UNIVERSIDAD DE ANTIOQUIA

REDO: SISTEMA DE RECONOCIMIENTO DE RESIDUOS RECICLABLES

Autor:
Manuela Castrillón Medina

Universidad de Antioquia
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas
Medellín, Colombia

2019.

REDO: Sistema de reconocimiento de desechos reciclables

Manuela Castrillón Medina

Informe de Práctica presentado como requisito parcial para optar al título de:
Ingeniera de Sistemas

Asesor:
Julian David Arias - Associate Professor

Universidad de Antioquia

Facultad de Ingeniería
Medellín, Colombia
2019.

Resumen

En este trabajo se aborda el proceso de clasificación de residuos desde dos enfoques, el primero haciendo uso de herramientas de Auto machine learning, las cuales a través de servicios cloud, permiten diseñar de forma automática un modelo de machine learning para un dataset dado, el segundo enfoque se centra en el diseño manual de un modelo para visión por computadora, haciendo uso de redes neuronales convolucionales y transfer learning, el cual permite tomar un modelo previamente entrenado, para el caso de nosotros con el dataset ImageNet, y ser adaptado para el problema en particular de reconocimiento que se tenga, para el diseño de este modelo se realizan un promedio de 72 experimentos, entre los cuales se variaron aspectos de la arquitectura tales como la arquitectura base, clasificador e hiper parámetros del modelo. Los resultados obtenidos con el modelo resultante de las herramientas de Auto machine learning fueron muy superiores a los resultados obtenido con el diseño manual, teniendo estos primeros una precisión entre 93% y 97% mientras que los segundos alcanzaron una precisión máxima de 72%.

Introducción

Según el ministerio de medio ambiente, en Colombia se producen anualmente doce millones de toneladas de basura, de las cuales, solo un 17% es aprovechada a través de procesos de reciclaje. Según un informe de 2015 del Banco Mundial y el Departamento de planeación nacional “si se continúa con la misma dinámica de generación de residuos, sin adecuadas medidas para mejorar su aprovechamiento o tratamiento, y con patrones de producción y consumo insostenibles, en el año 2030 tendremos emergencias sanitarias en la mayoría de ciudades del país y una alta generación de emisiones de gases de efecto invernadero”.

Una de las mayores limitaciones para mejorar estos indicadores es la falta de educación para la correcta clasificación de las basuras, motivo por el cual, surge la necesidad de crear herramientas con la capacidad de apoyar este proceso de aprendizaje.

El principal objetivo es construir un sistema cognitivo de visión por computadora que apoye el proceso de educación en temas de separación de desechos en ambientes laborales. Para este fin se construye *Redo*, un asistente creado con técnicas de aprendizaje profundo capacitado en el reconocimiento de desechos. Redo servirá como un puente amigable para los usuarios y centra sus esfuerzos en incentivar la cultura del reciclaje y educar sobre la correcta separación de basuras. El API de REDO está basada principalmente en una solución centrada en la nube, en donde, se realizó su respectivo entrenamiento.

Objetivos

Objetivo general:

Desarrollar un API basada en técnicas de visión por computadora soportada en servicios Cloud que permita realizar el reconocimiento de residuos y determine la pertenencia de un elemento al grupo de residuos reciclables.

Objetivos específicos:

- Realizar un marco comparativo de los diferentes servicios cloud para la visión por computadora, con base en los criterios de arquitecturas de redes disponibles, exactitud de la predicción, costos de uso y curva de aprendizaje de cada plataforma a evaluar.
- Diseñar una arquitectura basada en técnicas de visión por computadora para el reconocimiento y clasificación de imágenes de residuos sólidos.
- Diseñar una API que haga uso de la arquitectura diseñada para el reconocimiento de residuos y que pueda ser consumida como servicio.
- Validar la arquitectura propuesta usando una base de datos real de residuos sólidos.

Marco Teórico

El aprendizaje de máquina (Machine Learning) hace parte de nuestro diario vivir aunque no nos demos cuenta de ello, la cámara de nuestros celulares ajustando la imagen para capturar las sonrisas, la publicidad que aparece en las redes sociales de los productos que estamos planeando comprar, los videojuegos ajustándose al comportamiento del jugador para hacer más interesante la partida, los bancos cuando nos notifican que podríamos estar teniendo un fraude en nuestra cuenta bancaria, todas estas tareas requieren de modelos de Machine Learning entrenados para ejecutar satisfactoriamente estas tareas, para lograr estos resultados, es necesario contar con personas expertas en el tema. Esto requiere a su vez bastante experimentación, tiempo y recursos. Con el fin de optimizar este proceso, surge una nueva rama de la inteligencia artificial llamada AutoML (Auto Machine Learning).

