UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA EN GEOLOGÍA, MINAS, PETRÓLEO Y AMBIENTAL

CARRERA DE GEOLOGÍA

SOFTWARE APLICADO

NOVENO SEMESTRE

TEMA:

PROYECTO SEMESTRAL

“Clasificaciones y descripciones automáticas de deslizamientos mediante Machine Learning”

INTEGRANTES:

Andrade Carlos

Mejía Kevin

Naranjo Mariuxi

Noroña Kevin

Junio 2021 – Septiembre 2021

ÍNDICE

1. Planteamiento del Problema

Los Fenómenos de remoción en masa o también conocidos como deslizamientos, ocurren cuando grandes masas de tierra o rocas caen pendientes abajo, ya sea por una ladera o talud artificial. Los encargados de la caracterización de dichos fenómenos son los geólogos que, a través de incursiones al campo, herramientas satelitales, softwares especializados y su vasta experiencia, son capaces de construir mapas de susceptibilidad a fenómenos de remoción en masa. La cartografía de deslizamientos es un paso sumamente importante en la realización de mapas de susceptibilidad, el cual conlleva un arduo trabajo de identificación, para la posterior construcción de inventarios de deslizamientos, los cuales generalmente se los realiza mediante imágenes satelitales, por tal motivo se dificulta su rápida detección y posterior análisis, ya que muchas veces son áreas sumamente extensas a cubrir, obligando a destinar largos periodos de tiempo para esta actividad. La identificación automatizada de los rasgos morfológicos asociada a deslizamientos mediante algoritmos de Deep Learning, facilitaría el trabajo previo de identificación para la construcción inventarios de deslizamientos. De esta manera se pretende aplicar redes neuronales convolucionales y recurrentes, mediante un Dataset de imágenes satelitales, que ayude a agilizar el trabajo de identificación de deslizamientos, y que además presente una serie de parámetros que ayude a caracterizar el fenómeno en cuestión, mediante anotaciones automáticas. Una vez obtenido los inventarios de deslizamientos se puede proceder a la elaboración de mapas de susceptibilidad, el cual permite a las autoridades o gobiernos de turno, diseñar un plan de ordenamiento territorial efectivo, prohibiendo la construcción de viviendas o proyectos importantes, en zonas de peligro. Gracias a esta propuesta se espera que se acorte el tiempo para la identificación de estos fenómenos a comparación de las técnicas convencionales, lo cual es sumamente fundamental puesto que se puede ahorrar varios recursos y a su vez aumentar su efectividad.

2. Objetivos

2.1 General

- Aplicar Redes neuronales convolucionales y recurrentes para la identificación y caracterización de deslizamientos mediante imágenes satelitales.

2.2 Específicos

- Elaborar un Dataset extenso y confiable de diferentes tipos de deslizamientos a través de imágenes satelitales obtenidas mediante Google Earth y que incluye parámetros como: tipo, actividad, dirección, área y perímetro.

- Diseñar un modelo mediante Deep Learning utilizando redes neuronales convolucionales y recurrentes que describa las principales características de los deslizamientos.
- Demostrar la efectividad de las anotaciones automáticas resultantes, ante los procesos analíticos convencionales utilizados en la identificación de deslizamientos.

3. Antecedentes o estudios previos

Bhatt et al. Llevaron a cabo una investigación de Deep Learning para la clasificación de deslizamientos por medio de redes neuronales convolucionales. Para ello recopiló imágenes de buena calidad de varias fuentes como imágenes de Google y diferentes sitios web. Creó un conjunto de datos de buena calidad de unas 260 imágenes para entrenar y probar su modelo de Deep Learning propuesto. El 80% del conjunto de datos se utiliza como un conjunto de entrenamiento y el 20% restante se utiliza como un conjunto de pruebas. Por medio del Deep Learning extrajo características necesarias para detectar deslizamientos de tierra, clasificándolos en Deslizamiento y No deslizamiento. La primera capa utiliza la función de activación ReLu la cual produce no linealidad en el modelo, la segunda capa ocupa la función denominada max-pooling que toma el mayor valor máximo en cada bloque, Por último, el razonamiento complejo en la red se realiza en la tercera capa tomando todas las neuronas de la capa anterior y la vincula a cada neurona que tiene. Al final obtuvo un modelo que era capaz de reducir las imágenes a una forma que sea más fácil de procesar sin perder características que son significativas para obtener mejores resultados, además de compararlos con otros algoritmos similares, dando mejores resultados los propuestos por el autor.

Prakash et al. Desarrollo una nueva estrategia para mapear deslizamientos de tierra con una red neuronal convolucional generalizada, la cual consistía en compilar imágenes satelitales, topografía, en con junto con un inventario de deslizamientos local correspondientes a fechas antes y después de sismos que detonarán varios movimientos de masa cercanos a la zona de afectación. Para ello se seleccionaron 7 zonas en donde recopilaron la información pertinente antes mencionada. Trabajaron con las primeras 4 zonas (A,B,C,D) a manera entrenamiento para afinar el modelo una vez listo lo comprobaron en las 3 zonas restantes (E,F,G). Los resultados que arrojaron fueron precisos en los lugares donde se poseía imágenes de buena resolución (6m/pixel) y donde exista poca nubosidad, al contrario, los peores resultados fueron para las imágenes con resolución baja (30m/pixel). Se destacó también que, la falta de un inventario local para las zonas de prueba no afectó a los resultados finales, dando a notar que este no es un parámetro fundamental para el modelo.

Un tercer trabajo correspondiente a Zhengjing et al. (2020), quienes realizaron la: “Prevención de deslizamientos mediante aprendizaje automático”, centraron su estudio hacia la

predicción de zonas vulnerables, en las que probablemente se puedan presentar deslizamientos de tierra. La metodología de trabajo, estuvo basada en el desarrollo de redes neuronales convolucionales aplicada a imágenes satelitales. Igualmente, se abordaron temas relacionados al desarrollo de sistemas de alerta de deslizamientos, mediante la implementación de sistemas fiables para predecir el comportamiento a corto plazo de los deslizamientos de tierra, con el objetivo de prevenir eventos repentinos. El estudio en cuestión se relaciona con nuestro proyecto, ya que propone el aprendizaje automático para la detección de deslizamientos en una zona en particular, mediante redes neuronales convolucionales. Sin embargo, nuestro proyecto propone la utilización de anotaciones automáticas, para describir ciertos parámetros importantes para la evaluación del riesgo, como son: dirección de movimiento, área, volumen de material deslizado, entre otros, que ayuden a tener datos de primera mano, para la toma de decisiones.

An Buiv et al. Determinó la utilidad de los algoritmos de transformación de imágenes para detectar la ubicación de deslizamientos de tierra en imágenes de satélite. En el proceso a estudiarse para se utiliza una red neuronal de convolución para clasificar las imágenes de satélite que contienen deslizamientos de tierra. A partir de imágenes de deslizamientos clasificados, con el fin de identificar con precisión los deslizamientos bajo diferentes condiciones de iluminación, este trabajo propone un algoritmo de transformación. Descomposición en modo empírico bidimensional (H-BEMD) para determinar la región y el tamaño del deslizamiento. Después de que se detecta la ubicación del deslizamiento de tierra, descubrimos el cambio de tamaño del deslizamiento de tierra en función de diferentes momentos**.**

El algoritmo propuesto ha demostrado la combinación entre una clasificación de imágenes a través del aprendizaje profundo y el algoritmo de transformación adaptativa de CNN y H-BEMD.

(H. Wang et. al, 2020) “Identificación de deslizamientos de tierra mediante aprendizaje automático”. Este trabajo propone un método aplicando machine learning y deep learning para identificar deslizamientos de tierra mediante geodatabases integradas. En estas geodatabases se recopilan datos relacionados con los deslizamientos de tierra, incluidos datos topográficos, datos geológicos y datos relacionados con las precipitaciones. Dividen en 3 las geodatabases en donde distribuyen deslizamientos de tierra recientes, deslizamientos de tierra relictos y una base de datos conjunta de deslizamientos de tierra la cual incluye deslizamientos recientes y relictos. En cada base de datos, se entrenan y comparan varios modelos de aprendizaje automático y aprendizaje profundo basados en LR (Regresión Logística), SVM (Máquina de Vectores de Soporte), RF (Bosques Aleatorios), Impulsador y CNN (Red Neuronal Convolucional) para evaluar su rendimiento. El presente estudio se diferencia por la utilización de imágenes

satelitales como base de datos e implementar un modelo de Redes Neuronales Recurrentes para la identificación de deslizamientos de tierra.

4 Recursos Computacionales

4.1 Hardware

Para la realización de este proyecto se utilizó una laptop de marca Dell Inc, modelo Inspiron 14-3467 con un sistema operativo de Microsoft Windows 10 Pro. Esta laptop cuenta con un procesador Intel Core i5-7200 CPU @2.50 GHz, 2701 Mhz de 63 bits y una tarjeta de video de Intel HD Graphics 620. Posee las siguientes unidades almacenamiento; RAM de 8GB y un Disco C de 930GB.

Tabla 1. *Características del hardware*

COMPUTADORA	
Tipo	Laptop
Sistema	Dell Inc.
Modelo	Inspiron 14-3467
Sistema Operativo	Microsoft Windows 10 Pro
CPU	
Tipo SO	64 bits procesador x64
Procesador	Intel Core i5-7200 CPU @2.50 GHz, 2701 Mhz
Núcleos	2
Tarjeta de video	Intel HD Graphics 620
UNIDADES DE ALMACENAMIENTO	
RAM	8GB
Disco C:	930 GB

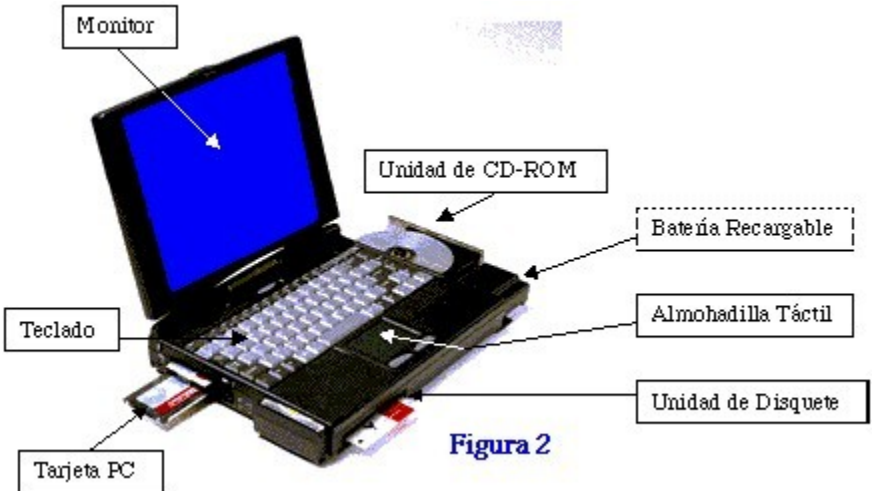


Ilustración 1. *Hadware de una computadora Portátil*

- **Características principales**

Ocupamos dispositivos locales de la computadora o laptop, para poder capturar las imágenes, para ello se guardarán en una carpeta, se clasificó las carpetas en base a los tipos de deslizamientos ya sea por Flujos, Caídas de Rocas, Rotacionales, Reptación de Suelos.

Adicional se utilizó la plataforma de la nube Google Colaboraty, el mismo que nos permitió subir la información ordenada y de manera sistemática.

Hardware Google Colaboraty

Para experimentar con Machine Learning necesitamos capacidades de computación adecuadas. Comúnmente es difícil permitirse una PC de escritorio o una computadora portátil con las capacidades adecuadas de hardware debido a su precio. Una buena alternativa es usar plataformas de nube pública de uso gratuito como Google Colab.

