Informe Práctica 2020 Trabajo en la Empresa II

Facultad de Ingeniería y Ciencias Aplicadas,
Universidad de los Andes

Paulo Morales Aguayo
Ingeniería Civil Eléctrica

José Delpiano
Ingeniería Civil Eléctrica

Resumen ejecutivo

El presente trabajo de investigación laboró sobre el planteamiento de confeccionar una tecnología capaz de procesar y cuantificar información en tiempo real, ya sea por video o streaming, el conteo y seguimiento poblacional bajo condiciones no ideales.

El informe de investigación se basa en dos etapas: (1) análisis y evolución final del primer prototipo, siendo un dispositivo que se compone principalmente por una Raspberry Pi 3B, un acelerador ML (Google Edge TPU coprocessor), una cámara HD, un teclado inalámbrico y una batería de 5V; y (2) desarrollo de la subsiguiente etapa del proyecto "DetPax", la que corresponde al manejo de tecnologías más actualizadas y con un respetado desempeño en el análisis de datos, como un procesador AMD Ryzen 7 3700U, al igual que la implementación de algoritmos computacionales pertinentes al manejo y desempeño de estas nuevas unidades tecnológicas, como el algoritmo de Kalman. Dicho mejoramiento se propone como un segundo y consecuente prototipo, con un desarrollo de alta calidad con el fin de desempeñar tareas de elevada precisión.

Los alcances del proyecto fueron probados sobre bases de datos de distintos medios de transporte públicos, como Metro de Santiago y/o TrenCentral Grupo EFE; o, también, en videos capturados en experimentos del Laboratorio de Dinámica Humana de la Universidad de los Andes, donde se simula la subida y bajada de pasajeros en un vagón de metro y su respectivo andén.

1. Introducción

1.1 Descripción del Área principal de Investigación del Profesor

Los experimentos conocidos más antiguos en visión computacional ocurrieron en los años 1950's, usando algunas de las primeras redes neuronales para detectar bordes de un objeto, y para ordenar figuras sencillas en categorías como círculos y cuadrados. En la década de los años 1970's, el primer uso comercial de visión computacional interpretó texto escrito a mano, o mecanografiado, utilizando el reconocimiento visual de caracteres. También, a medida que internet crecía en los años 1990's, se generaron grandes cantidades de imágenes completamente disponibles para que el análisis en reconocimiento facial floreciera.

Hoy en día, los sistemas de inteligencia artificial pueden ir un paso más allá y tomar acciones basadas en un entendimiento de la imagen. Existen varios tipos de visión por computadora que son usados en diferentes formas:

- Segmentación de imagen: particiones de una imagen en múltiples regiones o piezas para ser examinadas separadamente.
- Detección de objetos: identifica un objeto específico en una imagen. Detecciones de objetos avanzados reconocen varios objetos en una simple imagen.
- Reconocimiento facial: es un tipo avanzado de detección de objetos que no solo reconoce una cara humana en una imagen, sino que identifica un individuo en específico.
- Detección de bordes: es una técnica usada para identificar el borde exterior de un objeto o paisaje para identificar de mejor manera lo que hay en la imagen.
- Detección de patrones: es un proceso de reconocer figuras repetidas, colores, y otros indicadores visuales en imágenes.
- Clasificación de imágenes: agrupa imágenes en diferentes categorías.
- Emparejamiento de características: es un tipo de detección de patrones que coinciden en similitudes en imágenes para ayudar a clasificarlas.

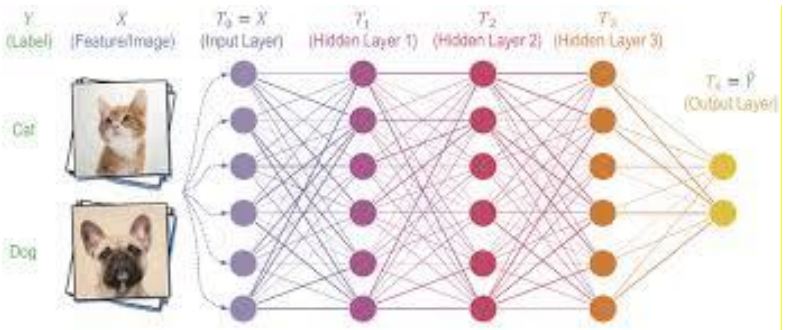

Fig. 1: Ejemplificación de detección y/o clasificación.