Según el teorema NFL (No Free Lunch) no es posible lograr un algoritmo que cuente buen desempeño en todas las tareas que tienen igual importancia, por esto, cada aspecto del proceso de diseño de un modelo de machine learning, como la extracción de características, selección del modelo y selección de algoritmos debe ser cuidadosamente configurado y para esto se cuenta con personas expertas en el área las cuales realizan el

diseño a través de ensayo y error, por lo cual, incluso teniendo la experticia necesaria, es necesario invertir mucho tiempo y recursos para obtener un buen resultado [1][2]. El Auto ML es la combinación de automatización y machine learning, su objetivo es automatizar el proceso de diseño de modelos de machine learning aún teniendo recursos computacionales limitados [1], dejando de esta forma el recurso humano de un lado del proceso. Este marco de trabajo permite así al factor humano, enfocarse en otro tipo de decisiones de negocio.

Uno de los trabajos que dirigió la atención de muchos hacia el Auto ML fue el de Zoph et al. [3] acerca del diseño de una red neuronal recurrente (RNN), la cual, siendo entrenada con la estrategia de entrenamiento por refuerzo logra encontrar la arquitectura con mejor precisión de manera automática (Figura 1). La RNN es entrenada en un ciclo: El controlador primero propone un modelo candidato y luego lo entrena para la tarea dada hasta converger, posteriormente, el controlador utiliza la precisión de la red como la señal "premio" para encontrar la mejor arquitectura posible, este proceso es repetido durante diferentes iteraciones.

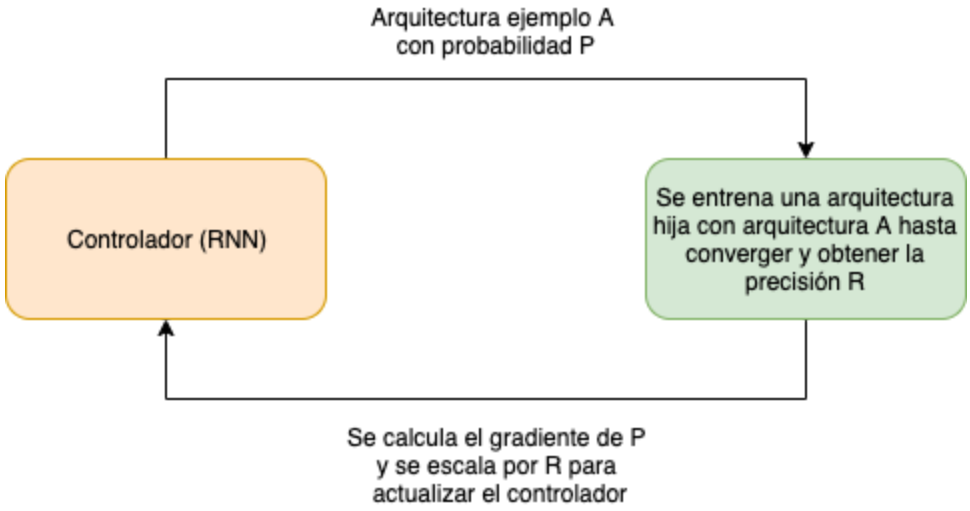


Figura 1: Una visión de de NAS utilizando entrenamiento por refuerzo [3]

Teniendo este trabajo como precedente, muchos otros acercamientos se enfocan en Neural Architecture Search (NAS), el cual busca generar una arquitectura de red neuronal robusta y de buen desempeño seleccionando y combinando diferentes componentes dentro de un espacio de búsqueda previamente definido.

Un ejemplo de esto es la estrategia propuesta por Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le y Jeff Dean [6] la cual busca disminuir el costo computacional del proceso de NAS, forzando a todos los sub-modelos a compartir parámetros sin afectar la precisión de estos.
El proceso de NAS puede ser entendido desde dos enfoques, el primero de ellos es la búsqueda de la estructura del modelo, entre las estructuras más comunes se encuentran:

estructuras completas [3][12], estructura basada en células [6], estructura jerárquica [13] y estructura basada en morfismo [14] entre otras. El segundo enfoque es la optimización de hiper parámetros (HPO), entre las técnicas más usadas para esto, se encuentran: entrenamiento por refuerzo (RL) [3][6][15][16][17], algoritmos evolutivos (EA) [18] y gradiente descendente (GD) [19][20][21]. Además de NAS, el AutoML también comprende otras técnicas que pueden ser divididas según el proceso de del Machine Learning (Figura 2): Preparación de los datos, ingeniería de características, generación del modelo y evaluación del modelo.