Tabla 2. *Características del hardware en Google Colab*

Google Colaboratory	
CPU	
Procesador	Intel(R) Xeon(R) CPU @ 2.30GHz
Núcleos	1
UNIDADES DE ALMACENAMIENTO	
RAM	13 GB
Disco C:	49 GB
GPU	
Nvidia Tesla K80	
Disco C:	64 GB

4.2 Software

4.2.1 Sistema Operativo

Tabla.3. - Características del Sistema Operativo utilizado

	Windows 10
Escritorio	Contiene accesos directos, permite configuración de fondo
Inicio	Botón de inicio parte izquierda inferior Despliegue de opciones Cajón de búsqueda Personal (Cambiar de usuario) Apagar, suspender o reiniciar
Barra de tareas	Horizontal parte inferior Incluye accesos directos Bandeja de estado parte derecha inferior (fecha, hora, red, batería, antivirus, idioma)

Ventanas	Cinta de opciones: portapapeles, organizar, nuevo, abrir, seleccionar Barra superior indica localización, crea una ruta Cajón de búsqueda Margen izquierdo jerarquía o unidades de almacenamiento Parte Central se despliega las carpetas y archivos Margen derecho superior opciones de minimizar, cambiar tamaño y cerrar
Unidades físicas y lógicas	1 Unidad física, Disco duro 2 Unidades lógicas, Disco C y D
Iconos	Accesos directos Figuras indican de que trata el programa

4.2.2 Lenguaje de programación Python

Este lenguaje de programación es uno de los más utilizados en proyectos de Machine Learning debido a su fácil uso y versatilidad haciendo que sea un lenguaje amigable y de amplio crecimiento.

Editor de programación Jupyter Notebook

Jupyter Notebook es una interfaz web de código abierto que permite la inclusión de texto, video, audio, imágenes, así como la ejecución de código a través del navegador en múltiples lenguajes. Esta ejecución se realiza mediante la comunicación con un núcleo (Kernel) de cálculo. Aunque en principio, el equipo de desarrollo de Júpiter Notebook incluye por defecto únicamente el núcleo de cálculo Python, el carácter abierto del proyecto ha permitido aumentar el número de núcleos disponibles, incluyendo, por ejemplo, Octave, Julia, R, Haskell, Ruby, C/C++, Fortran, Java, SageMath, Scala, Colab, Matlab y Mathematica. Esta interfaz, agnóstica del lenguaje (de ahí su nombre al unir 3 de los lenguajes de programación de código abierto más utilizados en el ámbito científico: Ju-lia, Python y R), puede suponer por tanto una estandarización para mostrar el contenido de cursos científicos, sin encontrarse limitado a la adopción de un ´único lenguaje. (Cabrera et al, 2014)

Google Colab

Google Colaboratory es un entorno gratuito de Jupyter Notebook que no requiere configuración y que se ejecuta completamente en la nube. Colab te permite escribir y ejecutar código de Python en un navegador, con las siguientes particularidades: sin configuración requerida, acceso gratuito a GPU, facilidad para compartir.

4.2.3 Principales librerías

Keras: es una Api para entrenar redes neuronales de alto nivel con Deep Learning, desarrollada bajo lenguaje de programación Python, se puede ejecutar sobre TensorFlow, CNTK o Theano. Keras permite implementar modelos complejos de Deep Learning solo se debe definir las capas que constituirán la red neuronal. En la detección de objetos, Keras trabaja con modelos que toman las imágenes como entrada y da probabilidades de clasificación como salida. (Keras Documentation, 2018)

TensorFlow: es una librería de python, desarrollada por Google, para realizar cálculos numéricos mediante diagramas de flujo de datos. Esto puede chocar un poco al principio, porque en vez de codificar un programa, codificaremos un grafo. Los nodos de este grafo serán operaciones matemáticas y las aristas representan los tensores (matrices de datos multidimensionales). Con esta computación basada en grafos, TensorFlow puede usarse para deep learning y otras aplicaciones de cálculo científico. (Quan et al, 2017)

Matplotlib. es una librería para generar gráficas a partir de datos contenidos en listas, vectores, en el lenguaje de programación Python y en su extensión matemática NumPy. Puede generar gráficos, histogramas, espectros de potencia, gráficos de barras, gráficos de error, gráficos de dispersión, etc., con solo unas pocas líneas de código. Matplotlib se puede utilizar en scripts de Python, el shell de Python e IPython, servidores de aplicaciones web y varios kits de herramientas de interfaz gráfica de usuario. (J.D.Hunter, 2007)

NumPy: es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, incluidas matemáticas, lógicas, manipulación de formas, clasificación, selección, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más. (Comunidad NumPy, 2020)

Glob: El módulo Glob devuelve una lista posiblemente vacía de nombres de ruta de acceso que coincidan con pathname, que debe ser una cadena que contenga una especificación de ruta de

acceso. pathname puede ser absoluto (como) o relativo (como), y puede contener comodines de estilo Shell.(Python, 2021)

5. Metodología

La metodología a desarrollar en el proyecto se presenta a continuación mediante un diagrama de flujo.

a. Diagrama de flujo

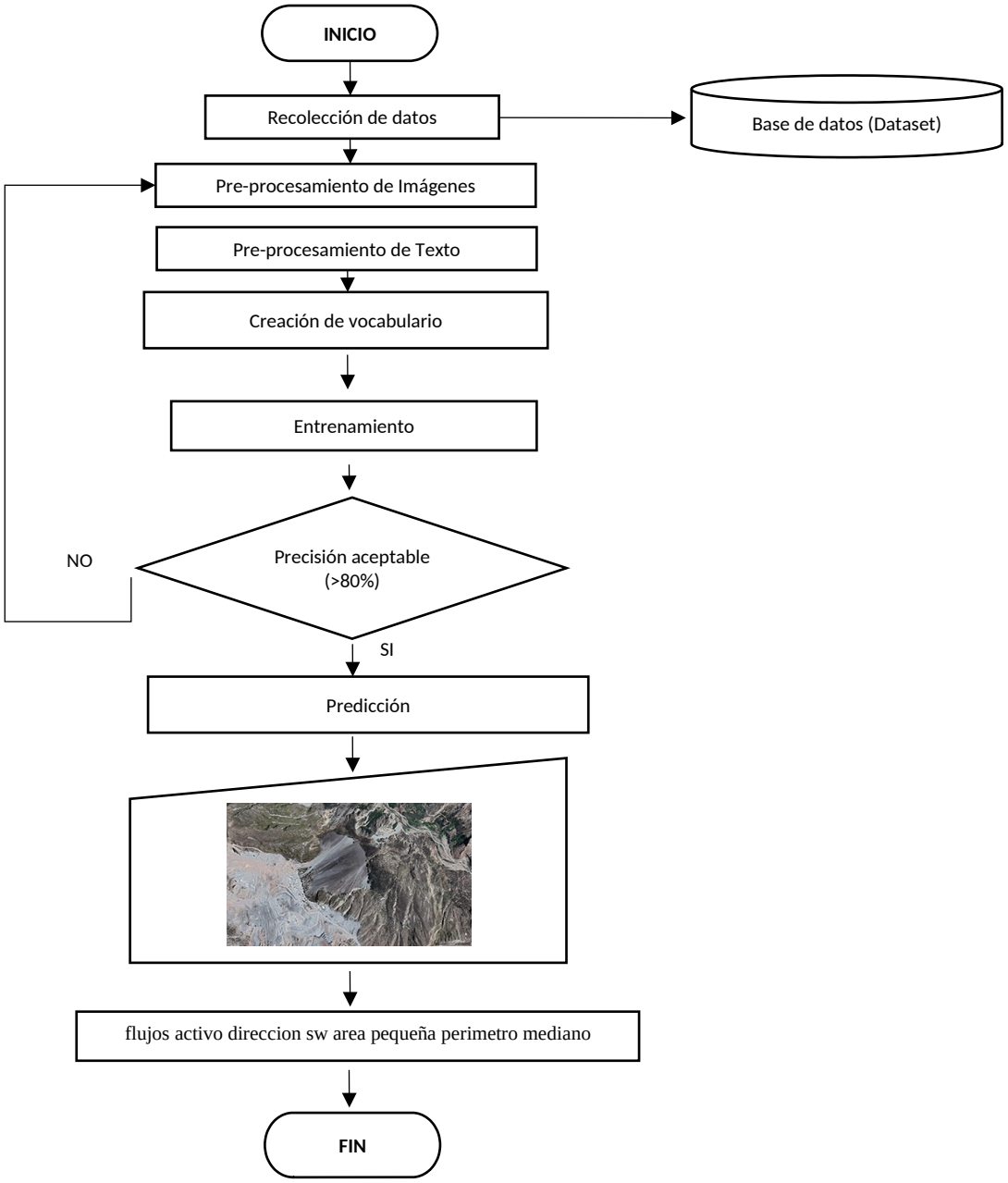


***Ilustración 2**. Diagrama de flujo en base a la simbología ANSI de la metodología implementada para la clasificacion y descripciones automáticas*

b. DATASET

El insumo principal para este proyecto se lo obtuvo mediante la descarga de imágenes de Google Earth con una resolución inicial de 1024 x 768 pixeles, con un total de 1000 imágenes con extensión .jpg que, subdivididas para cuatro tipos de deslizamientos, como son: 1) Deslizamiento rotacional, 2) Reptación de suelos, 3) Deslizamientos tipo flujo y 4) Caída de detritos, da un total de 250 imágenes para cada tipo de deslizamiento. Estás imágenes fueron extraídas de varias partes del mundo, entre los cuales figuran: Ecuador, Argentina, Perú, Colombia, Chile, EEUU, Bosnia y España, principalmente.

Las imágenes fueron organizadas en base a 1 directorio principal y 4 subdirectorios, en la nube de Google Drive, en donde, el directorio principal contiene la agrupación de varias carpetas destinadas a un fin particular, como se explicará en la sección de pre-procesamiento.

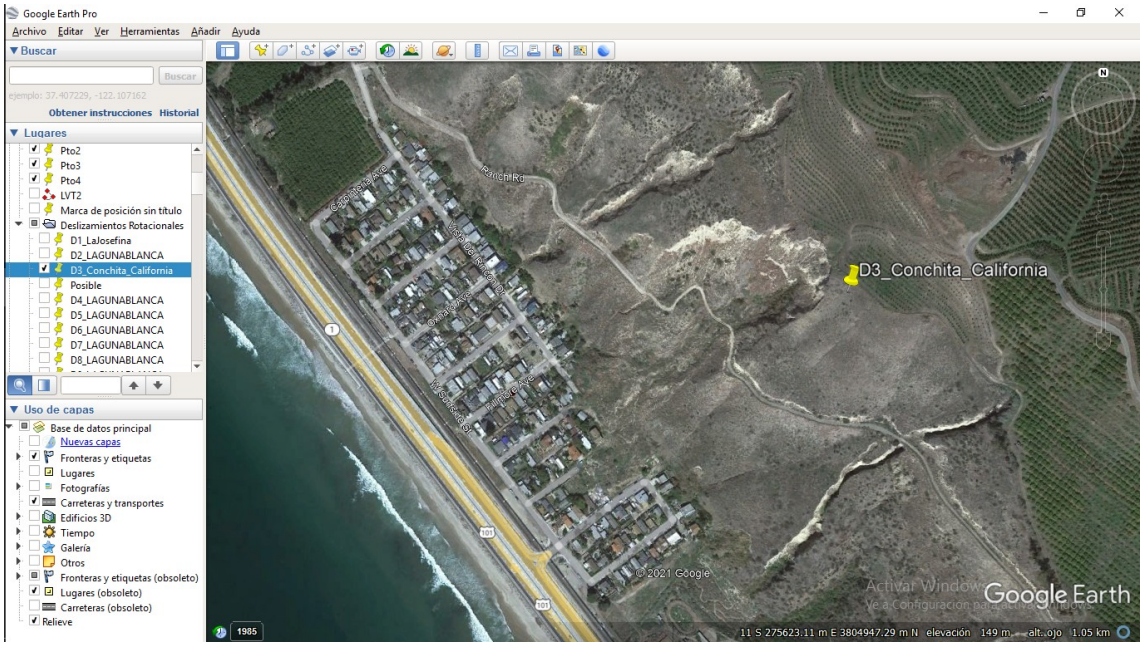
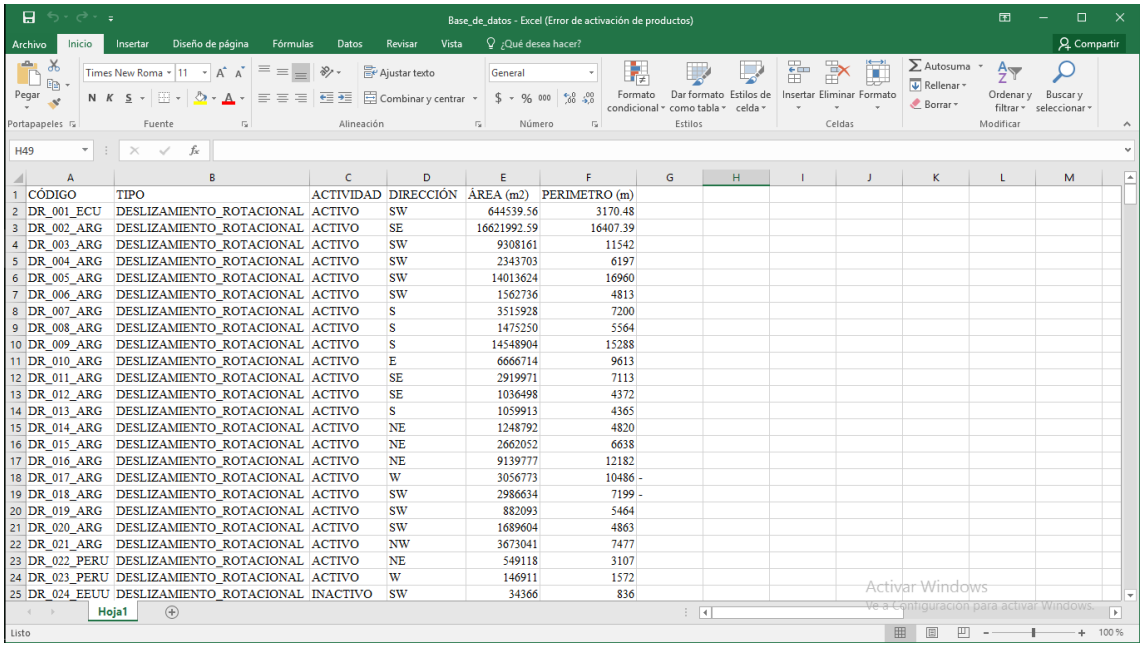