1.2 Descripción del Trabajo previo del Profesor

Uno de los trabajos más recientes del profesor José Delpiano es "Comparison of convolutional neural networks in fruit detection and counting: A comprehensive evaluation[1] ". En dicho trabajo se realizan comparaciones entre las redes "Faster R-CNN" con la arquitectura "Inception V2" y "Single Shot-Multibox Detector (SSD)" con "MobileNet", en conteo de frutas, como paltas Hass, limones y manzanas, bajo diferentes condiciones ambientales. Este trabajo se realizó junto a los profesores Juan Pablo Vasconez y Fernando Auat Cheein de la Universidad Técnica Federico Santa maría, y Stavros Vougioukas de University of California.

Fig. 2: Detección de limones con las diferentes arquitecturas.

Otro de los trabajos publicados es "Deep learning for image-based classification of OAM modes in laser beams propagating through convective turbulence[2] ". Esta entrega trata acerca de cómo el espacio libre de comunicaciones óptico es altamente sensible a distorsiones inducidas por turbulencias atmosféricas. Como las actuales técnicas de machine learning para visión computacional permiten clasificación precisa de imágenes en general, se estudió el uso de redes neuronales convolucionales para reconocimiento de intensidad de patrones de estados de OAM (orbital angular momentum) luego de experimentos de propagación en el laboratorio. El estudio fue realizado en conjunto a Gustavo Funes, Jaime Cisternas, Sebastián Galaz y Jaime Anguita, provenientes de la Universidad de Los Andes.

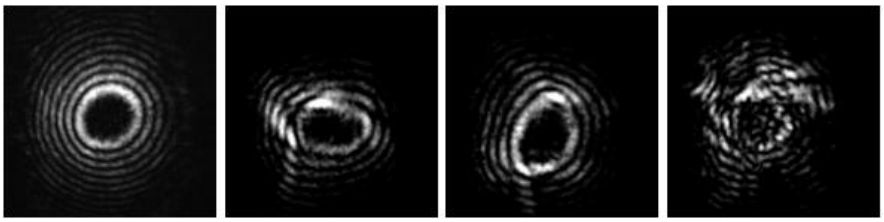

Fig. 3: Imágenes tomadas en el laboratorio.

[1] https://www.researchgate.net/publication/340953847_Comparison_of_convolutional_neural_networks_in_fruit_detection_and_counting_A_comprehensive_evaluation

[2] https://www.researchgate.net/publication/335668623_Deep_learning_for_image-based_classification_of_OAM_modes_in_laser_beams_propagating_through_convective_turbulence

2. Trabajo Realizado

2.1 Actividades Generales

La primera etapa del proyecto, correspondiente al manejo de Raspberry Pi 3B en conjunto al acelerador Coral Edge TPU, se caracterizó por implementar mejoras en códigos y estructura de funcionamiento, escritos en Python. El código se basó en el script "Opencv People Counter[3] ".

Fig. 4: Raspberry Pi, Coral Edge TPU y componentes.

Las modificaciones al script de base terminaron con la estructura resultante de la figura 5.

```
.
|--- pyimagesearch
|      |--- __init__.py
|      |--- centroidtracker.py
|      |--- trackeableobject.py
|--- Coral Edge TPU
|      |--- mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite
|      |--- coco_labels.txt
|--- output
|      |--- output_01.avi
|--- detect_video.py
```

Fig. 5: Estructura resultante de 1ra. Instancia del proyecto.

El proceso detrás de la estructura es el siguiente: (1) realizar inferencia sobre el frame entrante, con el fin de obtener detecciones; (2) calcular el centroide de la detección y asignarle un ID; (3) realizar seguimiento del respectivo ID, sin perder la información de identificación; (4) contabilizar detecciones a través de una línea amarilla en la mitad de la pantalla.

Los códigos "centroidtracker.py" y "trackeableobject.py" son, en su mayoría, los originales de pyimagesearch. Sus funciones son calcular el centroide del objeto detectado a través de la

[3] https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/

inferencia, y realizar un seguimiento del objeto detectado manteniendo su ID, respectivamente. En cuanto a la sección "Coral Edge TPU", "mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite" es el archivo con los pesos y estructura necesaria para ejecutar la red neuronal implementada mediante el modelo SSD (Single Shot-Multibox Detector), mientras que "coco_labels.txt" representa las clases de "objetos" que es capaz de detectar el modelo, teniendo entre ellas la clase "personas". Las inferencias son realizadas mediante el dispositivo Coral, en conjunto a los archivos de su respectiva sección en la estructura. La contabilización de personas y manejo de filtros/modificaciones por imágenes entrantes/salientes quedan en manos del archivo principal "detect_video.py", que convoca también a los anteriores mencionados. La sección "output" es donde se encuentra la grabación del desempeño del detector a través del archivo resultante "output_01.avi".