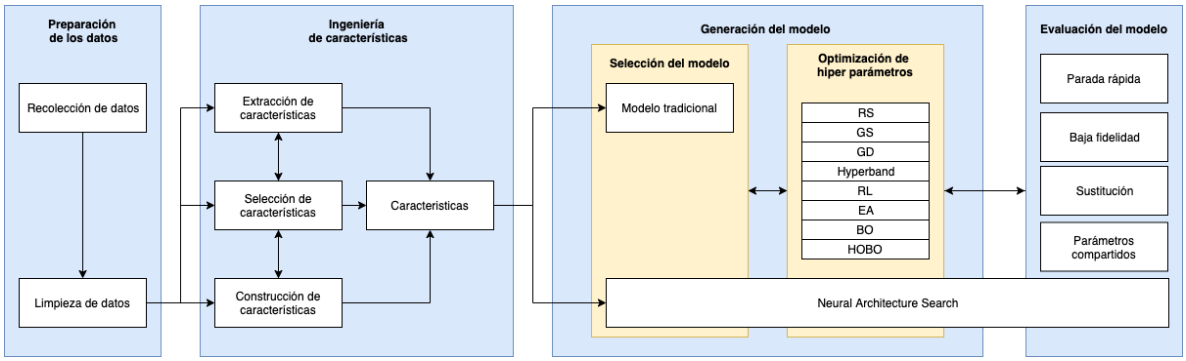


Figura 2: Proceso de creación de un modelo de machine learning [1]

Varias compañías ofrecen herramientas de Auto Machine Learning, las cuales permiten a los usuarios con poco conocimiento en esta área entrenar modelos de alta calidad y utilizarlos en sus aplicaciones desde la nube. Entre las cuales se encuentran: Google Cloud Platform (GCP), Amazon Web Services (AWS) y Microsoft Azure.

Google Cloud Platform

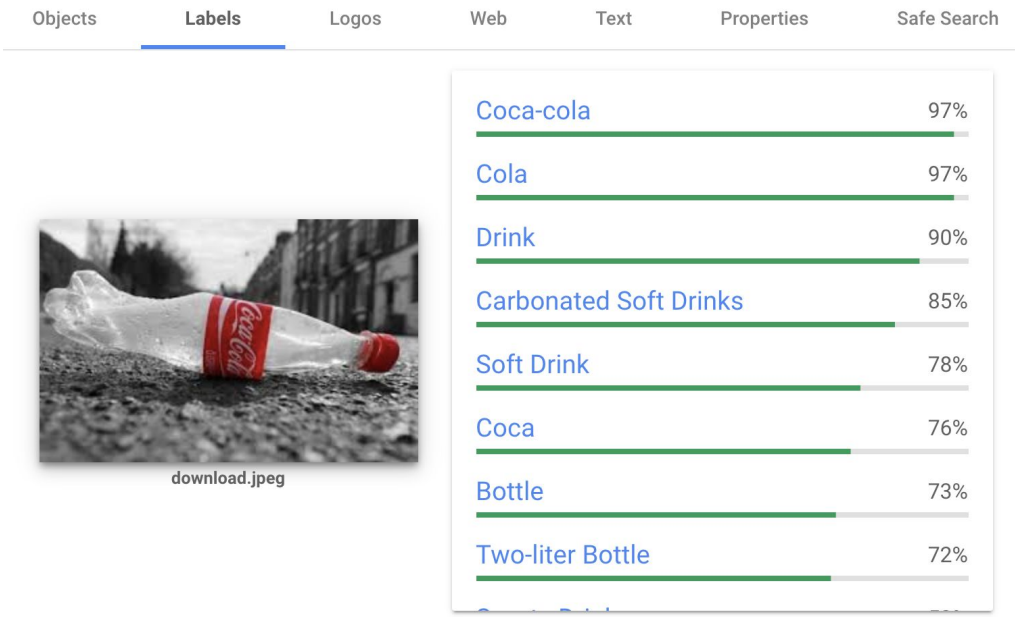
La plataforma de Google Cloud es una suite de computación en la nube que corre bajo la misma infraestructura que utiliza Google para sus productos de cara al usuario, tal como lo son el buscador Google y YouTube. Este provee infraestructuras como servicios (IaaS), plataformas como servicios (PaaS) y ambientes de computación serverless. Esta plataforma cuenta con más de 90 productos, entre los cuales se incluyen servicios de almacenamiento de datos, análisis de información y herramientas de machine learning, entre ellas Cloud Vision API [4] y AutoML las cuales son de particular interés para el desarrollo de este proyecto.

Cloud Vision API

Es una API que ofrece un modelo ya entrenado para el reconocimiento de imágenes, se consume vía REST, la cual recibe una imagen como entrada y nos brinda como respuesta información de la imagen tal como:

- Detección de características

- Detección de texto
- Detección de contenido explícito
- Detección de rostros
- Ubicación de puntos de referencia
- Detección de logos
- Descripción de propiedades de la imágen
- Descripción de entidades web relacionadas



AutoML

Esta herramienta permite realizar en la nube un proceso de Auto Machine Learning y posteriormente da la capacidad de consumir el modelo a través de una API rest.

En su capa gratuita Google Cloud Platform ofrece $300 dólares de crédito que se pueden utilizar en un periodo máximo de 12 meses.

