Denon, Nicole Alexandra

Desarrollo de un sistema de detección de malezas en cultivos de la región mediante el procesamiento de imágenes utilizando Inteligencia Artificial

2021

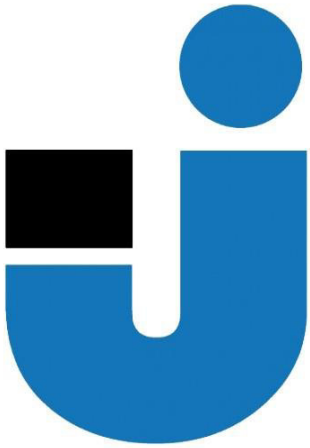
Instituto: Ingeniería y Agronomía

Carrera: Ingeniería en Informática

Universidad Nacional Arturo Jauretche

INSTITUTO DE INGENIERÍA Y AGRONOMÍA

INGENIERÍA EN INFORMÁTICA



TRABAJO FINAL DE LA PRÁCTICA PRFESIONAL SUPERVISADA

Desarrollo de un sistema de detección de malezas en cultivos de la región mediante el procesamiento de imágenes utilizando Inteligencia Artificial

Estudiante:

Nicole Alexandra Denon

Tutores:

Dr. Ing. Marcelo Cappelletti

Mg. Ing. Jorge Osio

Lic. Paula Mariana Bein

Buenos Aires, 2021

PRÁCTICA PROFESIONAL SUPERVISADA (PPS)

Desarrollo de un sistema de detección de malezas en cultivos de la región mediante el procesamiento de imágenes utilizando Inteligencia Artificial.

DATOS DEL ESTUDIANTE

Apellido y Nombres: DENON, Nicole Alexandra

DNI: 40094782

Nº de Legajo: 22400

Correo electrónico: nicole.a.denon@gmail.com

Cantidad de materias aprobadas al comienzo de la PPS: 41

PPS enmarcada en artículo (4 ó 7) de la Resolución (CS) 103/16. (En caso de ser artículo 7 aclarar en cuál de las dos alternativas posibles se encuadra)

DOCENTE SUPERVISOR

Apellido y Nombres: Dr. Ing. CAPPELLETTI, Marcelo

Correo electrónico: mcappelletti@unaj.edu.ar

DOCENTE TUTOR DEL TALLER DE APOYO A LA PRODUCCIÓN DE TEXTOS ACADÉMICOS DE LA UNAJ

Apellido y Nombres: Lic. BEIN, Paula Mariana

Correo electrónico: paula.bein@gmail.com

DATOS DE LA ORGANIZACIÓN DONDE SE REALIZA LA PPS

Firma Estudiante:	Firma Docente Supervisor:	Firma docente tutor TAPTA:	Firma tutor Organizacional:

Nombre o Razón Social: Universidad Nacional Arturo Jauretche

Dirección: Av. Calchaquí 6200, Florencio Varela, (1888) Buenos Aires, Argentina

Teléfono: +54 11 4275 6100

Sector: Programa Tecnologías de la Información y la Comunicación (TIC) en aplicaciones de Interés social, Instituto de Ingeniería y Agronomía

TUTOR DE LA ORGANIZACIONAL

Apellido y Nombres: Mg. Ing. OSIO, Jorge

Correo electrónico: josio@unaj.edu.ar

FIRMA DEL COORDINADOR DE LA CARRERA

Firma Estudiante:	Firma Docente Supervisor:	Firma docente tutor TAPTA:	Firma tutor Organizacional:

Contenido

1 Introducción 4
1.1 Objetivos 6
1.2 Evolución de la inteligencia artificial 6
2 Marco teórico 7
2.1 Machine Learning y Deep Learning 7
2.2 Redes Neuronales Convolucionales (CNN) 12
2.2.1 Capas Convolucionales 13
3 Caso de estudio: Malezas 16
3.1 La agricultura y la tecnología 16
3.2 La problemática de las malezas 17
4 Desarrollo 22
4.1 Herramientas y espacio de trabajo 22
4.2 Selección y edición de imágenes 25
4.3 El modelo de categorización 29
4.4 Resultados de la categorización 38
4.4.1 Resultados de las malezas: Rama negra 40
4.4.2 Resultados de las malezas: Yuyo Colorado 43