En la segunda etapa, se utilizó un procesador AMD Ryzen 7 3700U junto al acelerador de Google. En esta fase se apuntó a implementar algoritmos más adecuados a las nuevas características de hardware. Se utilizó como base el proyecto "experimenting-with-sort[4]", el que pone en funcionamiento el algoritmo de Kalman como tracking de objetos. Este algoritmo se caracteriza por ser un modelo de predicción en base a movimiento.

```
.
|--- experimenting-with-sort
|       |--- sort.py
|       |--- data_association.py
|       |--- kalman_tracker.py
|--- Coral Edge TPU
|       |--- mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite
|       |--- coco_labels.txt
|       |--- detect_tf.py
|       |--- detector.py
|--- output
|       |--- output_02.avi
|--- main.py
```

Fig. 6: Estructura modificada.

La idea del nuevo proyecto es la misma que en la primera fase.

El código "sort.py[5]", significando "simple online and realtime tracking", permite seguir múltiples objetos en secuencias de video en 2D. "data_associaton.py" se encarga de ordenar la información e identificarla como detecciones nuevas o antiguas. El script "Kalman_tracker.py" recibe la información del código anterior para calcular las variables con sus estimaciones de tracking.

En cuanto a la sección "Coral Edge TPU", se caracteriza por ser similar a la fase anterior, pero con filtros y ordenamientos de las detecciones a través de "detect_tf.py"; mientras que en el código "detector.py" se realizan las inferencias, bajo un margen de probabilidad de ocurrencia, a través de la sub librería tflite de Tensorflow.

Se planificaron reuniones cada 2 semanas con los profesores José Delpiano y Sebastián Seriani, con el fin de analizar los avances y comentarlos bajo una perspectiva científica y sustancial.

[4] https://github.com/ZidanMusk/experimenting-with-sort

[5] https://github.com/abewley/sort

2.2 Actividad Desafiante

Uno de los desafíos encontrados en la primera parte del proyecto fue la ejecución del algoritmo completo, del primer dispositivo, sin que el hardware colapsara. Esto requirió el ordenamiento y filtración de la información entrante en cada frame. La filtración correspondía a aplicar redimensionamientos a las imágenes entrantes, para luego entregar a Coral el trabajo de inferencias sobre dichas imágenes. También, se debió implementar opciones de reproducción, como una interfaz gráfica, para bajar la carga de la CPU de Raspberry Pi. Este modelo del dispositivo, Raspberry Pi 3B, posee sólo 1 GB de memoria RAM, por lo que la tarea requerida podía llegar a ser sofocante tanto en temas de procesamiento y aumento de temperatura, arriesgando así el rendimiento y el artefacto mismo. La solución planteada frente al problema de rendimiento fue la instalación del TPU de Google, mientras que en el aumento de temperatura fue de instalar un ventilador adecuado al dispositivo. El TPU también contribuyó a la disminución de calor en el dispositivo.

Fig. 7: Medición del primer prototipo en experimentos de Laboratorio de Dinámica Humana.

Si bien, la primera parte del proyecto significó un desafío importante en su implementación general, la segunda parte significó un desafío superior a medida que las metas en la investigación iban siendo alcanzadas.

El primer desafío importante encontrado en la segunda fase del proyecto fue la implementación de un algoritmo de seguimiento que fuera preciso. La idea de un mejoramiento en esta área se debe a que en la primera etapa se realizaban seguimientos a las personas detectadas, pero con un bajo nivel de re-identificación, es decir, el algoritmo incurría en el error de detectar una misma persona como otra distinta entre frames consecutivos. Esto se debe a que en el primer código, el tracking de la persona se realizaba mediante una librería llamada "dlib", la cual busca coincidencias mediante componentes estadísticas para verificar "visualmente" que se refiriera al mismo personaje anteriormente identificado. En sí, "dlib" es un algoritmo de seguimiento en base a apariencia. Esto puede incurrir en un problema importante al momento de realizar contabilizaciones en ambientes no ideales, ni tampoco en la deformación de los objetos a identificar. Por ejemplo, si una persona es identificada, sea cual sea su posición, en un andén de Metro de Santiago, y luego de un momento decide girar 60°, esto necesariamente cambia la perspectiva visual de la detección, por lo que el dispositivo identificará, posiblemente, que en ese lugar se encuentra una persona, pero no la misma que en el frame anterior.

Debido a situaciones como la recién explicada, se identificó la necesidad de escribir códigos que se adecuaran al dinamismo de situaciones reales. Aquí es donde se investiga la posibilidad de implementar el algoritmo de Kalman.

El algoritmo de Kalman se caracteriza por ser un algoritmo de predicción en base a movimiento, es decir, simula dentro de parámetros de incertidumbre y de velocidades, que para el caso de esta investigación se instituyeron como constantes, las posiciones futuras de las detecciones actuales, llegando a considerarse como un algoritmo de tracking.