Microsoft Azure

Microsoft Azure es un servicio de computación en la nube creado por Microsoft para crear, testear, desplegar y administrar aplicaciones y servicios en centros de datos manejados por Microsoft. Este provee Software como servicio (SaaS), plataforma como servicio (PaaS), e infraestructura como servicio (IaaS), cuenta con soporte para diversos lenguajes de programación, herramientas y frameworks, incluyendo no solo los propietarios por Microsoft sino también software de terceros.
Microsoft Azure ofrece más de 600 servicios, entre los cuales se encuentran herramientas para servicios de computación, servicios móviles, almacenamiento de datos, mensajería,

administración, machine learning, internet de las cosas, entre otros. en este trabajo haremos uso de la herramienta para visión por computadora ‘Custom Vision’.

Custom Vision

Custom Vision hace parte de la suite de herramientas que ofrece Microsoft Azure, permite realizar un proceso de Auto Machine Learning dentro de la plataforma Azure y exponerla posteriormente como un API [5].

Para el problema particular de reconocimiento y clasificación de imágenes el tipo de red neuronal las redes neuronales que se utilizan son las Redes Neuronales Convolucionales (CNN)

Metodología

La metodología bajo la cual se desarrollará el proyecto se dividirá en las fases expuestas a continuación.

a. Selección de servicios cloud

La fase 1 comprende la selección de los servicios cloud que serán comparados de acuerdo con los criterios establecidos en el objetivo 1.

Las plataformas cloud a evaluar deben cumplir con los siguientes requisitos:

- Ofrecer capa gratuita.
- Permitir cargar un dataset personalizado.

Algunas de las herramientas online que actualmente cumplen con estos requisitos son:

- Google Cloud Vision.
- Microsoft Azure Custom Vision.

Para el criterio de la curva de aprendizaje se tendrán en cuenta las siguientes pautas:

- La plataforma cuenta con documentación para su correcto uso.
- La plataforma tiene una interfaz intuitiva que genera una buena experiencia de usuario.

b. Desarrollo del modelo

En la fase 2, se desarrollará la arquitectura del modelo de predicción. En este caso, el reconocimiento de residuos estará basado en CNN.

Esta fase se desarrollará desde dos enfoques el primero de ellos haciendo uso de las plataformas cloud que ofrecen el servicio de Auto Machine Learning seleccionadas en la fase 1. Para el segundo enfoque se desarrollará de forma manual una arquitectura de CNN haciendo uso de transfer learning

c. Pruebas

En la fase 3, se realizarán pruebas de validación y verificación a la arquitectura implementada, para el caso de la validación se utilizará la técnica de validación cruzada.

El problema de reconocimiento de residuos puede ser entendido como un problema de clasificación de seis clases: vidrio, papel, cartón, plástico, metal y desechos no reciclables. Por lo tanto, se utilizaran las siguientes medidas de desempeño para la evaluación del modelo:

Precision: Posibilidades de que una imagen sea etiquetada correctamente.
Recall: Porcentaje de etiquetas asociadas correctamente.
Average Precision: Medida de rendimiento del modelo resumiendo las medidas 'Precision' y 'Recall' usando diferentes límites de decisión.

'Precision' y 'Recall' están basados en un límite de decisión de 0.5

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

precision = TP / (TP+FP)
recall = TP / (TP+FN)
accuracy = TP+TN / Total

Adicionalmente, se evaluará el sistema utilizando medidas de exactitud por clase y matrices
de confusión.

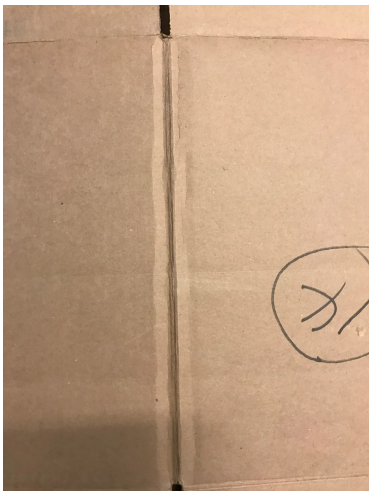
Base de datos

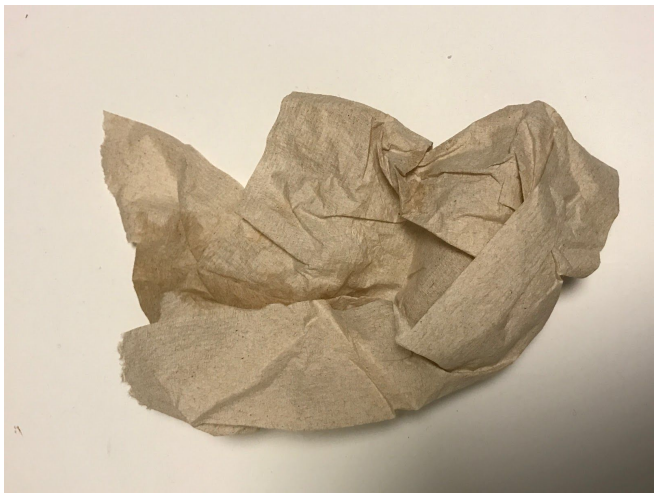
La base de datos que se usará para el desarrollo del proyecto está comprendida por 2527, correspondientes a seis clases de imágenes distribuidas de la siguiente forma:

- Vidrio: 501
- Papel: 594
- Cartón: 403

- Plástico: 482
- Metal: 401
- No reciclable: 137

Las imágenes fueron tomadas sobre un fondo blanco con luz tanto artificial como natural, el tamaño de de cada imagen es 512x384 pixeles. Los dispositivos utilizados para tomar las imágenes fueron: Apple iPhone 7 Plus, Apple iPhone 5S, and Apple iPhone SE.



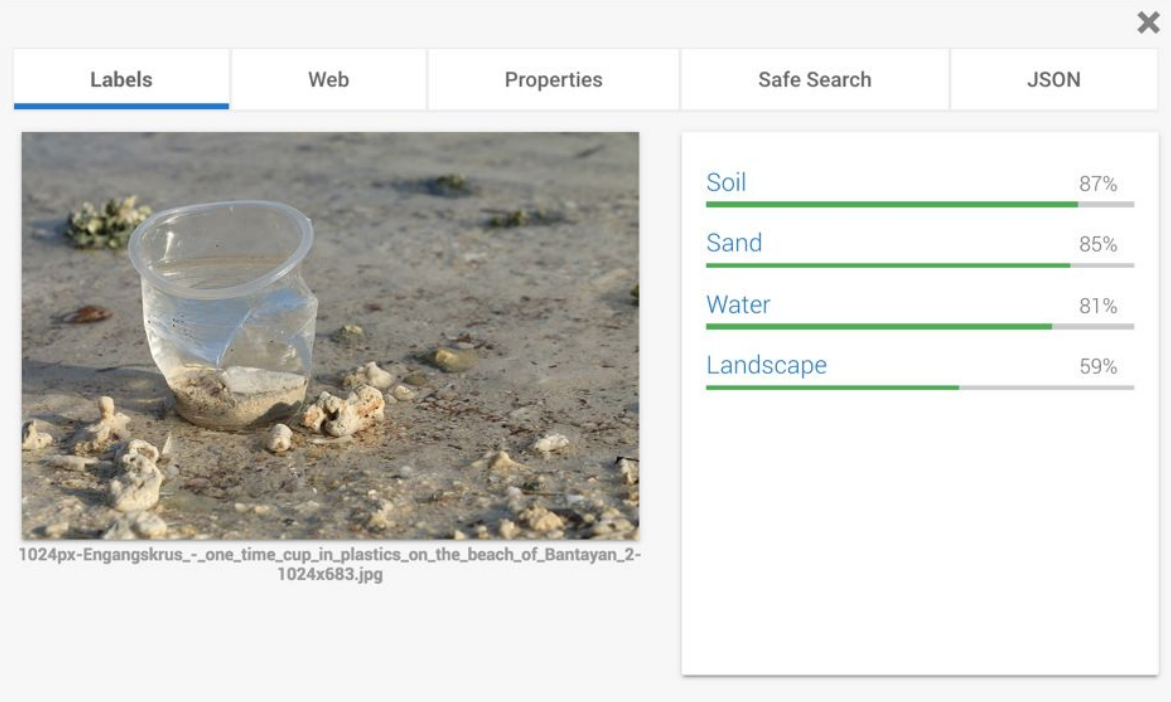



Muestras de imágenes del dataset

Esta base de datos, creada por Gary Thun y Mindy Jang está disponible de forma gratuita online [11].

Resultados y análisis

El primer prototipo de este proyecto fue realizado en una hackathon, para la cual se debía lograr un producto mínimo viable en un lapso de 16 horas. Debido a estas limitaciones de tiempo, la API de **Google Cloud Vision** fue la herramienta seleccionada para esta primera versión ya que permite obtener información inmediata de una imagen dada, sin realizar un previo entrenamiento, evitando a su vez el proceso de obtener o recolectar datos para este fin. Durante este proceso nos encontramos con una gran desventaja al utilizar esta herramienta y es el hecho de que **no está optimizada para nuestro problema particular de reconocimiento**, es decir, desechos. Ya que el API reconoce imágenes de manera genérica, mucha de la información que retorna es irrelevante al momento de decidir el recipiente correcto para depositar el desecho. Un ejemplo de esto, es los resultados obtenidos al enviar al API la imagen de un vaso de plástico tirado en la playa, las etiquetas obtenidas como “barro”, “tierra”, “agua”, “paisaje” no aportan información valiosa para identificar la correcta forma de reciclar un vaso de plástico, lo cual generó un bajo índice de precisión en la aplicación.