Ilustración 3. *Deslizamiento rotacional "La Conchita – EEUU"*

Por otro lado, la elaboración de las descripciones se las realizó analizando cada una de las imágenes obtenidas anteriormente, en donde se recopiló varias características, como son: el tipo de deslizamiento, perímetro, área, actividad y dirección. Por medio de las herramientas propias del software Google Earth Pro se obtuvo los valores de área y perímetro en metros, para cada tipo de deslizamiento.

Todas estas descripciones se las organizó a partir del programa de ofimática Excel, el cual nos permitió crear un listado de cada deslizamiento con su respectivo código de identificación y la descripción correspondiente, como se muestra a continuación:



CÓDIGO	TIPO	ACTIVIDAD	DIRECCIÓN	ÁREA (m2)	PERIMETRO (m)
DR_001_ECU	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	644539.56	3170.48
DR_002_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SE	16621992.59	16407.39
DR_003_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	9308161	11542
DR_004_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	2343703	6197
DR_005_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	14013624	16960
DR_006_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	1562736	4813
DR_007_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	S	3515928	7200
DR_008_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	S	1475250	5564
DR_009_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	S	14548904	15288
DR_010_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	E	6666714	9613
DR_011_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SE	2919971	7113
DR_012_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SE	1036498	4372
DR_013_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	S	1059913	4365
DR_014_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	NE	1248792	4820
DR_015_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	NE	2662052	6638
DR_016_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	NE	9139777	12182
DR_017_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	W	3056773	10486
DR_018_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	2986634	7199
DR_019_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	882093	5464
DR_020_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	SW	1689604	4863
DR_021_ARG	DESLIZAMIENTO_ROTACIONAL	ACTIVO	NW	3673041	7477
DR_022_PERU	DESLIZAMIENTO_ROTACIONAL	ACTIVO	NE	549118	3107
DR_023_PERU	DESLIZAMIENTO_ROTACIONAL	ACTIVO	W	146911	1572
DR_024_EEUU	DESLIZAMIENTO_ROTACIONAL	INACTIVO	SW	34366	836

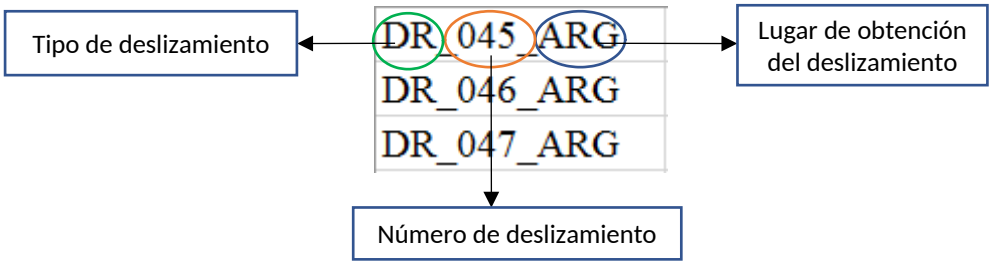


Ilustración 4. *Estructura de descripción de la data set de imágenes.*

i. Pre-Procesamiento de Imágenes

División

Una vez recopilada la base de datos de imágenes, para su pre-procesamiento, las mismas fueron clasificadas en 1 subdirectorio principal, 2 subdirectorios secundarios y dentro de estos últimos por 4 subdirectorios adicionales, que se describen a continuación:

1) Directorio principal: El mismo contiene a todos los subdirectorios, los cuales almacenan las imágenes de todos los tipos de deslizamientos descritos anteriormente, con una totalidad de 1000 imágenes.
2) Subdirectorios secundarios: Fueron destinos con el fin de contener 2 carpetas denominadas Train y Test, las mismas que contienen 800 y 200 imágenes respectivamente.
3) Subdirectorios adicionales: Dentro de la capeta de Train contiene 4 subcarpetas por cada tipo de deslizamiento, es decir, una carpeta para rotacional, flujo, caídas de rocas y reptación de suelos, las mismas que contienen 200 imágenes por cada uno. Para la

carpeta Test, de igual manera, contiene 4 subcarpetas por cada tipo de deslizamiento, las mismas que contienen 50 imágenes cada una.

Esta división se la realizó con el objetivo de que el modelo propuesto procese la mayor cantidad de imágenes posible, para que el aprendizaje sea extendido y no solamente limitado a un tipo de imagen específica que no permita que el modelo aprenda de la mejor manera para tener un resultado exitoso. Por otra parte, las imágenes fueron extraídas de tal manera que se tenga un conjunto de imágenes balanceado, como se puede ver a continuación:

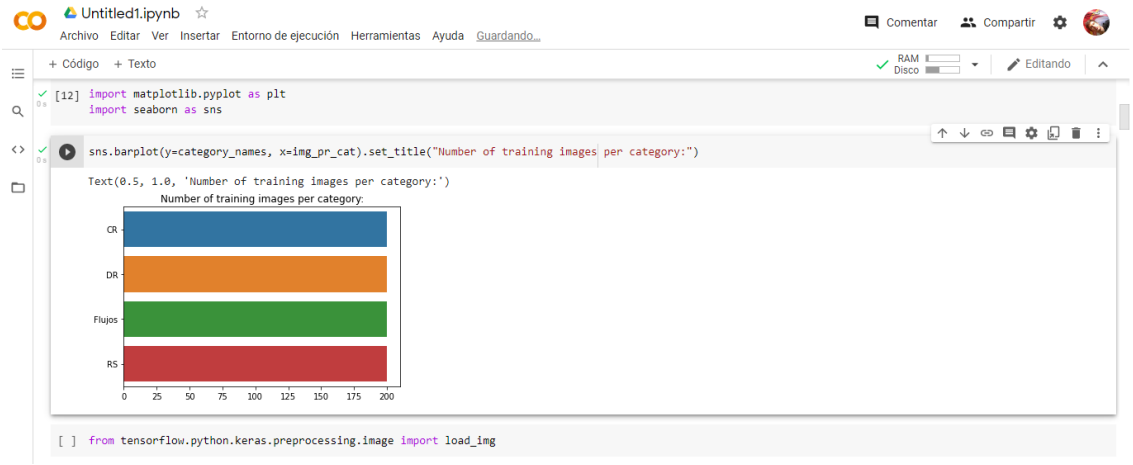


Ilustración 5. *Conjunto de datos balanceados para el subdirectorio Train*

A continuación, para designar las carpetas separadas anteriormente, se definió la ruta de las imágenes de train y test, se realiza esta división porque necesitamos imágenes para poder entrenar el modelo, y el conjuto de test se utiliza para tener una idea del rendimiento final de un modelo, dentro del conjunto de entrenamiento tenemos un subconjunto de validación y se utiliza para evaluar el rendimiento de modelos con diferentes valores de hiperparámetros, la proporción de la división q se utilizó fue de 80/20, obeteniendo así 800 imágenes para el entrenamiento y 200 para prueba.

Primeramente, se comenzó por crear una lista con todos y cada uno de los directorios correspondientes a las imágenes de Train, mediante la instrucción “glob.glob”, como se puede observar a continuación:

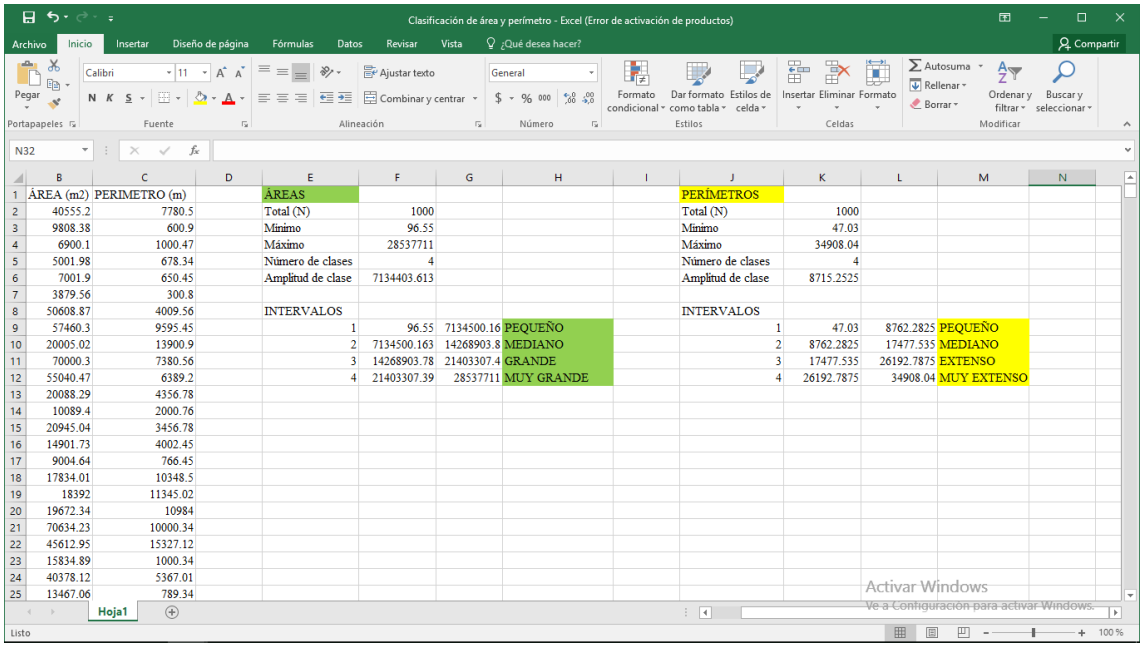
Ilustración 61*. Importación de imágenes Train, mediante Glob.glob.*

Después, las imágenes fueron procesadas directamente en Google Colab mediante la estructura del modelo convulucional, el mismo que nos permite re escalar la imagen mediante la aplicación de la operación denominada “Maxpooling 2D” y/o aplicando la operación “ZeroPadding2D”, como se podrá observar en el modelo convolucional.

ii. Pre-procesamiento de Texto

El pre-procesamiento del texto fue llevado a cabo en dos aspectos diferentes, una modificación en el programa de ofimática Excel y directamente mediante algoritmos en Google Colab. El primero principalmente consistió en la modificación de caracteres de una sola letra (N, S, E, W), a una estructura de palabras, puesto que el modelo recurrente (explicado posteriormente), no las considera como tal, por lo que se procedió a realizar un cambio en el programa de ofimática Excel, por las pablas norte, sur, este y oeste, respectivamente.