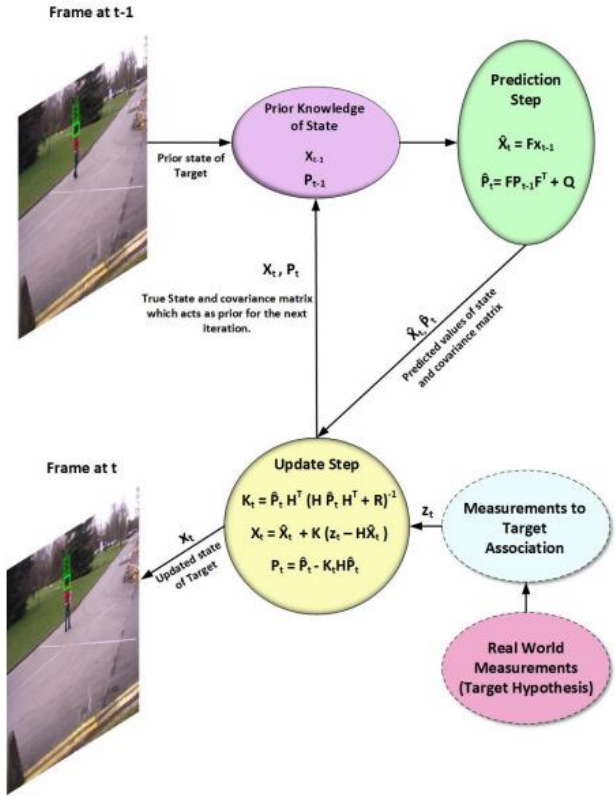

Fig. 8: Esquema de algoritmo de Kalman.

La dificultad de implementar este algoritmo son las variables para identificar si el objeto detectado es nuevo o antiguo. Para esto se calculó la intersección sobre la unión (IOU – intersection over union) sobre cada objeto seguido y detectado. Con esto, se genera la matriz IOU, en la que las posiciones de cada elemento de la matriz corresponden a una detección versus un tracking. Se realiza el cálculo de IOU de todas las detecciones versus todas las posibilidades de tracking.

$$\text{Detecciones} \begin{bmatrix} iou_{1,1} & \dots & iou_{1,n} \\ \dots & \dots & \dots \\ iou_{n,1} & \dots & iou_{n,n} \end{bmatrix} \quad \text{Tracking}$$

Fig. 9: Matriz IOU.

El cálculo de IOU se realiza tomando las áreas de los elementos predichos y las detecciones actuales. Estas áreas son de carácter cuadrado, rectangular, etc. Por lo tanto, se toma la intersección geométrica de ambos cuadros, sobre la unión de los mismos, obteniendo así una métrica puesta en evaluación para identificarla como nueva detección, o como seguimiento.

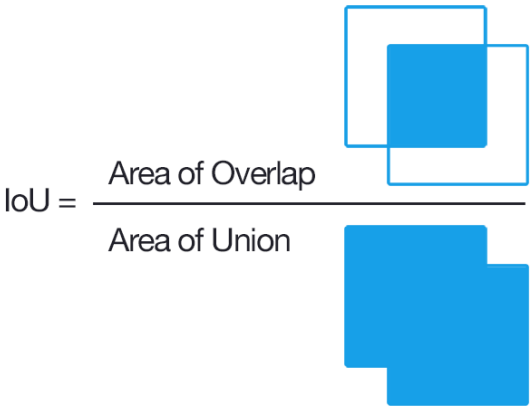

Fig. 10: Demostración del cálculo de IOU.

Luego de obtener la matriz, se aplica el algoritmo húngaro para asignar cada detección al mejor posible objeto seguido, es decir, al que posee el máximo IOU.

El algoritmo húngaro que se utilizó fue con respecto al problema de asignación (assignment problem), el cual es un problema de optimización combinatorial. Para ejemplificar y explicar este algoritmo se presenta el siguiente caso:

"*En el siguiente problema se tiene a varios agentes y varias tareas. Se puede asignar a cualquier agente para realizar cualquier tarea, incurriendo en un costo que puede variar dependiendo de la asignación de la tarea del agente. Se requiere realizar tantas tareas como sea posible asignando como máximo un agente a cada tarea y como máximo una tarea a cada agente, de tal manera que el costo total de la asignación se minimice*[6]*."*

Pero como el problema de asignación se enfoca en minimizar, y el cálculo de IOU de detecciones-seguimiento se enfoca en maximizar, entonces el algoritmo húngaro se aplicó sobre una matriz IOU negativa, es decir, se antepuso un signo menos a la matriz.

Con las herramientas explicadas, al algoritmo de Kalman solo le queda ser implementado como en la figura 8.