4.4.3 Resultados de las malezas: Roseta 47
4.4.4 Conclusiones de los resultados 48
4.5 Trabajo futuro 50
5 Conclusión 51
6 Bibliografía 53
7 Anexo: Código implementado 54

1 INTRODUCCIÓN

En los últimos años, han ocurrido distintos avances tecnológicos; Innovaciones como Big Data, Inteligencia Artificial, Cloud Computing, Machine Learning y Deep Learning han sido introducidas en diversas áreas tales como en finanzas, marketing, medicina, agricultura, académica, turismo, entre muchas otras. Esto se debe a que la transformación digital no es propia de un sector en particular, sino que toda la humanidad en su conjunto puede beneficiarse de ella de diferentes formas.

Este trabajo se enfocará principalmente en la aplicación de Machine Learning (Aprendizaje Automático) y Deep Learning (Aprendizaje Profundo), aunque en realidad, también se emplean otras tecnologías relacionadas con las anteriores mencionadas.

Machine Learning (ML) es un subcampo de la Inteligencia Artificial, cada vez más utilizado hoy en día por distintas instituciones (empresas, laboratorios, organizaciones gubernamentales, entre otras) dado que es aplicable a múltiples

campos con resultados exitosos. Se trata de un modelo de análisis de datos en el cual, a partir de datos de entrada y de salida, el sistema “aprende” automáticamente detectando patrones y generando reglas. Estas últimas pueden aplicarse a otros datos de entrada para obtener resultados finales originales. De esta forma, el sistema se “entrena” para poder predecir respuestas con un margen de éxito.

Por otro lado, el Deep Learning es un área del ML, que trata de simular el aprendizaje humano y está compuesto por distintas capas de procesamiento de datos. A esto se lo conoce como Redes Neuronales Artificiales profundas. Es posible encontrar fácilmente ejemplos de estas aplicaciones en la vida cotidiana, al acceder a una red social y visualizar las publicidades seleccionadas y que se le ofrecen al usuario, haciendo uso de técnicas de ML, específicamente en este caso, de una técnica llamada “segmentación del cliente”.

En este sentido, la presente Práctica Profesional Supervisada (PPS) tiene como finalidad el desarrollo de un sistema de detección de malezas que se encuentren con frecuencia en cultivos del territorio argentino -principalmente la región pampeana- mediante el procesamiento de imágenes utilizando técnicas de Machine Learning.

La PPS se desarrollará en el marco del Proyecto de Investigación de la Universidad Nacional Arturo Jauretche UNAJ INVESTIGA 2017 (Código del Proyecto 80020170200025UJ y Resolución Rectoral N° 148/18 de fecha 29/06/2018), cuyo título es “Tecnologías de la información y las comunicaciones mediante IoT para la solución de problemas en el medio socioproductivo” (Director: Jorge Osio).

1.1 OBJETIVOS

El objetivo general de este trabajo es generar conocimiento en el sector agronómico, con el fin de optimizar el rendimiento de los cultivos intensivos en la propiedad de la UNAJ. Se espera lograr esto cumpliendo distintos objetivos específicos, los cuales son:

- Investigar sobre la problemática de las malezas en los cultivos.
- Definir un conjunto de datos limpio, comúnmente llamado “dataset”, que contenga las suficientes imágenes para que pueda ser procesado por un modelo de ML. Los datos deberán ser significativos y relevantes.
- Estudiar las principales características de diferentes técnicas de Machine Learning.
- Contar con un sistema capaz de procesar imágenes e identificarlas con una tasa de éxito mayor al 70%.

1.2 EVOLUCIÓN DE LA INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial (IA) es la simulación de la inteligencia humana por medio de máquinas que permite realizar distintas tareas de forma rápida y eficiente. El investigador, ingeniero y uno de los autores de la librería “Keras”, François Chollet, la define como: “el esfuerzo por automatizar las tareas intelectuales que normalmente realizan los humanos”. Surgió formalmente en el año 1956, cuando se definió la por primera vez el término, aunque se presentan muchos trabajos sobre la misma anteriores a ese año. Es un área muy general y universal. La IA abarca

distintos subcampos alguno de los cuales serán investigados en este trabajo: el Machine Learning y Deep Learning.