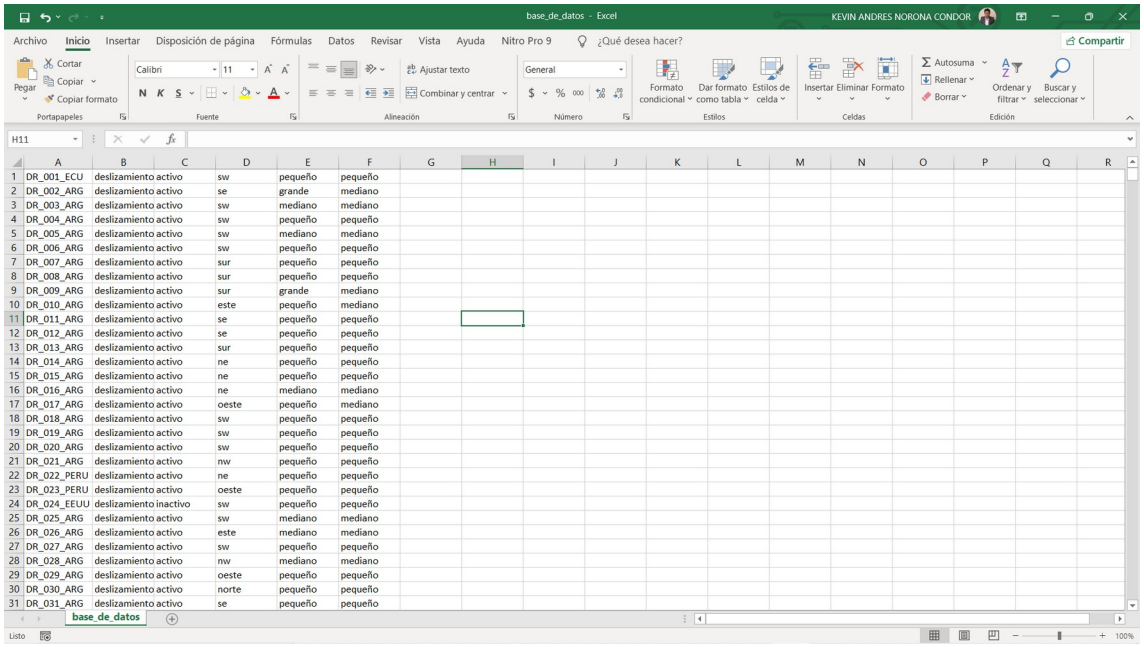
En cuanto a los valores de área y perímetro al ser estos de tipo numérico, consistía en un problema para el modelo recurrente, ya que el mismo está diseñado para el procesamiento de texto, por lo cual se procedió a realizar una clasificación mediante estadística, tomando como base los datos de área y perímetro de todos los deslizamientos, con un total de números de clase igual a 4, en donde se calculó la amplitud de clase, y con todo aquello se asignaron las categorías de: Muy grande, grande, mediano y pequeño. Todo esto para facilitar el reconocimiento y resultados de predicción del modelo. Así mismo, se procedió con los datos perímetro, como se observa a continuación:



	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ÁREA (m2)	PERIMETRO (m)		ÁREAS					PERÍMETROS				
2	40555.2	7780.5		Total (N)	1000				Total (N)	1000			
3	9808.38	600.9		Mínimo	96.55				Mínimo	47.03			
4	6900.1	1000.47		Máximo	28537711				Máximo	34908.04			
5	5001.98	678.34		Número de clases	4				Número de clases	4			
6	7001.9	650.45		Amplitud de clase	7134403.613				Amplitud de clase	8715.2525			
7	3879.56	300.8											
8	50608.87	4009.56		INTERVALOS					INTERVALOS				
9	57460.3	9595.45		1	96.55	7134500.16	PEQUEÑO		1	47.03	8762.2825	PEQUEÑO	
10	20005.02	13900.9		2	7134500.163	14268903.8	MEDIANO		2	8762.2825	17477.535	MEDIANO	
11	70000.3	7380.56		3	14268903.78	21403307.4	GRANDE		3	17477.535	26192.7875	EXTENSO	
12	55040.47	6389.2		4	21403307.39	28537711	MUY GRANDE		4	26192.7875	34908.04	MUY EXTENSO	
13	20088.29	4356.78											
14	10089.4	2000.76											
15	20945.04	3456.78											
16	14901.73	4002.45											
17	9004.64	766.45											
18	17834.01	10348.5											
19	18392	11345.02											
20	19672.34	10984											
21	70634.23	10000.34											
22	45612.95	15327.12											
23	15834.89	1000.34											
24	40378.12	5367.01											
25	13467.06	789.34											

Ilustración 7. *Reclasificación de datos de área y perímetro*

Con todo este proceso y cambiando a la vez a toda la descripción en palabras en minúscula, se obtuvo la siguiente base de datos:



	A	B	C	D	E	F
1	DR_001_ECU	deslizamiento activo		sw	pequeño	pequeño
2	DR_002_ARG	deslizamiento activo		se	grande	mediano
3	DR_003_ARG	deslizamiento activo		sw	mediano	mediano
4	DR_004_ARG	deslizamiento activo		sw	pequeño	pequeño
5	DR_005_ARG	deslizamiento activo		sw	mediano	mediano
6	DR_006_ARG	deslizamiento activo		sw	pequeño	pequeño
7	DR_007_ARG	deslizamiento activo		sur	pequeño	pequeño
8	DR_008_ARG	deslizamiento activo		sur	pequeño	pequeño
9	DR_009_ARG	deslizamiento activo		sur	grande	mediano
10	DR_010_ARG	deslizamiento activo		este	pequeño	mediano
11	DR_011_ARG	deslizamiento activo		se	pequeño	pequeño
12	DR_012_ARG	deslizamiento activo		se	pequeño	pequeño
13	DR_013_ARG	deslizamiento activo		sur	pequeño	pequeño
14	DR_014_ARG	deslizamiento activo		ne	pequeño	pequeño
15	DR_015_ARG	deslizamiento activo		ne	pequeño	pequeño
16	DR_016_ARG	deslizamiento activo		ne	mediano	mediano
17	DR_017_ARG	deslizamiento activo		oeste	pequeño	mediano
18	DR_018_ARG	deslizamiento activo		sw	pequeño	pequeño
19	DR_019_ARG	deslizamiento activo		sw	pequeño	pequeño
20	DR_020_ARG	deslizamiento activo		sw	pequeño	pequeño
21	DR_021_ARG	deslizamiento activo		nw	pequeño	pequeño
22	DR_022_PERU	deslizamiento activo		ne	pequeño	pequeño
23	DR_023_PERU	deslizamiento activo		oeste	pequeño	pequeño
24	DR_024_EEUU	deslizamiento inactivo		sw	pequeño	pequeño
25	DR_025_ARG	deslizamiento activo		sw	mediano	mediano
26	DR_026_ARG	deslizamiento activo		este	mediano	mediano
27	DR_027_ARG	deslizamiento activo		sw	pequeño	pequeño
28	DR_028_ARG	deslizamiento activo		nw	mediano	mediano
29	DR_029_ARG	deslizamiento activo		oeste	pequeño	pequeño
30	DR_030_ARG	deslizamiento activo		norte	pequeño	pequeño
31	DR_031_ARG	deslizamiento activo		se	pequeño	pequeño

Ilustración 8. *Base de datos de texto ajustado*

Posteriormente, el libro de Excel fue guardado como un archivo de texto separada por espacios, con la codificación de tipo “UTF-8”, la misma que después nos permitirá formar los captions o descripciones. Realizado este proceso, el archivo de tipo txt., fue cargada al Google Drive, para su posterior pre-procesamiento en el cuaderno de programación de Google Colab.

El segundo pre-procesamiento de texto consistió en la depuración de cualquier error que pudiera haber en la base de datos, eliminando caracteres especiales, puntuaciones y números, directamente mediante instrucciones en el cuaderno de programación de Google Colab, además de también asignar los tokens, para posteriormente guardar todo el proceso en un archivo de txt., depurado como se muestra a continuación:

```
[ ] filename = "/content/drive/My Drive/new_model/descripciones.txt"
    doc = load_doc(filename)
    print(doc[:500])

    DR_001_ECU.jpg  deslizamiento rotacional activo    direccion sw        area      pequeña    perimetro pequeño
    DR_002_ARG.jpg  deslizamiento rotacional activo    direccion se        area      grande     perimetro mediano
    DR_003_ARG.jpg  deslizamiento rotacional activo    direccion sw        area      mediana    perimetro mediano
    DR_004_ARG.jpg  deslizamiento rotacional activo    direccion sw        area      pequeña    perimetro pequeño
    DR_005_ARG.jpg  deslizamiento rotacional activo    direccio
```

```
[ ]  def load_descriptions(doc):
         mapping = dict()
         # process lines
         for line in doc.split('\n'):
             # dividir línea por espacio en blanco
             tokens = line.split()
             if len(line) < 2:
                 continue
             # tome el primer token como la identificación de la imagen, el resto como la descripción
             image_id, image_desc = tokens[0], tokens[1:]
             # extraer el nombre de archivo de la identificación de la imagen
             #image_id = image_id.split('.')[0]
             # convertir tokens de descripción de nuevo a cadena
             image_desc = ' '.join(image_desc)
             #crea la lista si es necesario
             if image_id not in mapping:
                mapping[image_id] = list()
             #store description
                mapping[image_id].append(image_desc)
         return mapping
```

```
[ ]  descriptions = load_descriptions(doc)
     print('Loaded: %d ' % len(descriptions))

     Loaded: 1000
```

***Ilustración 9**. Generación de Tokens.*

Mediante la aplicación de los “tokens”, se logró que el programa identifique al primer token como la identificación de la imagen (id), y al segundo como toda la descripción de la misma.

Posteriormente, se procedió con la depuración, como se indicó anteriormente y, como se puede observar a continuación:

```
[ ]  def clean_descriptions(descriptions):
         # preparar tabla de traducción para eliminar la puntuación

         table = str.maketrans('', '', string.punctuation)
         for key, desc_list in descriptions.items():
             for i in range(len(desc_list)):
                 desc = desc_list[i]
                 # tokenize
                 desc = desc.split()
                 # convertir a minúsculas
                 desc = [word.lower() for word in desc]
                 # remove punctuation from each token
                 desc = [w.translate(table) for w in desc]
                 # remove hanging 's' and 'a'
                 desc = [word for word in desc if len(word)>1]
                 # remove tokens with numbers in them
                 desc = [word for word in desc if word.isalpha()]
                 # store as string
                 desc_list[i] =  ' '.join(desc)
     clean_descriptions(descriptions)
```

```
[ ]  def to_vocabulary(descriptions):
         # build a list of all description strings
         all_desc = set()
         for key in descriptions.keys():
             [all_desc.update(d.split()) for d in descriptions[key]]
         return all_desc
     vocabulary = to_vocabulary(descriptions)
     print('Original Vocabulary Size: %d' % len(vocabulary))

     Original Vocabulary Size: 29
```

```
[ ]  def save_descriptions(descriptions, filename):
         lines = list()
         for key, desc_list in descriptions.items():
             for desc in desc_list:
                 lines.append(key + ' ' + desc)
         data = '\n'.join(lines)
         file = open(filename, 'w')
         file.write(data)
         file.close()

     save_descriptions(descriptions, '/content/drive/My Drive/new_model/DESCRIPCIONES_03.txt')
```

Ilustración 10. *Depuración de las descripciones de las imágenes.*

Con todas estas instrucciones, permitió definir el tamaño del vocabulario en primera instancia, depurar el mismo, y guardar el archivo depurado como “DESCRIPCIONES_03.txt”. Este proceso fue llevado a cabo debido a que existían problemas al momento de leer directamente el archivo txt, sin un pre-procesamiento previo, puesto que el archivo txt original no era cargado correctamente, al momento de importarlo al cuaderno de programación.