El segundo desafío importante respecto a la segunda fase del proyecto tiene que ver con implementar una red neuronal desde cero que permitiera identificar cabezas. Este desafío presenta varias aristas en dificultad. Una de ellas es obtener la base de datos, es decir, obtener un conjunto de al menos 2000 imágenes de cabezas de personas, en condiciones de transporte público o de flujo de personas, para luego etiquetar a mano dichas imágenes. Este problema genera varios contratiempos, entre ellos específicamente el mismo tiempo, y la búsqueda de estas imágenes. Si bien, en páginas de internet como Google o Youtube se puede dar una búsqueda eficiente, el

[6] https://en.wikipedia.org/wiki/Assignment_problem

problema sigue recayendo en la cantidad de imágenes, el tiempo y comodidad necesaria. Con este problema presente, se generaron 2 soluciones posibles. La primera de ellas fue generar un código en base a "Tracking Multiple Objects with OpenCV[7] ", el cual permite seleccionar con el puntero los objetos que se desea rastrear en el frame, hasta el final del video. Esto se logra a través del algoritmo CSRT de la librería OpenCV.

CSRT tracker es un algoritmo que usa un mapa de confiabilidad espacial que se ajusta a la parte de la región seleccionada del frame para rastrear. Esto asegura dimensionar y localizar la región seleccionada y mejorar el seguimiento de las regiones no rectangulares u objetos.

Con esto, finalmente, el código se encarga de emitir los archivos e imágenes correspondientes a ese frame para ser cargados en algún entrenamiento de redes. El código descrito se llama "CSRT-manual.py".

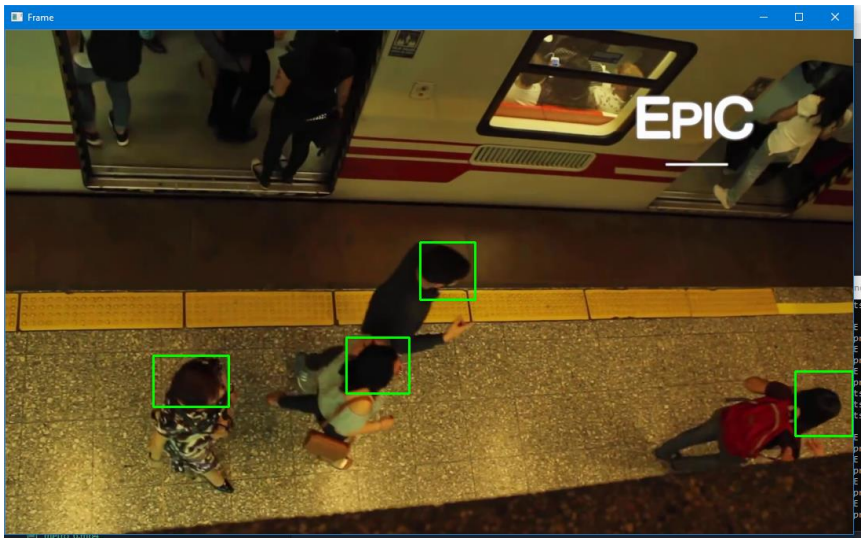

Fig. 11: Código "CSRT-manual.py" en funcionamiento en video de metro encontrado en youtube. Las cajas verdes fueron seleccionadas por el puntero.

La segunda solución posible se encuentra en la investigación "Fast Human Head and Shoulder Detection Using Convolutional Networks and RGBD Data[8] ". Este escrito presenta una solución parcial frente al problema, ya que entrega de forma gratuita la base de datos con las que trabajaron, facilitando el manejo de la información. A pesar de que esta base de datos viene con sus etiquetas en formato JSON, no es de utilidad/comodidad para este trabajo; además, las imágenes recuperadas no son similares a las mostradas en la investigación, ya que les adicionan ruido a las caras de las personas, por lo que el trabajo manual se reduce como máximo en un 50%.

[7] https://www.pyimagesearch.com/2018/08/06/tracking-multiple-objects-with-opencv/

[8] https://openaccess.thecvf.com/content_CVPRW_2020/papers/w6/Ahmar_Fast_Human_Head_and_Shoulder_Detection_Using_Convolutional_Networks_and_CVPRW_2020_paper.pdf

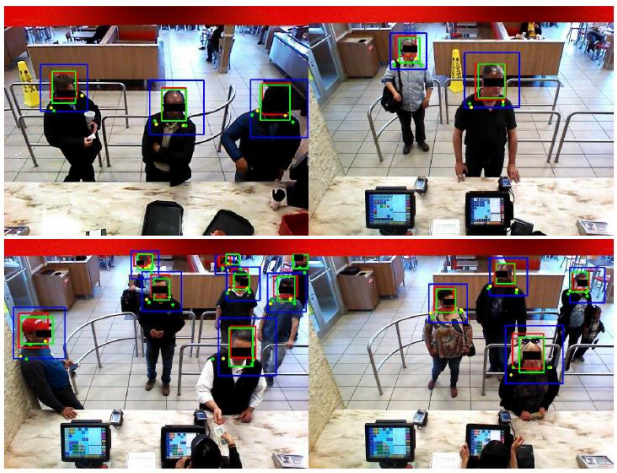

Fig. 12: Imagen facilitada por "Fast Human Head and Shoulder Detection Using Convolutional Networks and RGBD Data".

