La Inteligencia Artificial como herramienta de identificación de penitentes en el volcán nevado Coropuna: Primeros ensayos

Joshua Iparraguirre[1], Pablo Masías[2], Jose Úbeda[3]

[1]*Instituto Geológico, Minero y Metalúrgico, Lima, Perú – iparraguirrea.joshua@gmail.com*
[2]*Observatorio Vulcanológico del Perú, Yanahuara, Arequipa*
[3]*Universidad Complutense de Madrid, Madrid, España*

Palabras clave: Penitentes, Inteligencia Artificial, Coropuna

Introducción

El volcán nevado Coropuna (15°32’S, 72°39’O, 6377 m) se ubica al oeste de los Andes peruanos, al norte de la Zona Volcánica Central (ZVC). Debido a su topografía, permite la acumulación de precipitación sólida, que en determinados años por los fenómenos ENSO se intensifican, formando estructuras conocidas como penitentes. Los penitentes se forman por ablación diferencial, con el punto de rocío siempre debajo de 0°C, lo que significa que la nieve se sublima en lugar de fundirse. Además, múltiples penitentes altos pueden crear un microclima que desarrolle una circulación convectiva del calor, empujando el aire más cálido hacia arriba durante el día y provocando el atrapamiento del aire más frío por la noche (Yoshikawa et al., 2020).

La Inteligencia Artificial (IA), en los últimos años, es la disciplina, que no sólo viene siendo aplicado en ramas de ciencias de la computación o ingeniería de sistemas. Sino que, con el desarrollo de nuevos algoritmos englobados en módulos, también se está aplicando en diferentes ramas de la ingeniería (ambiental, geología, salud, etc.) por su facilidad de uso, fácil aprendizaje y su menor consumo de memoria para el procesamiento de datos masivos. La IA tiene como finalidad que un algoritmo programado haga el trabajo que haría cualquier persona, previamente habiéndole enseñado a razonar e identificar el objeto de estudio. Bajo este criterio, la IA puede identificar personas, animales, cosas; siempre y cuando nuestro algoritmo aprenda las diferentes formas que puede adquirir el objeto de estudio. Así mismo, si lo complementamos con el uso de sistemas de detección remoto complejo, se puede monitorear diferentes elementos que se encuentren en zonas muy alejadas y/o que conlleve demasiado recurso humano para su análisis.

Los algoritmos se caracterizan por seguir un procedimiento, paso a paso, para realizar una determinada tarea. Primero se definen los valores iniciales como son el tipo de captura de datos (fotografía, imagen o vídeo), si la identificación será de izquierda a derecha o viceversa, mediciones de distancia entre objetos, centroides, etc. Posteriormente, se adentra en el aprendizaje autónomo. Actualmente, existen módulos en python para la identificación de diferentes objetos comúnmente encontrados en nuestro día a día. A su vez, los mismos módulos instruyen, a través de su documentación, cómo se pueden crear algoritmos para objetos particulares. Ya que, exclusivamente la identificación de Penitentes nunca antes se ha realizado, el objetivo ha sido elaborar los algoritmos personalizados para la identificación y reconocimiento de Penitentes.

Metodología

Para una correcta identificación del objeto a analizar, es necesario contar con una amplia colección de fotografías. Un número promedio apropiado sería entre 500 a 800 fotografías. Debido a que aún no hay un sistema remoto instalado, se realizó este primer ensayo con 100 fotografías tomadas con una GoPro durante salidas a campo en años anteriores hacia el volcán nevado Coropuna. Se realizaron 03 pasos, detallados a continuación:

Paso 1: Preparación de imágenes con el módulo YOLO

Primero, se recolectaron todas las fotografías disponibles que contengan penitentes. Incluyendo aquellas tomadas en diferentes ángulos y direcciones. Esto permitió mejorar la calidad del algoritmo; ya que, amplió la data de reconocimiento de las diferentes geometrías.

La preparación consistió en clasificar aquellas fotografías con alto brillo, otras que se encuentren movidas, aquellas que presenten doble visión, etc. La finalidad de esta clasificación fue importante, porque adaptó el algoritmo, lo cual es útil cuando las capturas no tienen buena resolución. Luego, se descargó el software LABELLMG (Versión 1.8.0), que requirió de entrada la carpeta que contienen las fotografías clasificadas. El software contiene dos módulos con los cuales etiquetar el objeto a identificar: Yolo y

Pascal. Ambas metodologías son eficientes, pero en este caso, se decidió escoger Yolo, porque genera matrices numéricas en archivos textos que son de fácil exportación.