La segunda aproximación a la solución se realizó haciendo uso de plataformas cloud que ofrecen servicios de Auto ML, los criterios tomados en cuenta para la selección de las plataformas cloud a probar fueron:

- Ofrecer capa gratuita.
- Permitir cargar un dataset personalizado.

Después de considerar las herramientas web que cumplían con los criterios de selección establecidos, se seleccionaron 'Google Cloud Vision AutoML' y 'Microsoft Azure Custom Vision' para realizar el proceso de Auto ML. En ambas herramientas la única interacción del usuario se realiza al cargar los archivos, el resto del proceso es una caja negra, no se tiene acceso a ningún tipo de configuración de parámetros, arquitecturas, estrategias de validación ni tampoco se puede observar la arquitectura del modelo resultante. Cuando el proceso de entrenamiento ha finalizado, se pueden visualizar las métricas resultantes del algoritmo, las cuales se exponen a continuación.

Las métricas arrojadas por las plataformas fueron:

Google AutoML

Las métricas globales arrojadas por la plataforma al realizar el entrenamiento fueron:

Average Precision	97,4%

Precision	95,60%
Recall	87%

Las métricas por clase fueron:

	Precision	**Recall**
Cardboard	97,70%	93,50%
Glass	96,40%	93%
Metal	93,90%	92%
Paper	92%	89,10%
Plastic	94,30%	73,30%
Trash	100%	58,30%

Se puede evidenciar que para la clase *trash*, la cual contiene la menor cantidad de imágenes (un ~80% menos en comparación con la cantidad de imágenes de las otras clases) se obtuvo un sobreajuste del modelo.

Microsoft Azure Custom Vision

Las métricas globales arrojadas por la plataforma son:

Average Precision	96,00%
Precision	95,70%
Recall	97,00%

Las métricas por clase fueron:

	Precision	**Recall**
Cardboard	98,70%	93,80%
Glass	99,00%	95%
Metal	95,20%	98%
Paper	96%	98,30%
Plastic	94,00%	96,90%
Trash	89%	82,10%

Con ambas plataformas se obtuvo muy buenos resultados en poco tiempo, ya que sus entrenamientos no tardaron más que un par de horas y de forma inmediata se tenía la posibilidad de consumir los modelos desde cualquier aplicación, haciendo uso de servicios REST.

Para realizar una comparación entre los resultados obtenidos con procesos de Auto ML vs. El trabajo manual de una persona diseñando el modelo a través de prueba y error como se realiza clásicamente, en la tercer aproximación a la solución del problema de clasificación de residuos se diseña de forma manual la red neuronal a través de dos aproximaciones: la primera entrenando una red neuronal convolucional desde cero y la segunda haciendo uso de fine-tuning, el cual permite hacer uso de una red neuronal convolucional pre-entrenada para clasificación de imágenes y posteriormente ser ajustada para nuestro problema en particular, es decir, la clasificación de residuos.

Estos modelos se diseñan haciendo uso de TensorFlow y Keras.

Fine Tuning

Para el diseño del modelo de fine tuning se realizaron de forma manual varios experimentos, combinando diferentes arquitecturas base, clasificadores e hiper parámetros y así observar con cual combinación se logran obtener mejores métricas.

Las arquitecturas base a probar fueron:

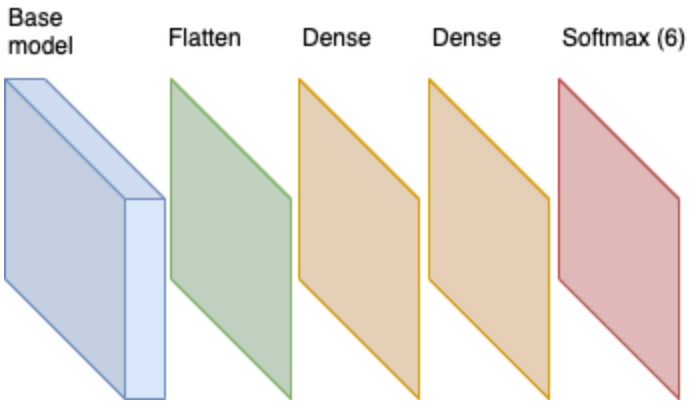
- VGG16 [7]
- VGG19 [7]
- MobileNet [8]
- MobileNetV2 [9]

Una de las limitantes encontradas para realizar este proyecto es el tamaño de la base de datos, ya que se cuentan con muy pocas imágenes. Se decide probar estas arquitecturas ya que son de las más pequeñas ofrecidas por Keras, ya que en arquitecturas más robustas como ResNet [10] se requiere un conjunto de entrenamiento mayor para lograr buenos resultados.

Los hiper parámetros a iterar fueron:

- Número de épocas (10, 20 y 30)
- Tamaño de batch (8, 16 y 32)

El clasificador a probar fue:



Variando el número de unidades entre 128, 256 y 512 en ambas capas densas.