Para seguir, las imágenes de esta base de datos debieron ser etiquetadas manualmente a través de la aplicación "LabelImg[9] " en formato "Pascal VOC". La aplicación permitía seleccionar con el puntero algun área cuadrada de interés, que en este caso corresponde a cabezas.

Otro de los desafíos presentes con la implementación de la red neuronal es entrenar la misma red. Por lo general, se necesita una tarjeta GPU para acelerar el proceso de entrenamiento, de lo contrario, un computador se demorará días, y hasta semanas posiblemente, en cambio una GPU se demorará horas. Por eso, para este tipo de labores las tarjetas GPU son indispensables. Sin embargo, pueden llegar a ser muy difíciles de conseguir y/o muy costosas. Por esta razón, se utilizó Google Cloud Platform, el cual provee de todas las herramientas tecnológicas digitales necesarias para utilizar una GPU de manera remota.

La red designada para el entrenamiento fue "Faster RCNN", junto a la arquitectura "inception V2". Se utilizó la librería "TensorFlow", pudiendo descender la cantidad de imágenes etiquetadas de 2000 a 1470, gracias a "data augmentation".

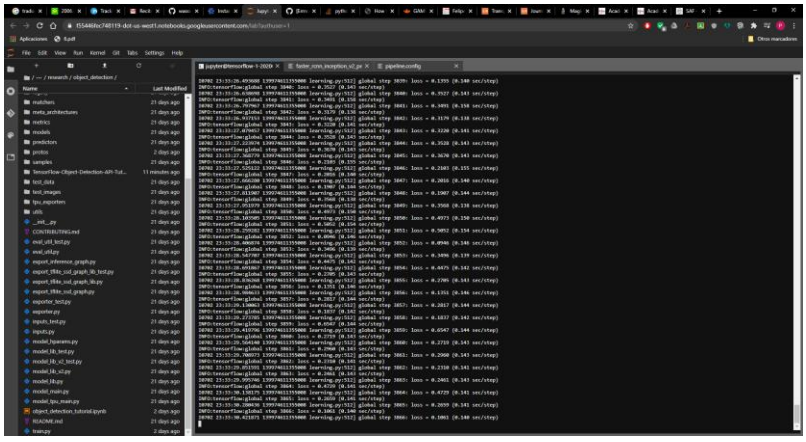

Fig. 13: Entrenamiento de red en Google Cloud Platform.

[9] https://github.com/tzutalin/labelImg

3. Análisis

3.1 Análisis del Área de Investigación

Si bien el área de investigación asignada no se presenta como una completa novedad, sí lo fue la petición de contabilización por densidades y flujos poblacionales.

Como esta área es un tema actual y recurrente, la obtención de códigos y técnicas para sus diversas implementaciones son abundantes en internet y libros. Sin embargo, lo que sí puede presentarse como escaso y distante son las bases de datos, sobre todo si estas requieren características específicas. Todo trabajo con redes neuronales desde cero se presenta como un desafío producto de la obtención de la información de entrenamiento, como de la red a entrenar y su proceso.

El papel de las bases de datos es significativo a la hora de entrenar una red. Estos datos deben poseer características de acuerdo a sus requisitos, pero lo que es más preponderante es la cantidad de elementos a procesar. Se designa como una métrica estándar el que cada clase de objeto a detectar deba incluir al menos 2000 imágenes en el entrenamiento de la red. Además, se debe agregar un porcentaje inferior de imágenes para realizar pruebas, es decir, este porcentaje por lo general se toma en un 20% del conjunto de entrenamiento. También, se debe considerar agregar imágenes que no representan nada, es decir, imágenes "negativas". Esto se realiza para no producir overfitting en la red. El tamaño del conjunto negativo debe ser como máximo igual al total de los objetos a detectar.

Afortunadamente, existen variadas técnicas para aumentar la cantidad de imágenes a detectar en el caso que estas sean escasas. Estas técnicas se incluyen en lo que se denomina como "data augmentation", y pueden realizar acciones sobre las imágenes como cambiar las intensidades de colores, rotar, trasladar, recortar, sobreponer, entre otras más. En la actualidad existen diversas librerías o API's que permiten manejar de manera cómoda el data augmentation, siendo unas más eficientes que otras.