Seleccionada la metodología, el software separó un 80% de las fotografías (80 del total) para el entrenamiento y el 20% restante para testeo (20 del total). De las 80 fotografías destinadas al entrenamiento, se agruparon las diferentes geometrías de los penitentes y etiquetaron con el nombre de "Penitente" (Fig. 1).

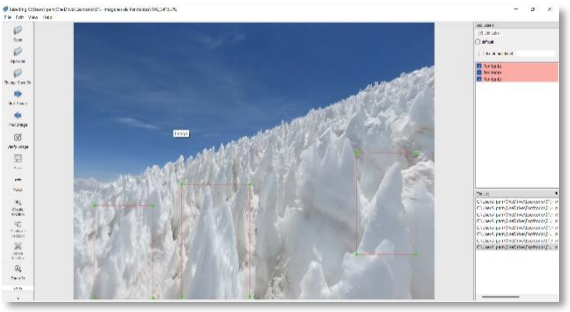

Fig. 1: Ejemplo de etiquetado de penitentes. Al centro la fotografía seleccionada para entrenamiento. Lado derecho, las etiquetas sombreadas en rojo.

Toda fotografía está compuesta de píxeles; por ende, el software brindó un archivo texto con las coordenadas superior e inferior de los penitentes que han sido señalados en el entrenamiento de cada una de las fotografías usadas.

Finalmente, todas las fotografías con etiquetado, junto a sus archivos textos de coordenadas se comprimieron en una sola carpeta (.zip), terminando así la preparación de los datos con Yolo.

Paso 2: Creación del algoritmo de entrenamiento para identificación de penitentes

Se procedió a utilizar el conjunto de datos comprimidos (.zip) para el entrenamiento de identificación de penitentes, utilizando Google Colab, debido a que este servidor cuenta con una Unidad de Procesamiento Gráfica (*GPU, por sus siglas en inglés*) necesaria para procesamientos de datos masivos como el que se requirió para el presente trabajo.

Se cargaron todas las fotografías comprimidas en el servidor Google Drive; y se conectó virtualmente con el servidor Google Colab. Conectados ambos servidores, se procedió al entrenamiento:

Primero, se descargó la red neuronal de pre-entrenamiento conocido como Darknet. Esta red neuronal permitió escalar las fotografías y procesarlas de manera automatizada. Luego, se compilaron los archivos de la Darknet necesarios para el funcionamiento según la GPU de Google Colab. Esto permitió, la habilitación del módulo especializado OpenCV, así como de la GPU – NVIDIA.

Luego, se configuró la Darknet para su funcionamiento; ya que, al descargar y compilar los archivos, estos reconocen una serie de elementos por defecto (personas, animales, cosas, etc.). Por lo tanto, la configuración consistió en que sólo se reconozca una clase: Penitentes.

Luego, se utilizó una línea de código que permitió descomprimir el archivo zip y almacenarlos en la memoria interna de la Darknet para que puedan procesarse. Una vez realizado el desempaquetado, se utilizaron todos los archivos textos, generándose una matriz de todas las coordenadas superior e inferior de las selecciones realizadas de los penitentes. Es en esta sección donde el algoritmo empezó a asignar según los espectros que píxeles son considerados como penitentes y cuales no. Esto lo hace posible a través de una serie de iteraciones para asignar valores según la coloración del pixel. Es el paso que requirió el mayor tiempo de procesamiento, ya que se tuvo que desglosar uno a uno todos los pixeles que conforman la fotografía, asignarles un valor y correlacionarlos con la etiqueta a cada una de las 80 fotografías asignadas al entrenamiento.

Finalmente, después del procesamiento se generaron 02 archivos, que fueron denominados de acuerdo a la etiqueta trabajada: Yolo_training _Penitente_1000 y Yolo_training; ambos con extensión "weight". Ambos archivos son originados dependiendo de la cantidad de archivos, el primero porque superó las 1000 iteraciones y por lo tanto es más completo; pero a la vez, necesita de mayor memoria de procesamiento. Mientras que, el segundo es todo lo contrario. Dependiendo de la cantidad de fotografías y el hardware que se posea, se escoge uno. En este primer ensayo se utilizó el segundo archivo.