Los experimentos fueron realizados en la plataforma Google Cloud Platform haciendo uso de una máquina virtual con las siguientes características:

- **Tipo de máquina:** n1-standard-4 (4 CPU virtuales, 15 GB de memoria)
- **Sistema operativo:** ubuntu 18.04
- **GPU:** 1 x NVIDIA Tesla P4
- **Disco Duro:** 40GB

Para la partición de los datos de entrenamiento, validación y test se hace uso de la estrategia de partición estratificada, la cual divide los datos en iguales porcentajes por clase, para este caso, el dataset fue dividido de la siguiente forma:

- 80% entrenamiento
- 10% test
- 10% validación

Tras varias iteraciones combinando las variables anteriormente mencionadas se exponen a continuación aquellas arquitecturas con las que se obtuvo mejores resultados globales:

	Arquitectura					**Resultados**			
	Modelo Base	**Tamaño de batch**	**Épocas**	**Unidades (FC 1)**	**Unidades (FC 2)**	**Average Precision**	**Precision**	**Recall**	**Accuracy**
1	VGG19	8	10	256	128	0.73	0.76	0.62	0.70
2	VGG19	16	20	256	128	0.68	0.62	0.56	0.65
3	VGG19	16	30	256	128	0.74	0.71	0.65	0.68
4	VGG19	8	10	128	64	0.67	0.60	0.59	0.65
5	VGG19	16	10	128	64	0.70	0.73	0.57	0.65

Resultados por clase para cada experimento en el mismo orden expuesto en la tabla anterior:

Arquitectura 1:

	Precision	Recall
Cardboard	88,00%	54,00%
Glass	56,00%	87%
Metal	76,00%	55%
Paper	89%	88,00%
Plastic	70,00%	63,00%
Trash	88%	26,00%

Arquitectura 2:

	Precision	Recall
Cardboard	73,00%	49,00%
Glass	72,00%	60%
Metal	54,00%	83%
Paper	65%	95,00%
Plastic	88,00%	38,00%
Trash	21%	11,00%

Arquitectura 3:

	Precision	Recall
Cardboard	84,00%	67,00%
Glass	54,00%	88%
Metal	78,00%	65%
Paper	77%	85,00%
Plastic	92,00%	36,00%
Trash	42%	52,00%

Arquitectura 4:

	Precision	Recall
Cardboard	57,00%	77,00%
Glass	56,00%	93%

Metal	74,00%	72%
Paper	93%	55,00%
Plastic	58,00%	55,00%
Trash	25%	4,00%

Arquitectura 5:

	Precision	**Recall**
Cardboard	95,00%	46,00%
Glass	76,00%	34%
Metal	71,00%	62%
Paper	72%	90,00%
Plastic	50,00%	89,00%
Trash	75%	22,00%

Al igual que lo observado en los modelos generados por los sistemas de AutoML, la clase 'Trash' es con la que se presentan más problemas, en este caso, presenta para todos los casos una de las precisiones más bajas en la predicción, otro fenómeno observado a partir de las matrices de confusión generadas al realizar las pruebas de validación, se evidencia que los errores en la clase 'Paper' son en la mayoría de los casos clasificados como 'Cardboard'.

Para el caso de los experimentos realizados con las redes mobileNet y mobileNetV2, se encontró que para todos los casos solo una clase estaba siendo entrenada y el total de las predicciones eran hechas en esta.

Los resultados obtenidos en este proceso no fueron tan buenos como se esperaban, y no alcanzaron ni siquiera a igualar aquellos obtenido a través del proceso de AutoML, siendo la mayor precisión obtenida con fine tuning de 76% mientras que los modelos generados en herramientas cloud tuvieron precisiones entre 93% y 98%.

El código fuente de los experimentos realizados así como los resultados completos pueden ser encontrados en https://github.com/ManuCastrillonM/redo.

Discusión

El estado del arte más reciente del proyecto que busca abordar el mismo problema de clasificación de residuos es el realizado por los estudiantes de la universidad de Stanford Gary Thung y Mindy Yang [24], los cuales desarrollaron el dataset utilizado para el entrenamiento y validación del modelo de este proyecto. Uno de los logros alcanzados en

el desarrollo del presente proyecto fue mejorar la precisión en el desempeño lograda en el trabajo anteriormente mencionado, el cual tiene una precisión de 63%, en comparación con el 72% logrado con el modelo diseñado. El modelo generado por el proceso de Auto machine learning logró superar la precisión de ambos modelos diseñados de forma manual.

Conclusiones

Los procesos de Auto Machine Learning presentaron resultados mucho mejores que aquellos obtenidos diseñando el modelo de forma manual, permitiendo a su vez realizar su integración en un entorno real de forma mucho más fácil y rápida ya que exponen un API rest sin necesidad de realizar trabajo extra. Los procesos de diseño de modelos de machine learning requieren un alto grado de experticia y requieren mucho tiempo y recursos para encontrar el modelo más óptimo. A pesar de que no se obtuvieron los resultados esperados realizando el diseño del modelo de forma manual, se logró una muy buena comprensión del proceso de reconocimiento y clasificación de imágenes haciendo uso de redes neuronales convolucionales, así como una comprensión teórica de algunos de los algoritmos de Auto machine learning.

