y A. Noy, "ML-Decoder: Scalable and Versatile Classification Head," Nov. 2021, doi:10.48550/arXiv.2111.12933.

[18] H. Zhang *et al.*, "DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection," Jul. 2022, doi:10.48550/arXiv.2203.03605.