Posteriormente, se realizó la importación de las descripciones de cada imagen con el procesamiento previo indicado anteriormente, al cuaderno de programación de Google Colab, como se muestra a continuación:

```
[ ]  caption_path =  '/content/drive/My Drive/new_model/DESCRIPCIONES_03.txt'
```

```
[ ]  captions = open(caption_path, 'rb').read().decode('utf-8').split('\n')
```

```
[ ]  captions
```

```
'DR_024_EEUU.jpg deslizamiento rotacional inactivo direccion sw area pequeña perimetro pequeño',
'DR_025_ARG.jpg deslizamiento rotacional activo direccion sw area mediana perimetro mediano',
'DR_026_ARG.jpg deslizamiento rotacional activo direccion este area mediana perimetro mediano',
'DR_027_ARG.jpg deslizamiento rotacional activo direccion sw area pequeña perimetro pequeño',
'DR_028_ARG.jpg deslizamiento rotacional activo direccion nw area mediana perimetro mediano',
'DR_029_ARG.jpg deslizamiento rotacional activo direccion oeste area pequeña perimetro pequeño',
'DR_030_ARG.jpg deslizamiento rotacional activo direccion norte area pequeña perimetro pequeño',
'DR_031_ARG.jpg deslizamiento rotacional activo direccion se area pequeña perimetro pequeño',
'DR_032_ECU.jpg deslizamiento rotacional activo direccion se area pequeña perimetro pequeño',
'DR_033_ESP.jpg deslizamiento rotacional activo direccion este area pequeña perimetro pequeño',
'DR_034_ARG.jpg deslizamiento rotacional activo direccion sw area pequeña perimetro pequeño',
'DR_035_ARG.jpg deslizamiento rotacional activo direccion sw area pequeña perimetro pequeño',
'DR_036_ARG.jpg deslizamiento rotacional activo direccion sw area pequeña perimetro pequeño',
'DR_037_ARG.jpg deslizamiento rotacional activo direccion oeste area pequeña perimetro mediano',
'DR_038_ARG.jpg deslizamiento rotacional activo direccion nw area pequeña perimetro pequeño',
'DR_039_ARG.jpg deslizamiento rotacional activo direccion nw area pequeña perimetro mediano',
'DR_040_ARG.jpg deslizamiento rotacional activo direccion sw area pequeña perimetro mediano',
'DR_041_ARG.jpg deslizamiento rotacional activo direccion se area pequeña perimetro pequeño',
```


Ilustración 11. *Importación de descripciones al modelo Resnet50*

Para poder asociar el archivo txt importado con el id (identificación de la imagen) y caption (descripción de la imagen) de la imagen, se utilizó las siguientes líneas de código:

```
captions_dict = {}
for i in captions:
    try:
        img_name = i.split(" ")[0]
        caption = i.split(".jpg")[1]
        if img_name in images_features:
            if img_name not in captions_dict:
                captions_dict[img_name] = [caption]

            else:
                captions_dict[img_name].append(caption)

    except:
        pass
```

***Ilustración 12**. Reconocimiento de captions e image_name.*

En donde le decimos al programa que en “img_name” le tome como el id de la imagen y cuando encuentre un espacio le designe el valor de 0, y después de la extensión .jpg le designe un valor de 1 a caption, que vendría siendo la descripción de la misma, y así se forma el directorio de todos los captions de cada una de las imágenes, como se muestra a continuación:

```
[ ] captions_dict
```

```
{'CD_001_ECU.jpg': [' caida detritos activo direccion ne area pequeña perimetro pequeño'],
 'CD_002_ECU.jpg': [' caida detritos inactivo direccion sw area pequeña perimetro pequeño'],
 'CD_003_ECU.jpg': [' caida detritos activo direccion se area pequeña perimetro pequeño'],
 'CD_004_ECU.jpg': [' caida detritos activo direccion se area pequeña perimetro pequeño'],
 'CD_005_ECU.jpg': [' caida detritos inactivo direccion ne area pequeña perimetro pequeño'],
 'CD_006_ECU.jpg': [' caida detritos inactivo direccion nw area pequeña perimetro pequeño'],
 'CD_007_ECU.jpg': [' caida detritos activo direccion nw area pequeña perimetro pequeño'],
 'CD_008_ECU.jpg': [' caida detritos activo direccion nw area pequeña perimetro pequeño'],
 'CD_009_ECU.jpg': [' caida detritos activo direccion nw area pequeña perimetro pequeño'],
 'CD_010_ECU.jpg': [' caida detritos activo direccion este area pequeña perimetro pequeño'],
 'CD_011_ECU.jpg': [' caida detritos activo direccion nw area pequeña perimetro pequeño'],
 'CD_012_ECU.jpg': [' caida detritos inactivo direccion norte area pequeña perimetro pequeño'],
 'CD_013_ECU.jpg': [' caida detritos inactivo direccion ne area pequeña perimetro pequeño'],
 'CD_014_ECU.jpg': [' caida detritos activo direccion oeste area pequeña perimetro pequeño'],
 'CD_015_ECU.jpg': [' caida detritos activo direccion nw area pequeña perimetro pequeño'],
 'CD_016_ECU.jpg': [' caida detritos activo direccion sw area pequeña perimetro pequeño'],
 'CD_017_ECU.jpg': [' caida detritos activo direccion se area pequeña perimetro pequeño'],
 'CD_018_ECU.jpg': [' caida detritos inactivo direccion oeste area pequeña perimetro pequeño'],
 'CD_019_ECU.jpg': [' caida detritos inactivo direccion oeste area pequeña perimetro pequeño'],
 'CD_020_ECU.jpg': [' caida detritos inactivo direccion oeste area pequeña perimetro pequeño'],
 'CD_021_ECU.jpg': [' caida detritos activo direccion nw area pequeña perimetro pequeño'],
 'CD_022_ECU.jpg': [' caida detritos activo direccion ne area pequeña perimetro pequeño'],
 'CD_024_ECU.jpg': [' caida detritos activo direccion se area pequeña perimetro pequeño'],
 'CD_025_ECU.jpg': [' caida detritos activo direccion se area pequeña perimetro pequeño'],
 'CD_026_ECU.jpg': [' caida detritos activo direccion este area pequeña perimetro pequeño'],
 'CD_027_ECU.jpg': [' caida detritos activo direccion este area pequeña perimetro pequeño'],
 'CD_028_ECU.jpg': [' caida detritos activo direccion norte area pequeña perimetro pequeño'],
```


***Ilustración 13**. Directorio "captions".*

Luego se procede a la creación del vocabulario, generando inicialmente un contador de palabras como se observa en el código:

```
[243] count_words = {}
      for k,vv in captions_dict.items():
          for v in vv:
              for word in v.split():
                  if word not in count_words:

                      count_words[word] = 0

                  else:
                      count_words[word] += 1
```

```
[244] len(count_words)

      27
```

Ilustración 14. *Contador de palabras*

De esta manera se cuanta las palabras se utilizan en cada descripción, mediante un for anidado, el cual controla columnas y filas respectivamente, también se utiliza un codificador de palabras, asignando números a cada palabra encontrada en la descripción, posteriormente necesitamos efectuar una matriz con todos los números asignados anteriormente, es decir transformamos nuestra descripción de texto en una secuencia de números, formando vectores de palabras, los cuales son aptos para que la red neuronal recurrente lo procese, como se muestra a continuación:

```
[ ] THRESH = -1
    count = 1
    new_dict = {}
    for k,v in count_words.items():
        if count_words[k] > THRESH:
            new_dict[k] = count
            count += 1
```

```
[ ] len(new_dict)

    31
```

```
[ ] new_dict

    {'activo': 4,
     'area': 7,
     'caida': 26,
     'de': 24,
     'depequeño': 31,
     'deslizamiento': 2,
     'detritos': 27,
     'direccion': 5,
     'endofseq': 11,
     'este': 17,
     'extenso': 30,
     'flujos': 28,
     'grande': 8,
```

```
[ ]  captions_dict
```

```
{'CD_001_ECU.jpg': [[1, 26, 27, 4, 5, 18, 7, 14, 9, 15, 11]],
 'CD_002_ECU.jpg': [[1, 26, 27, 21, 5, 12, 7, 14, 9, 15, 11]],
 'CD_003_ECU.jpg': [[1, 26, 27, 4, 5, 6, 7, 14, 9, 15, 11]],
 'CD_004_ECU.jpg': [[1, 26, 27, 4, 5, 6, 7, 14, 9, 15, 11]],
 'CD_005_ECU.jpg': [[1, 26, 27, 21, 5, 18, 7, 14, 9, 15, 11]],
 'CD_006_ECU.jpg': [[1, 26, 27, 21, 5, 20, 7, 14, 9, 15, 11]],
 'CD_007_ECU.jpg': [[1, 26, 27, 4, 5, 20, 7, 14, 9, 15, 11]],
 'CD_008_ECU.jpg': [[1, 26, 27, 4, 5, 20, 7, 14, 9, 15, 11]],
 'CD_009_ECU.jpg': [[1, 26, 27, 4, 5, 20, 7, 14, 9, 15, 11]],
 'CD_010_ECU.jpg': [[1, 26, 27, 4, 5, 17, 7, 14, 9, 15, 11]],
 'CD_011_ECU.jpg': [[1, 26, 27, 4, 5, 20, 7, 14, 9, 15, 11]],
 'CD_012_ECU.jpg': [[1, 26, 27, 21, 5, 22, 7, 14, 9, 15, 11]],
 'CD_013_ECU.jpg': [[1, 26, 27, 21, 5, 18, 7, 14, 9, 15, 11]],
 'CD_014_ECU.jpg': [[1, 26, 27, 4, 5, 19, 7, 14, 9, 15, 11]],
 'CD_015_ECU.jpg': [[1, 26, 27, 4, 5, 20, 7, 14, 9, 15, 11]],
 'CD_016_ECU.jpg': [[1, 26, 27, 4, 5, 12, 7, 14, 9, 15, 11]],
 'CD_017_ECU.jpg': [[1, 26, 27, 4, 5, 6, 7, 14, 9, 15, 11]],
 'CD_018_ECU.jpg': [[1, 26, 27, 21, 5, 19, 7, 14, 9, 15, 11]],
 'CD_019_ECU.jpg': [[1, 26, 27, 21, 5, 19, 7, 14, 9, 15, 11]],
 'CD_020_ECU.jpg': [[1, 26, 27, 21, 5, 19, 7, 14, 9, 15, 11]],
 'CD_021_ECU.jpg': [[1, 26, 27, 4, 5, 20, 7, 14, 9, 15, 11]],
 'CD_022_ECU.jpg': [[1, 26, 27, 4, 5, 18, 7, 14, 9, 15, 11]],
 'CD_024_ECU.jpg': [[1, 26, 27, 4, 5, 6, 7, 14, 9, 15, 11]],
 'CD_025_ECU.jpg': [[1, 26, 27, 4, 5, 6, 7, 14, 9, 15, 11]],
 'CD_026_ECU.jpg': [[1, 26, 27, 4, 5, 17, 7, 14, 9, 15, 11]],
```

Ilustración 15. *Vector de palabras*

Una vez realizado este proceso, pasamos a la construcción de la función generador, el cual va a servir de entrada para el modelo recurrente, se inicia obteniendo la longitud máxima de palabras en las descripciones de los deslizamientos, esto se lo obtiene evaluando cada descripción proporcionada al programa y obtenemos la variable Max_Len, que posteriormente nos va a servir para la generación del vocabulario, como se muestra a continuación:

```
351] MAX_LEN = 0
     for k, vv in captions_dict.items():
         for v in vv:
             if len(v) > MAX_LEN:
                 MAX_LEN = len(v)
                 print(v)
```

Ilustración 16*. Variable Max_Len.*

```
[ ]  Batch_size = 50
     VOCAB_SIZE = len(new_dict)

     def generator(photo, caption):
         n_samples = 0

         X = []
         y_in = []
         y_out = []

         for k, vv in caption.items():
             for v in vv:
                 for i in range(1, len(v)):
                     X.append(photo[k])

                     in_seq= [v[:i]]
                     out_seq = v[i]

                     in_seq = pad_sequences(in_seq, maxlen=MAX_LEN, padding='post', truncating='post')[0]
                     out_seq = to_categorical([out_seq], num_classes=VOCAB_SIZE)[0]

                     y_in.append(in_seq)
                     y_out.append(out_seq)

         return X, y_in, y_out
```

```
[ ]  VOCAB_SIZE

     32
```

Ilustración 17. *Definición del vocabulario.*

Después, se forma el embedding que es una representación del texto aprendido que genera palabras que van a estar representadas de manera vectorial.

c. Creación del Modelo

Se elaboró el modelo de Deep Learning a partir de uno preexistente, construido para la clasificación de imágenes de personas, animales u otros objetos en diferentes lugares o realizando alguna actividad. Este modelo ha sido adaptado a nuestro proyecto aportando con el data set tanto de imágenes de deslizamientos como de descripciones.