Referencias bibliográficas

[1] Xin, Zhao, Chu, and Xiaowen, “AutoML: A Survey of the State-of-the-Art,” arXiv.org, 14-Aug-2019. [Online]. Available: https://arxiv.org/abs/1908.00709. [Accessed: 15-Oct-2019].

[2] Yao, Wang, Chen, Yuqiang, Dai, Qiang, Yang, Yang, and Yu, “Taking Human out of Learning Applications: A Survey on Automated Machine Learning,” arXiv.org, 17-Jan-2019. [Online]. Available: https://arxiv.org/abs/1810.13306. [Accessed: 15-Oct-2019].

[3] Zoph, Barret, and Q. V., “Neural Architecture Search with Reinforcement Learning,” arXiv.org, 15-Feb-2017. [Online]. Available: https://arxiv.org/abs/1611.01578. [Accessed: 15-Oct-2019].

[4] “Vision AI | Derive Image Insights via ML | Cloud Vision API | Google Cloud,” Google. [Online]. Available: https://cloud.google.com/vision/. [Accessed: 15-Oct-2019].

[5] “Visual Intelligence Made Easy,” Custom Vision - Home. [Online]. Available: https://www.customvision.ai/. [Accessed: 15-Oct-2019].

[6] Pham, Hieu, Guan, Zoph, Barret, and Jeff, “Efficient Neural Architecture Search via Parameter Sharing,” arXiv.org, 12-Feb-2018. [Online]. Available: https://arxiv.org/abs/1802.03268. [Accessed: 15-Oct-2019].

[7] Simonyan, Karen, Zisserman, and Andrew, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv.org, 10-Apr-2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed: 15-Oct-2019].

[8] Howard, Z. Andrew G., Chen, Bo, Dmitry, Wang, Weijun, Weyand, Tobias, Marco, Adam, and Hartwig, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv.org, 17-Apr-2017. [Online]. Available: https://arxiv.org/abs/1704.04861. [Accessed: 15-Oct-2019].

[9] Sandler, Mark, Howard, Andrew, Zhu, Zhmoginov, Andrey, Chen, and Liang-Chieh, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” arXiv.org, 21-Mar-2019. [Online]. Available: https://arxiv.org/abs/1801.04381. [Accessed: 15-Oct-2019].

[10] Zhang, Ren, Sun, and Jian, “Deep Residual Learning for Image Recognition,” arXiv.org, 10-Dec-2015. [Online]. Available: https://arxiv.org/abs/1512.03385. [Accessed: 15-Oct-2019].

[11] Garythung, “garythung/trashnet,” GitHub. [Online]. Available: https://github.com/garythung/trashnet. [Accessed: 15-Oct-2019].

[12] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “SMASH: ONESHOT MODEL ARCHITECTURE SEARCH THROUGH HYPERNETWORKS,” p. 22.

[13] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, “Hierarchical representations for efficient architecture search,” in ICLR, p. 13

[14] T. Chen, I. Goodfellow, and J. Shlens, “Net2net: Accelerating learning via knowledge transfer,” arXiv preprint arXiv:1511.05641, 2015.

[15] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, “Learning transferable architectures for scalable image recognition.” [Online]. Available: http://arxiv.org/abs/1707.07012. [Accessed: 15-Oct-2019].

[16] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, “Practical block-wise neural network architecture generation.” [Online]. Available: http://arxiv.org/abs/1708.05552.[Accessed: 15-Oct-2019].

[17] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing neural network architectures using reinforcement learning,” vol. ICLR. [Online]. Available: http://arxiv.org/abs/1611.02167. [Accessed: 15-Oct-2019].

[18] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” vol. 10, no. 2, pp. 99–127. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/106365602320169811. [Accessed: 15-Oct-2019].

[19] H. Liu, K. Simonyan, and Y. Yang, “DARTS: Differentiable architecture search.” [Online]. Available: http://arxiv.org/abs/1806.09055. [Accessed: 15-Oct-2019].

[22] K. Ahmed and L. Torresani, “MaskConnect: Connectivity learning by gradient descent.” [Online]. Available: http://arxiv.org/abs/1807.11473. [Accessed: 15-Oct-2019].

[23] R. Shin, C. Packer, and D. Song, “DIFFERENTIABLE NEURAL NETWORK ARCHITECTURE SEARCH,” p. 4.

[24] Thung Gary, Yang Mindy, “Classification of Trash for Recyclability Status” [Online]. Available: http://cs229.stanford.edu/proj2016/poster/ThungYang-ClassificationOfTrashForRecyclabilityStatus-poster.pdf. [Accessed: 15-Oct-2019].