Fig. 14: Algunas librerías destinadas a la inteligencia artificial.

Actualmente, la cantidad de bases de datos para entrenar redes ha ido en crecimiento gracias a la comunidad científica. Es posible encontrar bases de datos etiquetadas, ya sea por segmentación, detección, clasificación, etc., en "ImageNet[10] ", "Kaggle[11] ", entre otros más.

[10] http://www.image-net.org/

[11] https://www.kaggle.com/tags/video-data

Cabe mencionar que es posible encontrar bases de datos que no se vinculan únicamente a imágenes. Existe una amplia gama de variedades. Esto se debe a que en la comunidad científica se realizan competencias con relación al área de inteligencia artificial en general. Como ejemplo, es posible ver que en "Kaggle" existen competencias con distintos enfoques y requerimientos[12].

Adicionalmente, es necesario mencionar los distintos congresos que se llevan a cabo en diversas áreas de tecnología. Es el caso de CVPR (Computer Vision Pattern Recognition[13]), el que se presenta como un congreso anual, y siendo el principal evento de visión computacional, en el que es posible asistir a workshops y talleres cortos, con el fin de aportar conocimientos al área mencionada.

Finalmente, es importante mencionar el interés que las empresas y centros de estudio han demostrado respecto al tema, ya que en reiteradas ocasiones se han liberado de forma temporal o permanente cursos abiertos al público para aprender conceptos y materias relacionadas a la inteligencia artificial. Algunos ejemplos son "Tensorflow", el cual entrega capacitaciones para distintos niveles de usuarios[14]; y "Coursera[15] ", la cual es una página web que ofrece cursos de distintas materias, en una gran mayoría gratis, pero que tienen la opción de ser acreditados.

En vista de la gran variedad de opciones y preparaciones frente a la visión computacional, es remarcable la efervescencia que está teniendo el rubro.

Fig. 15: Capacitaciones de Tensorflow.

[12] https://www.kaggle.com/competitions

[13] http://cvpr2020.thecvf.com/

[14] https://www.tensorflow.org/resources/learn-ml

[15] https://www.coursera.org/

3.2 Análisis del Trabajo Realizado

Uno de los primeros problemas que se presentaron fue el conocimiento de arquitecturas de hardware sencillas como lo es una Raspberry Pi. Este problema produjo severas tardanzas en los primeros desarrollos, debido a que este dispositivo no se emplea igual que un computador tradicional, a pesar de que su funcionalidad es la misma, pero a menor escala. La solución a esta problemática fue la obtención de conocimientos de este aparato, en cuanto a hardware, arquitectura, funcionalidad, etc. Este problema se pudo evitar pensando en la formación académica previa, es decir, impartir cursos en la carrera de Ingeniería Civil Eléctrica donde se requiera la utilización y experimentación con estos dispositivos. Es necesario tener conocimientos previos de estos gadgets, puesto que no poseen protecciones eléctricas frente a malos manejos o instalaciones de elementos externos, pudiendo provocar desperfectos.

Fig. 16: Raspberry Pi.

Otro problema importante a mencionar son las redes neuronales.

La implementación de una red requiere de conocer dicha red y su arquitectura, pero también el procedimiento que existe por detrás, es decir, saber a ciencia cierta qué es lo que se necesita realizar, y cómo se implementará, requiriendo de conocimientos previos. Dicho esto, uno de los mayores problemas que surgieron fue la decisión de qué estructura usar al momento de entrenar la red. Esta estructura se relaciona directamente con las capacidades de hardware poseídas. Si bien, a lo largo del informe se han mencionado las redes "SSD" y "Faster RCNN", estas tienen distintos niveles de desempeño, siendo "Faster RCNN" más eficiente que su competidora. Sin embargo, esto va de la mano con un costo computacional, es decir, el hardware tiene una responsabilidad importante en dicho procesamiento. Por esto, la implementación de la primera etapa del proyecto, en Raspberry Pi, no se efectúa con la última red, sino que con "SSD". Este tipo de decisión debería ser clara e inmediata al momento de presentarse el caso. Por suerte, la implementación de la primera etapa ya venía ligada a la red "SSD". Aunque para el caso de la segunda parte del proyecto, hubo que decidir qué tipo de red se implementaría, teniendo que realizar búsquedas a cabalidad para obtener y validar esta información. Obviamente, se incurre en gastos de tiempo en dicha instrucción. Para solucionar este problema, se propone incluir en los programas de estudio, referente a las áreas de computación y electricidad, la enseñanza de redes actuales y cómo funcionan sus estructuras.