La tarea de subtítulos de imágenes se puede dividir lógicamente en dos módulos: uno es un modelo basado en imágenes, que extrae las características y matices de nuestra imagen, y el otro es un modelo basado en el lenguaje, que traduce las características y objetos dados por nuestra imagen. modelo basado en una oración natural. Para nuestro modelo basado en imágenes (encoder), generalmente confiamos en un modelo de red neuronal convolucional. Y para nuestro modelo basado en lenguaje (decoder), confiamos en una red neuronal recurrente. La imagen a continuación resume el enfoque dado anteriormente.

i. ResNet50

El modelo utilizado mediante trasnferlearning es Resnet50 el cual posee una arquitectura que busca que el incremento de capas se realice de manera distinta a la tradicional, por lo

que agrega una conexión residual con una capa identidad, la cual pasa a la siguiente capa de manera directa mejorando considerablemente el entrenamiento del modelo.

El procesamiento se encuentra divido en 5 convoluciones las cuales se detalla en el siguiente cuadro:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Ilustración 18. *Estructuras de modelos ResNet. Recuperado de: https://programmerclick.com/article/4154243862/*

Las dos capas convolucionales de 3x3 se reemplazan con 1x1 + 3x3 + 1x1, como se muestra a continuación. La capa convolucional media de 3x3 en la nueva estructura primero reduce el cálculo bajo una capa convolucional 1x1 reducida en una dimensión y luego la restaura bajo otra capa convolucional 1x1, que mantiene la precisión y reduce la cantidad de cálculos (Bach_normalization). El bloque base de esta red se llama residual block y está compuesto, cuando la red tiene 50 o más capas, por tres convoluciones secuenciales, una 1×1, una 3×3 y una 1×1, y una conexión que une la entrada de la primera convolución a la salida de la tercera convolución

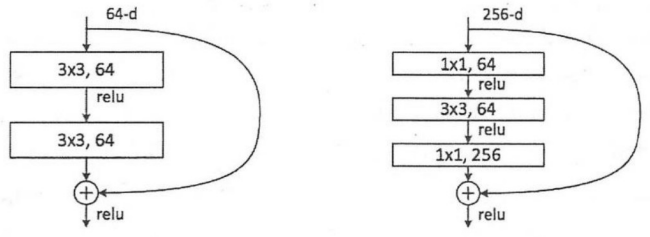


Ilustración 19. *Operación del modelo ResNet. Recuperado de: https://programmerclick.com/article/4154243862/*

La operación del modelo ResNet50 inicia con una imagen de dimensiones (224,224,3), aplica un padding (ZeroPadding2D) rellenando los bordes con pixeles de valor 0, modificando el tamaño original de la imagen (230,230,3), posterior se procede con la primera convolución aplicando un filtro de 64*64, con una dimensión de kernel de 7*7*3 obteniendo un mapa de

características con dimensiones (112,112,64) y con un numero de parámetros de 9472, a continuación se aplica una normalización por lotes (Batch_Normalization) manteniendo las dimensiones, pero reduciendo el número de parámetros a 256, con el fin de mejorar los procesos de aprendizajes. La función ReLu se encarga de encargada de transformar los valores introducidos, anulando los valores negativos y dejando los positivos tal y como entran. La función 'Maxpooling' encargada de disminuir la altura y longitud de la imagen, pero manteniendo su profundidad, modificando sus dimensiones a (56,56,64). Al final de realizar las convoluciones por bloques de cada capa, se procede a calcular la salida promedio de cada mapa de características, esto reduce los datos de manera significativa y prepara el modelo para la capa de clasificación final. Cabe recalcar que para este proyecto se retiró las últimas dos capas de modelo, dado que el objetivo final no es un clasificador de imágenes

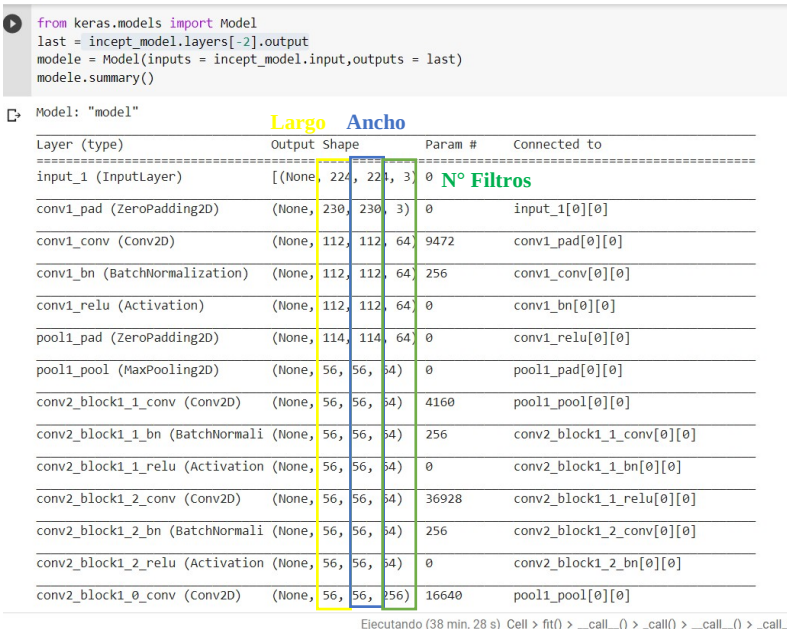


Ilustración 20. *Sumario del modelo ResNet*

Tabla 4. *Cálculo de parámetros modelo ResNet*

Alto	Ancho	Profundidad	#Filtros	Bias	Parámetros#
7	7	3	64	64	9472
1	1	64	64	64	4160
3	3	64	64	64	36928
1	1	64	256	256	16640

```
[ ] Model: "model"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 224, 224, 3) 0
_________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 230, 230, 3)  0           input_1[0][0]
_________________________________________________________________________________________________
conv1_conv (Conv2D)          (None, 112, 112, 64) 9472        conv1_pad[0][0]
_________________________________________________________________________________________________
conv1_bn (BatchNormalization)(None, 112, 112, 64) 256         conv1_conv[0][0]
_________________________________________________________________________________________________
conv1_relu (Activation)      (None, 112, 112, 64) 0           conv1_bn[0][0]
_________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)    (None, 114, 114, 64) 0           conv1_relu[0][0]
_________________________________________________________________________________________________
pool1_pool (MaxPooling2D)    (None, 56, 56, 64)   0           pool1_pad[0][0]
_________________________________________________________________________________________________
conv2_block1_1_conv (Conv2D) (None, 56, 56, 64)   4160        pool1_pool[0][0]
_________________________________________________________________________________________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_1_conv[0][0]
_________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0           conv2_block1_1_bn[0][0]
_________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D) (None, 56, 56, 64)   36928       conv2_block1_1_relu[0][0]
_________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256         conv2_block1_2_conv[0][0]
_________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0           conv2_block1_2_bn[0][0]
```

Ilustración 21. *Sumario final modelo Res_Net*

ii. LSTM

La implementación actual asume Resnet50 como un extractor de características que consta de bloques de construcción de convolución, integrados con conexiones de acceso directo. Recibe una muestra de imagen recortada aleatoriamente de 224x224 con transformadores aplicados y extrae 2048 mapas de características con un tamaño de 7x7 cada uno. El embedding, permite disminuir las dimensiones de los datos de entrada, es decir permite comprimir o compactar los datos y ordenarlos de manera inteligente, de esta manera tenemos un vector de entrada de tamaño (None, 2048), el cual se obtuvo, a través del modelo convolucional, el mismo que ingresara a una capa de 128 neuronas, el cual nos permite obtener un vector de tamaños (None, 128), lo mismo se realiza para las capas de LSTM, correspondiente al texto, permitiéndonos así, formar la concatenación, como se mostrará a continuación:

```
embedding_size = 128
max_len = MAX_LEN
vocab_size = len(new_dict)

image_model = Sequential()

image_model.add(Dense(embedding_size, input_shape=(2048,), activation='relu'))
image_model.add(RepeatVector(max_len))

image_model.summary()

language_model = Sequential()

language_model.add(Embedding(input_dim=vocab_size, output_dim=embedding_size, input_length=max_len))
language_model.add(LSTM(256, return_sequences=True))
language_model.add(TimeDistributed(Dense(embedding_size)))

language_model.summary()

conca = Concatenate()([image_model.output, language_model.output])
x = LSTM(128, return_sequences=True)(conca)
x = LSTM(512, return_sequences=False)(x)
x = Dense(vocab_size)(x)
out = Activation('softmax')(x)
model = Model(inputs=[image_model.input, language_model.input], outputs = out)

model.compile(loss='categorical_crossentropy', optimizer='RMSprop', metrics=['accuracy'])
model.summary()
```

Ilustración 22. *Embedding.*

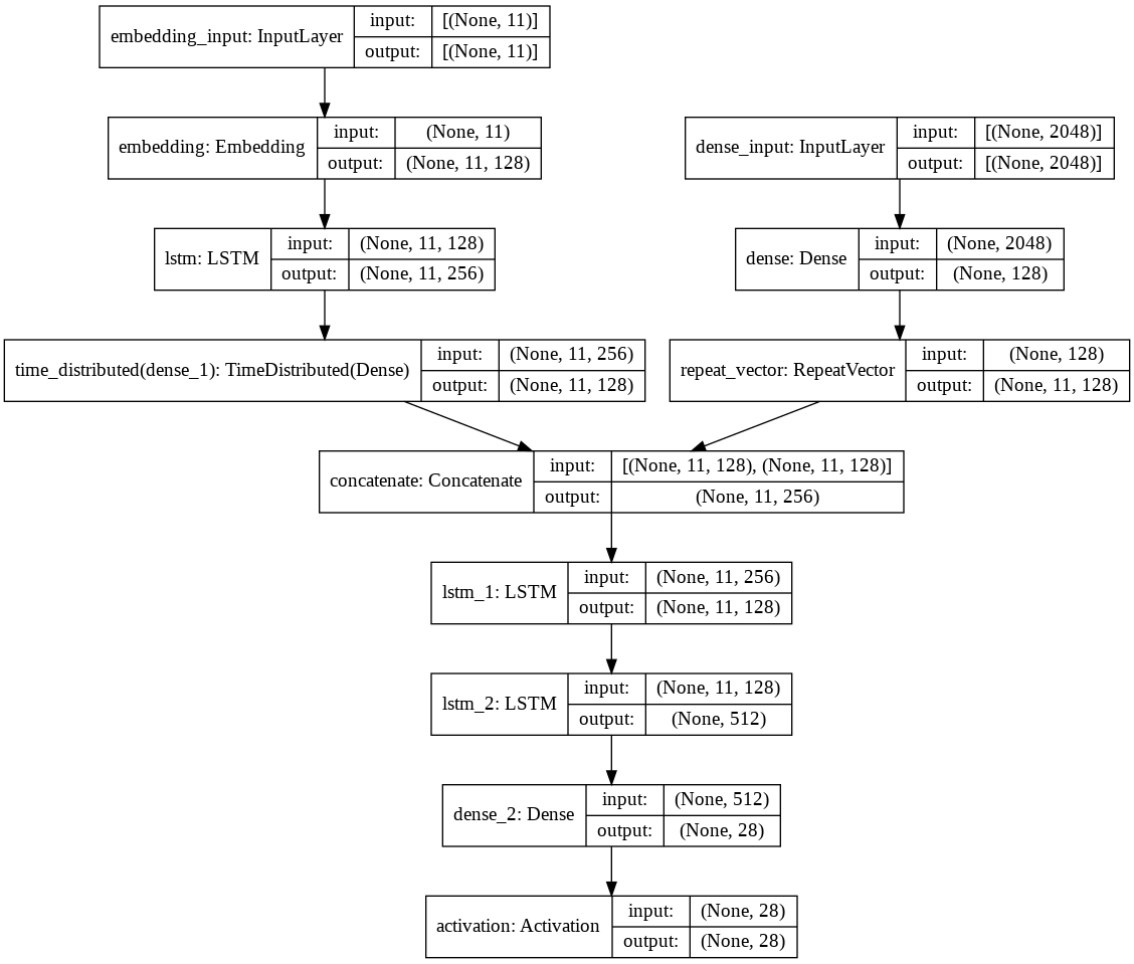


Ilustración 23. *Diagrama del modelo completo Resnet+LSTM*

Se debe tener en cuenta que obtenemos los vectores de características del último bloque de convolución sin aplicar la capa completamente conectada. Se debe concatenar los datos del vector de características (2048) junto con los datos del embedding, esto ingresa a nuestro conjunto de capas LSTM

```
Model: "model_1"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
embedding_input (InputLayer)    [(None, 11)]         0
_________________________________________________________________________________________________
dense_input (InputLayer)        [(None, 2048)]       0
_________________________________________________________________________________________________
embedding (Embedding)           (None, 11, 128)      3584        embedding_input[0][0]
_________________________________________________________________________________________________
dense (Dense)                   (None, 128)          262272      dense_input[0][0]
_________________________________________________________________________________________________
lstm (LSTM)                     (None, 11, 256)      394240      embedding[0][0]
_________________________________________________________________________________________________
repeat_vector (RepeatVector)    (None, 11, 128)      0           dense[0][0]
_________________________________________________________________________________________________
time_distributed (TimeDistribut (None, 11, 128)      32896       lstm[0][0]
_________________________________________________________________________________________________
concatenate (Concatenate)       (None, 11, 256)      0           repeat_vector[0][0]
                                                                 time_distributed[0][0]
_________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 11, 128)      197120      concatenate[0][0]
_________________________________________________________________________________________________
lstm_2 (LSTM)                   (None, 512)          1312768     lstm_1[0][0]
_________________________________________________________________________________________________
dense_2 (Dense)                 (None, 28)           14364       lstm_2[0][0]
_________________________________________________________________________________________________
activation (Activation)         (None, 28)           0           dense_2[0][0]
=================================================================================================
Total params: 2,217,244
Trainable params: 2,217,244
Non-trainable params: 0
```