Paso 3: Creación del algoritmo de testeo para identificación de penitentes

Se creó un algoritmo propio para el testeo. La finalidad de este algoritmo fue comprobar si el archivo de entrenamiento de extensión "weight" (apartado 2.2) reconoce, señaliza y cuantifica qué son penitentes y cuantificarlos.

Lo primero, consistió en cargar el archivo de extensión "weight". Luego, se englobó en una variable la clase de análisis denominada "Penitente", con la finalidad de que pueda ser llamado en cualquier línea de código y en cualquier computadora que haga

uso del script. Luego, se brindó acceso a la carpeta que tiene las imágenes no usadas (el 20% reservado para el testeo).

Finalmente, se configuró los parámetros de visualización. Por ejemplo, qué figura geométrica servirá de delimitador (rectángulo en este caso), la coloración, identificación de derecha a izquierda de los penitentes y si se reconocen muchas geometrías, no se superpongan, evitando así cargar la fotografía con demasiados rectángulos (Fig. 2).

Fig. 2: Líneas de código para la configuración de visualización.

Por último, se realizaron unas configuraciones finales: Todas las coordenadas de los penitentes identificados se almacenaron en un archivo CSV y se pueda trabajar con fotografías y vídeos.

Resultados

El algoritmo se configuró para la identificación de penitentes. De las 20 fotografías de testeo, fueron 2 en las que no se pudo determinar geometría alguna. El problema pudo deberse a que los penitentes eran muy pequeños. Por lo tanto, se identificó la primera limitante: no se dispone de la cantidad suficiente de fotografías con diversas geometrías.

Por otro lado, el algoritmo YOLO, por defecto contiene pre establecido la identificación de personas, animales y cosas que comúnmente se ve en el día a día. Pero, el algoritmo modificado para la exclusiva identificación de penitentes, fue capaz de discretizar a personas que aparecían en las fotos (**Figura 3**). Esto otorga una mejor calidad de datos, y la confianza de que, si en la captura de fotografías provenientes del sistema remoto aparecieran otros elementos, el algoritmo es capaz de sólo capturar penitentes.

En las 18 fotografías restantes, el algoritmo no fue capaz de identificar todos los penitentes. Esto permitió determinar la segunda limitante: la falta de datos de entrenamiento; esto quiere decir que contar con 100 fotografías no es suficiente; por lo que, aún no se cuenta una larga data que capturen a penitentes en diferentes direcciones y en diferentes ángulos.

Fig. 3: Identificación de penitente, incluyendo una discretización de la persona en la fotografía.

La cantidad necesaria aproximada debería ser entre 500 a 800 fotografías, lo cual conllevaría a una tercera limitante: la necesidad de contar con una potente Unidad de Procesamiento Gráfica (GPU). El procesamiento de testeo con 100 imágenes conllevó cerca a las 12 horas; por lo que, si se requiere conseguir la mejora en los resultados, la máquina tendría que ser capaz de procesar por encima de las 24 horas. Lamentablemente, la GPU proporcionada por el servidor Google Colab sólo funciona por 12 horas continuas, posterior a ese tiempo se reinicia y pierde el procesamiento.

Agradecimientos

Un agradecimiento especial al Observatorio Vulcanológico del INGEMMET por brindar los accesos a los datos utilizados y al FONDECYT por financiar este proyecto, necesario para monitorear la evolución de los penitentes con una visión a futuro de su rol que cumplirán con el cambio climático.

Referencias

Bradski, G. (2000). The OpenCV Library. Dr. Dobb' s Journal of Software Tools.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R

Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

Van Rossum, G., & Drake Jr, F. L. (1995). Python reference manual. Centrum voor Wiskunde en Informatica Amsterdam.

Yoshikawa, K., Úbeda, J., Masías, P., Pari, W., Apaza, F., Vasquez, P., Ccallata, B., Concha, R., Luna, G., Iparraguirre, J., Ramos, I., De la Cruz, G., Cruz, R., Pellitero, R., & Bonshoms, M. (2020). Current thermal state of permafrost in the southern Peruvian Andes and potential impact from El Niño–Southern Oscillation (ENSO). Permafrost and Periglacial Processes, 31(4), 598–609. Portico. https://doi.org/10.1002/ppp.2064