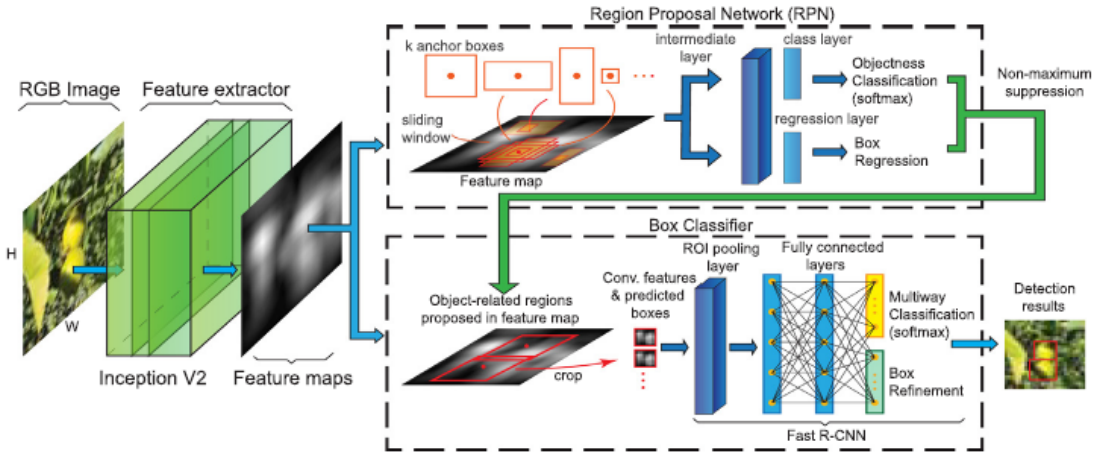

Fig. 17: Esquematización de Faster RCNN.

Otra problemática relacionada a las redes neuronales es el procedimiento que se debe llevar a cabo para entrenar una red. Si bien, existen varios tutoriales en internet, para distintas librerías y lenguajes de programación, el manejo computacional no deja de ser importante. En reiteradas ocasiones, dichos tutoriales no se encuentran actualizados, ni mencionan los pormenores que se pudieran encontrar en el proceso, produciendo una gran pérdida de tiempo y frustración. En otros momentos, hubo conceptos computacionales que escapaban del manejo eléctrico usual, generando bastante confusión e incertidumbre. La solución a este problema es, nuevamente, enseñar a través de algún curso relacionado a la temática cuales son las instrucciones de estas implementaciones y los porqués de estas, para así poder llegar a tener una comprensión y manejo de datos absoluto. Incluso, se ha llegado a pensar que la incorporación de materias computacionales deberían ser "electivos-obligatorios" en la carrera de Ingeniería Civil Eléctrica.

4. Reflexión Personal

El trabajo de investigación entregó al alumno conocimientos de lo que es realizar un trabajo de esta especie. Esto requirió de metodología, paciencia y solidaridad. En cuanto a la solidaridad, se refiere a que es productivo hablar con otras personas relacionadas al mismo tema, y compartir con ellas distintos puntos de vista o conceptos. Esto ayuda de sobremanera a plantear ideas y calmar las frustraciones que se podían encontrar en el camino. Gracias a esto, la metodología se plantó como un pilar fundamental al momento de averiguar nuevas cosas, ya fuera por alguna lectura, o por un intercambio de palabras con otras personas.

Los aportes mencionados permiten no malgastar recursos valiosos como la energía y el optimismo, los que se pueden describir como impulsores en los momentos de estancamiento.

Con estas ideas fue posible realizar un trabajo de investigación extenso y fructífero en cuanto a resultados.

Además, la investigación permitió reforzar entendimientos que pueden haber sido mal aprendidos o haberse erosionado con el tiempo. Pero, adicionalmente, proporcionó intelecto y herramientas con utilidades que servirán en el día a día, e incluso en proyectos más ambiciosos. Se agradece pensar que esto alimentó no solo el espacio intelectual, sino que también lo hizo en el espacio de la confianza personal.

La presente investigación ayudó a crecer no solo profesionalmente, sino que también personalmente. Si se visualiza desde el comienzo, el proyecto pasó por etapas de absoluto desconocimiento, es decir, hubo que explorar terrenos desconocidos y ser capaz de afrontar los desafíos encontrados. Esto presupone un desarrollo superior sobre los conocimientos y templanza frente a situaciones exigentes.

Finalmente, se agradece la oportunidad otorgada, ya que esto permitió reforzar el interés y curiosidad frente a temas de visión computacional, admitiendo así el ingreso de materias y conceptos nuevos; y, también, la afiliación con otras áreas en las cuales se pueden aplicar los conocimientos aprendidos y poder seguir aportando a la ciencia y a la calidad de vida de las personas.