Ilustración 24. *Sumario del modelo LSTM*

iii. Compilación y ajuste

La compilación del modelo se llevó a cabo con los siguientes hiperparámetros:

Tabla 5. *Tabla de hiperparámetros para la compilación del modelo*

HIPERPARÁMETROS	
embedding_size	128
max_len	11
batch_size	32
epochs	10-20-50
vocab_size	27

Como optimizador se utilizó RMSprop, el cual tiene la función de mantener un promedio móvil (descontado) del cuadrado de gradientes y dividir el gradiente por la raíz de este promedio, como función de pérdida se utilizó categorical_crossentropy, y para las métricas recogidas se utilizó el acurracy.

Como se observa la tabla anterior se realizaron 3 entrenamientos, en donde se varió el número de épocas, para de esta manera determinar el modelo con la mejor aproximación.

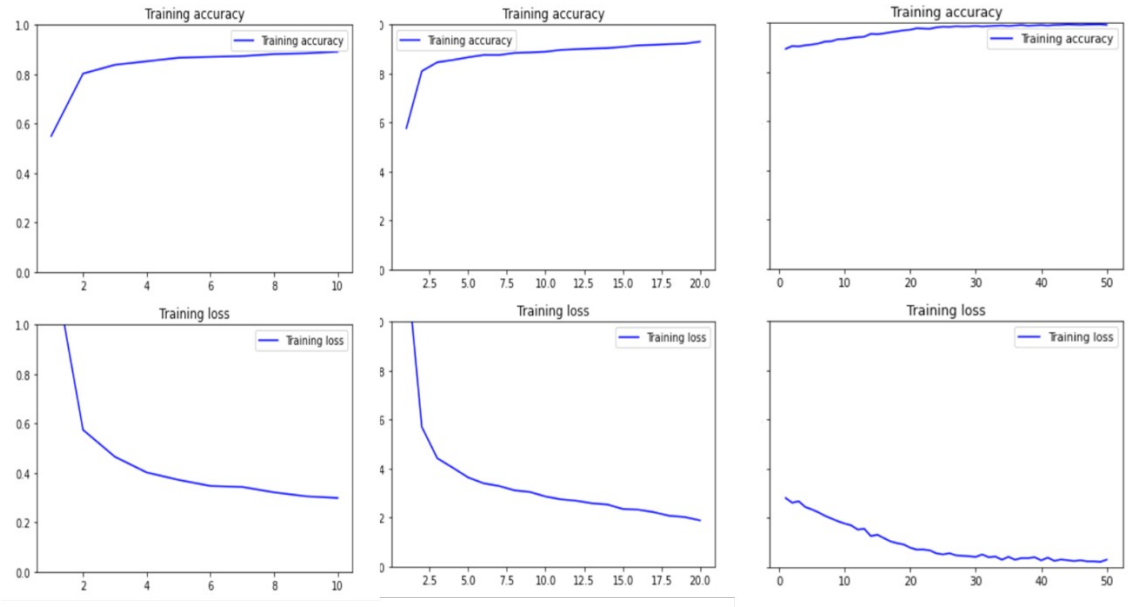


Ilustración 25. *Graficas de precisión y pérdida para los modelos con 10,20 y 50 épocas*

En base a las gráficas obtenidas se puede apreciar que el modelo con mejores valores de precisión, es el de 50 épocas. Se aprecia que desde el inicio mantiene unos valores altos de precisión manteniéndose constante hasta el final. De igual manera los valores de pérdida del entrenamiento se mantienen bajos de manera constante.

Tabla 6. *Tiempo de compilación para cada época*

ÉPOCAS	TIEMPO(s)
10	447,07
20	948,55
50	2782.495

d. Evaluación del modelo

Para la prueba del modelo se utiliza las 200 imágenes restantes asignadas al dataset test, en donde de igual manera se las trato para que sean perceptibles por el modelo. En el código obtenido de Kaggle se plantea realizar la evaluación mediante el código:

```
for i in range(5):

    path = images_test[i]
    pred_img = cv2.imread(path)
    pred_img = cv2.cvtColor(pred_img, cv2.COLOR_BGR2RGB)
    pred_img = cv2.resize(pred_img, (299,299))
    pred_img = np.reshape(pred_img, (1,299,299,3))
    pred_feature = modele.predict(pred_img).reshape(1,2048)
    pred_img = cv2.imread(path)
    pred_img = cv2.cvtColor(pred_img, cv2.COLOR_BGR2RGB)
    #y_pred = {}
```

Ilustración 26. *Código para evaluar las imágenes pertenecientes al dataset de test*

Donde se obtuvieron los siguientes resultados:

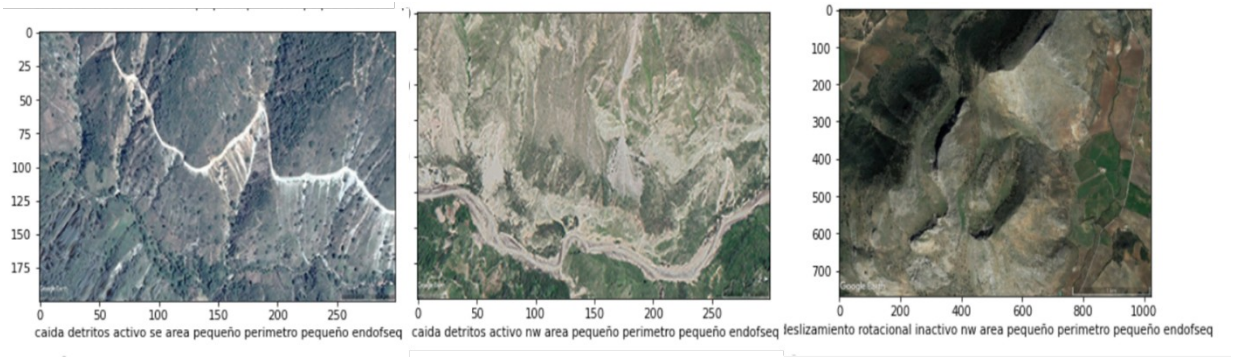


Ilustración 27. *Resultados de evaluación del modelo*

Se aprecia que posee una gran precisión con respecto a la descripción de cada deslizamiento.

e. Matriz de Confusión

La matriz de confusión es una herramienta muy útil para valorar que tan bueno es un modelo clasificación basado en aprendizaje automático. En particular, sirve para mostrar de forma explícita cuándo una clase es confundida con otra, lo cual nos, permite trabajar de forma separada con distintos tipos de error.

El objetivo era crear este modelo es que predijera el tipo de deslizamiento, dirección, tipo de área y perímetro, estableciéndolo en una “categorización”, basándose en el valor de esas 9 variables a partir de los datos de entrenamiento.

```
[ ] from sklearn.metrics import confusion_matrix, f1_score, roc_curve, precision_score, recall_score, accuracy_score, roc_auc_score
    from sklearn import metrics
    from mlxtend.plotting import plot_confusion_matrix
    from keras.models import load_model
    import numpy as np
    import matplotlib.pyplot as plt
    %matplotlib inline


    names = ['CR','CR_ETQ','DR','DR_ETQ','Flujos',
             'Flujos_ETQ','RS','RS_ETQ']


    test_data_dir = '/content/drive/MyDrive/DATA_SET/TEST'

    test_datagen = ImageDataGenerator()

    test_generator = test_datagen.flow_from_directory(
        test_data_dir,
        target_size=(width_shape, height_shape),
        batch_size = batch_size,
        class_mode='categorical',
        shuffle=False)

    custom_Model= load_model("model_VGG16.h5")

    predictions = custom_Model.predict_generator(generator=test_generator)

    y_pred = np.argmax(predictions, axis=1)
    y_real = test_generator.classes


    matc=confusion_matrix(y_real, y_pred)

    plot_confusion_matrix(conf_mat=matc, figsize=(9,9), show_normed=False)
```

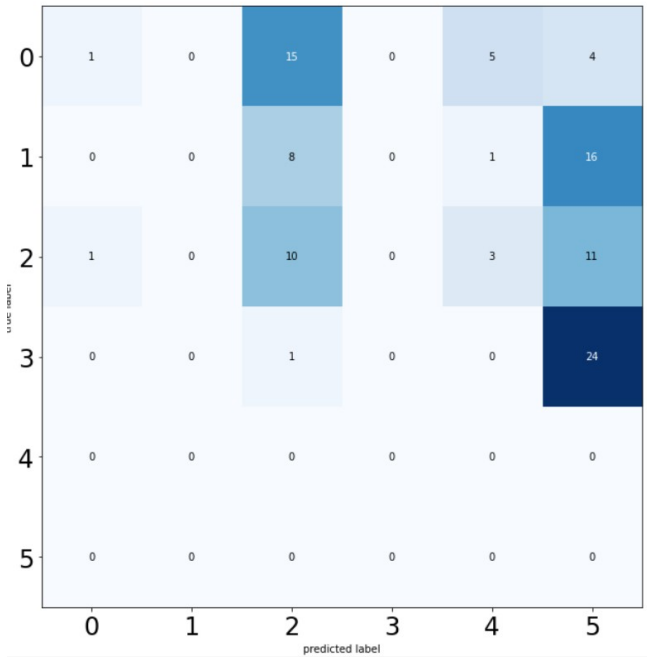


Ilustración 38. *1ra Matriz*

```
names = ['CR','CR_ETQ','DR','DR_ETQ','Flujos',
         'Flujos_ETQ','RS','RS_ETQ']


test_data_dir = '/content/drive/MyDrive/DATA_SET/TEST'

test_datagen = ImageDataGenerator()

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(width_shape, height_shape),
    batch_size = batch_size,
    class_mode='categorical',
    shuffle=False)

custom_Model1= load_model("/content/drive/MyDrive/DATA_SET/new_model/resnet/model1.h5")

predictions = custom_Model1.predict_generator(generator=test_generator)

y_pred1 = np.argmax(predictions, axis=1)
y_real1 = test_generator.classes


matc=confusion_matrix(y_real1, y_pred1)

plot_confusion_matrix(conf_mat=matc, figsize=(9,9), show_normed=False)
plt.tight_layout()

print(metrics.classification_report(y_real,y_pred, digits = 4))
```