2 MARCO TEÓRICO

2.1 MACHINE LEARNING Y DEEP LEARNING

Si bien es normal que se consideren al Machine Learning (ML) y al Deep Learning (DL) como sinónimos, esto es erróneo. El DL es una implementación específica del ML. El Machine Learning, aprendizaje de máquinas o aprendizaje automático, es una rama de la inteligencia artificial que automatiza la construcción de un modelo analítico. Es un método científico en donde la máquina “aprende” a extraer patrones que se encuentran en un conjunto de datos, de forma automática y sin la explícita programación de los humanos. Por lo tanto, la computadora aprende las reglas que la ayudarán a predecir comportamientos y, posteriormente, a tomar decisiones.

Para comprender la diferencia con otras implementaciones de inteligencia artificial, es necesario definir tres elementos básicos: los datos, las reglas que se aplicarán a los mismos y los resultados finales. En la aplicación de IA clásica, se programan las reglas que se van a aplicar a la información que se reciba de entrada, dando como resultado final una respuesta que fue procesada en base a las reglas aplicadas. El Machine Learning, por su parte, realiza un proceso diferente en donde el programa analiza los datos de entrada y las respuestas correctas a esos datos. De esta forma, el sistema va detectando en cada iteración los patrones de los datos, y así desarrolla

un modelo analítico que “aprendió” a relacionar los datos de entrada con las respuestas correctas.

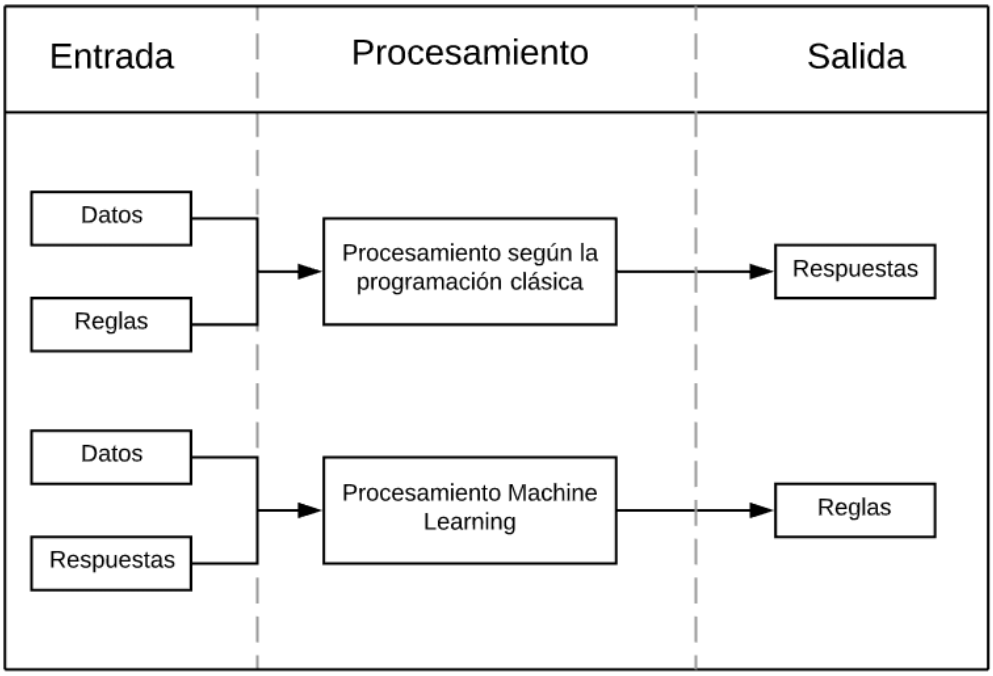


Figura 1. Gráfica sobre la diferencia de procesamiento entra la IA clásica y el ML adaptada del libro “Deep Learning with python” de François Chollet

Por otro lado, el Deep Learning surge como respuesta a la necesidad de resolver problemas complejos mediante el reconocimiento de patrones que el ML no puede detectar, como el procesamiento de imágenes, reconocimiento de voz, la categorización de videos, entre otros. Es una arquitectura jerárquica y entrelazada compuesta por redes neuronales profundas, es decir, que está compuesta por más de una capa oculta (de allí la palabra “deep” en su nombre) y está inspirado en el funcionamiento y arquitectura del cerebro humano. En la figura 2 se puede visualizar

la arquitectura básica de un modelo de Deep Learning, el cual puede estar compuesto por miles de neuronas y múltiples capas ocultas.