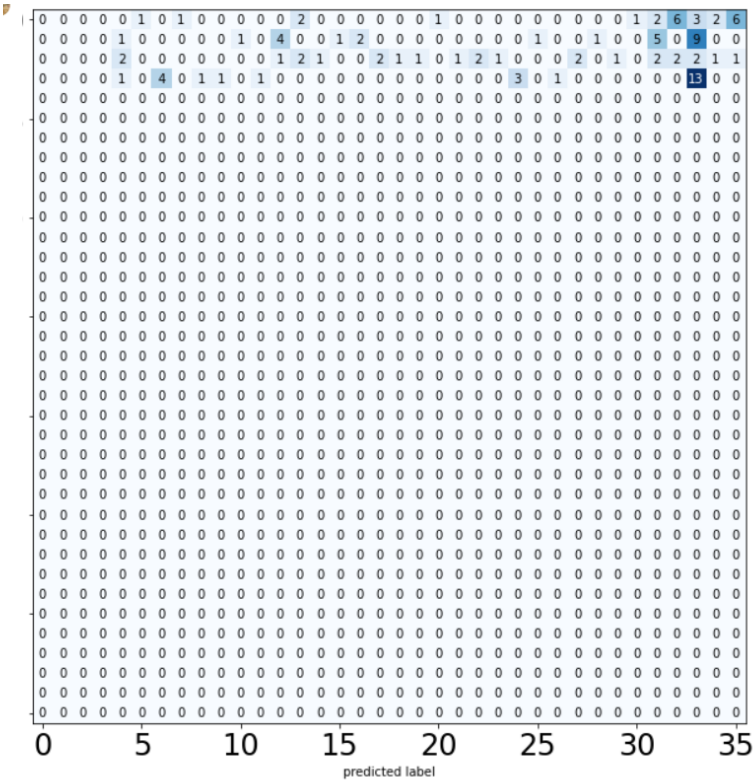


Ilustración 49. *Matriz de Confusión*

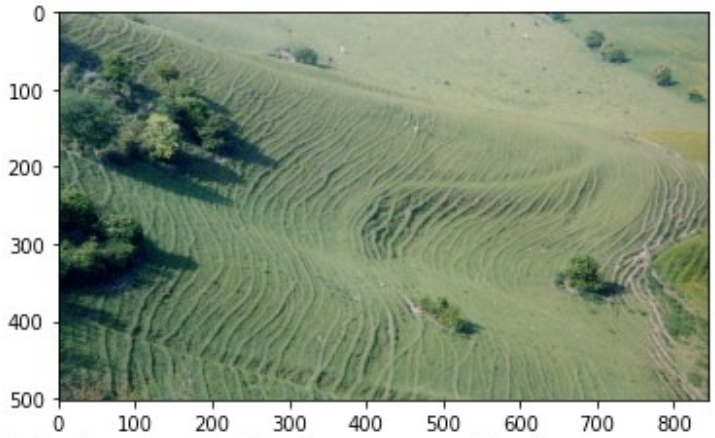
f. Predicción

Para la predicción del modelo se optó por utilizar imágenes obtenidas desde Google Earth Pro o descargadas directamente desde la Web, que el programa no haya visto, es decir q no se encontraban dentro de la base de datos de train o test, utilizadas para la compilación.

Elegir archivos prueba10.png

- **prueba10.png**(image/png) - 870548 bytes, last modified: 26/9/2021 - 100% done

```
Saving prueba10.png to prueba10.png
User uploaded file "prueba10.png" with length 870548 bytes
```

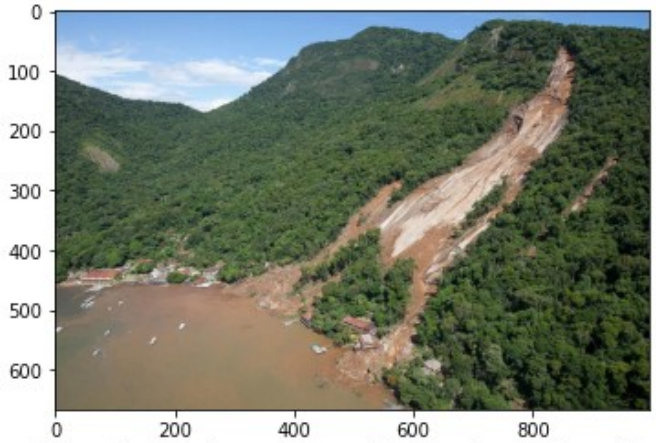


reptacion de suelos activo direccion sw area pequeña perimetro pequeño endofseq

Elegir archivos prueba7.jpg

- **prueba7.jpg**(image/jpeg) - 173347 bytes, last modified: 26/9/2021 - 100% done

```
Saving prueba7.jpg to prueba7.jpg
User uploaded file "prueba7.jpg" with length 173347 bytes
```



flujos activo direccion ne area pequeña perimetro pequeño endofseq

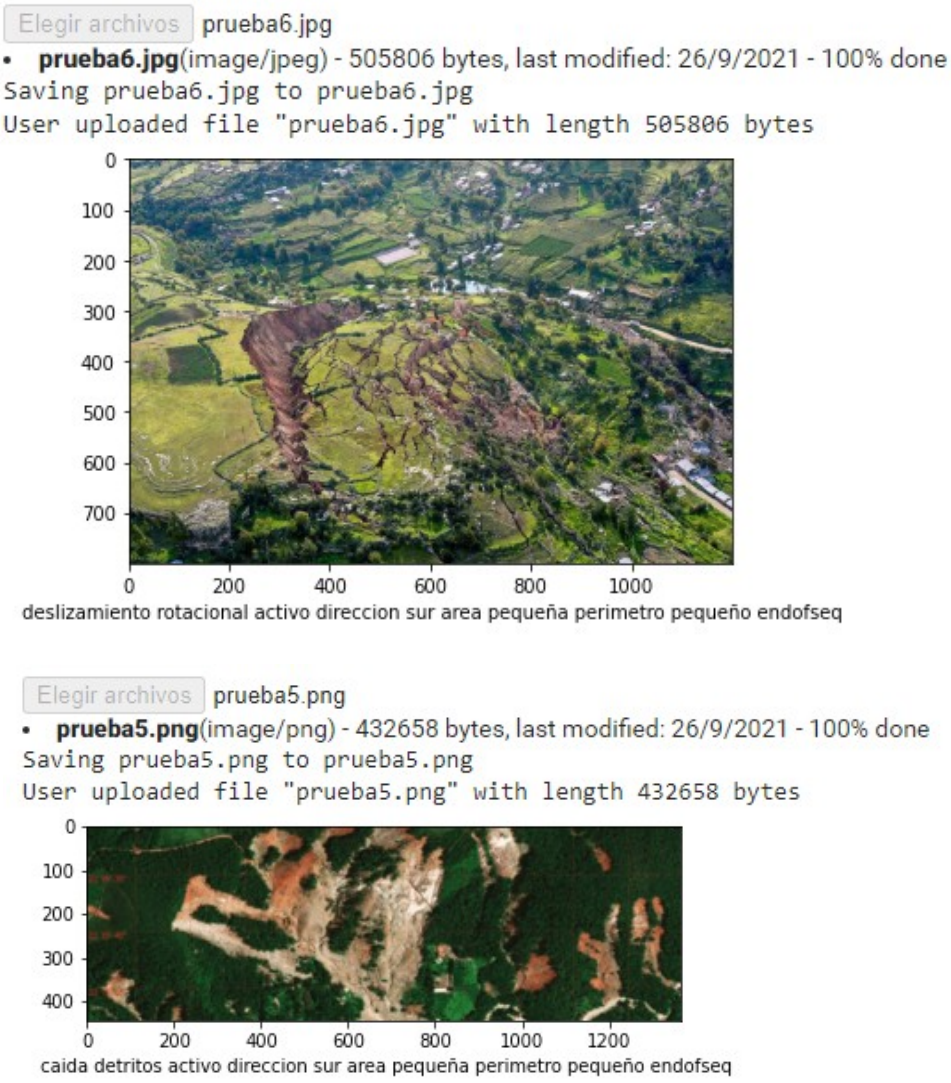


Ilustración 30. *Resultados de predicción del modelo*

Las predicciones en las 4 imágenes fueron de gran precisión validando de esta manera nuestro modelo propuesto.

g. Conclusiones y Recomendaciones

i. Conclusiones

- ✓ El dataset recopilado inicialmente se lo corrigió varias veces, debido a que el modelo propuesto acepta un cierto tipo de datos, por tal motivo se aplicó un preprocesamiento al igual que las imágenes en conjunto con una depuración de datos erróneos y una clasificación estadística para el área y perímetro de los deslizamientos, evitando de esta manera trabajar con caracteres numéricos.
- ✓ En cuanto a la implementación del modelo no necesariamente un mayor número de capas o aumentar épocas al momento de compilar nos garantiza que el modelo posea una precisión mayor, puesto que se puede caer en un sobre ajuste, afectando a los resultados finales.

- ✓ Se diseñó un modelo Deep Learning a partir de uno preexistente, mediante transferlearning y que ha sido adaptado a nuestro proyecto con base de datos propia (80% train, 20% test) en base a los objetivos planteados del proyecto, de tal manera que los parámetros del modelo final fueron: un modelo convolucional ResNet de 152 capas el cual nos permitió extraer las características más importantes de cada imagen de entrenamiento (encoder), un vocabulario el cual se lo realizó con las descripciones de los deslizamientos (decoder). Para finalmente concatenarlos y poder realizar el entrenamiento por medio de una red recurrente LSTM con un modelo de 28 capas, con una compilación de 50 épocas.
- ✓ En base a la predicción realizada con imágenes ajenas al dataset se puede concluir que la implementación de redes neuronales para la caracterización de deslizamientos es de suma eficacia, dado que nos facilita la caracterización de los principales parámetros que se necesita saber para un inventario de deslizamientos ocupado en la construcción de mapas de susceptibilidad.

ii. Recomendaciones

- ✓ Es fundamental aumentar el dataset, dado que, con mayor número de datos, tanto de imágenes como de descripciones, perfeccionaría el entrenamiento del modelo, arrojando mejores valores al momento de realizar las predicciones.
- ✓ Se recomienda hacer una mejor adaptación del modelo propuesto en la plataforma de Kaggle, dado sus buenos resultados obtenidos durante la predicción, para que de esta manera se pueda optar por métodos de evaluación de la precisión como matriz de confusión o bleu score.
- ✓ Es recomendable realizar varias compilaciones modificando los hiperparámetros, para de esta manera seleccionar el modelo que mejores resultados arrojen tanto en perdida como en precisión.
- ✓ El aumento de parámetros de deslizamientos nos permitirá realizar un dataset más exhaustivo mejorando de esta manera la elaboración de inventarios de deslizamientos.

6. Bibliografía

An Buiv, A. & Jun lee, P. (2020). *Deep Learning for Landslide Recognition in Satellite Architecture*. Escuela de Ingeniería, Universidad Politécnica de Nanyang, Singapur.

Bello, I (2021) Revisiting ResNets: *Improved Training and Scaling Strategies*. Recuperado de https://arxiv.org/pdf/2103.07579.pdf .

Bhatt, J., Gangwar, A., Nijhawan, R. & Gangodkar, D. (2019). *A Research on Deep Learning Advance for Landslide Classification using Convolutional Neural Networks*. Dept. of CSE, GEU, Dehradun (Uttarakhand), India.

Bisong, E. (2019). *Google Colaboratory.* Inglaterra: Ekaba Bisong 2019.

Challenger-Pérez, I., Díaz-Ricardo, Y., & Becerra-García, R. A. (2014). *El lenguaje de programación Python/The programming language Python*. Cuba: Centro de Información y Gestión Tecnológica de Santiago de Cuba.

Gómez-Ros, A., Tabik, S., Luengo, J., Herrera, F., Shihavuddin, A. S. M., & Krawczyk, B. (2019). *Redes Neuronales Convolucionales para Una Clasificacion Precisa de Imagenes de Corales.* In XVIII Conferencia de la Asociacion Espanola para la Inteligencia Artificial (pp. 1171-1176)

H. Wang et. al. (2020). *Landslide identification using machine learning.* China University of Geosciences, Beijing.

Perez, D (2021) *Transfer learning en la clasificación binaria de imágenes térmicas Transfer Learning for Binary Classification of Thermal Images*. ∗Laboratorio de Sistemas Automáticos de Control, Universidad de Piura, Perú.

Prakash, N., Manconi, A. & Loew, S. (2021). *A new strategy to map landslides with a generalized convolutional neural network*.

Ramirez, A. O. (2010). *Python como primer lenguaje de programaci´on.* Mexico: Tecnologico de Monterrey, Campus Estado de Mexico.

Sarmient, J .(2020). *Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica*. Grupo de investigación SISTECBIO, Ingeniería Biomédica, Universidad Manuela Beltrán, Colombia.

Zhengjing, M., Gang, M., & Piccialli, F. (2020). *Machine learning for landslides prevention*. Beijing: Springer.

WEBGRAFÍA

https://www.kaggle.com/programminghut/imagecaptioning/notebook

https://keras.io/api/layers/reshaping_layers/zero_padding2d/

https://medium.com/analytics-vidhya/image-captioning-with-attention-part-1-e8a5f783f6d3

https://medium.com/analytics-vidhya/image-captioning-with-attention-part-2-f3616d5cf8d1

https://programmerclick.com/article/4154243862/