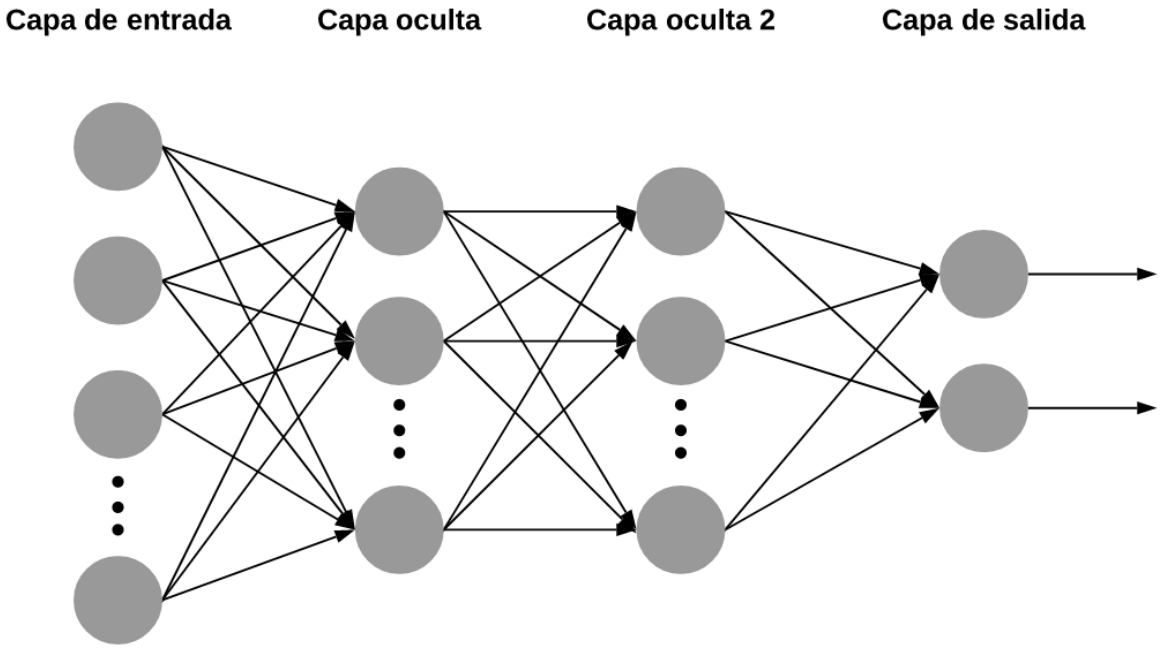


Figura 2. Ejemplificación de una arquitectura de Deep Learning en donde cada círculo gris representa a una neurona.

Por lo tanto, un modelo de Deep Learning está compuesto por:

- Neuronas: también llamadas *nodos*. Son unidades simplificadas de neuronas biológicas que se encargan de recibir información, procesarla a través de las conexiones, y finalmente, dar un valor de salida. En la figura 3 se puede observar cómo está compuesta una neurona en un modelo de ML. Sus componentes son:

- Datos de entrada (*X*): Son los valores de salida de la capa anterior.
- Pesos (*W*): Cada conexión entre dos neuronas tiene un peso. Estos se modificarán en cada iteración para adaptarse a un valor de salida que minimice el error del resultado real de la red.
- Bias: o sesgo es un parámetro que le agrega libertad al modelo.
- Función de activación: Es una función matemática que determina un valor de salida una vez que se hayan procesado cada una de las entradas. En la figura 3, la función de activación que se utiliza es la sigmoidal, la cual da valores de salida entre 0 y 1. Otras funciones de activación son la de la tangente hiperbólica (valores entre -1 y 1) y la ReLU, la cual será utilizada en este proyecto y su principal característica es que no está acotada en un rango finito de números.
- Datos de Salida: serán los datos de entradas de la siguiente capa.

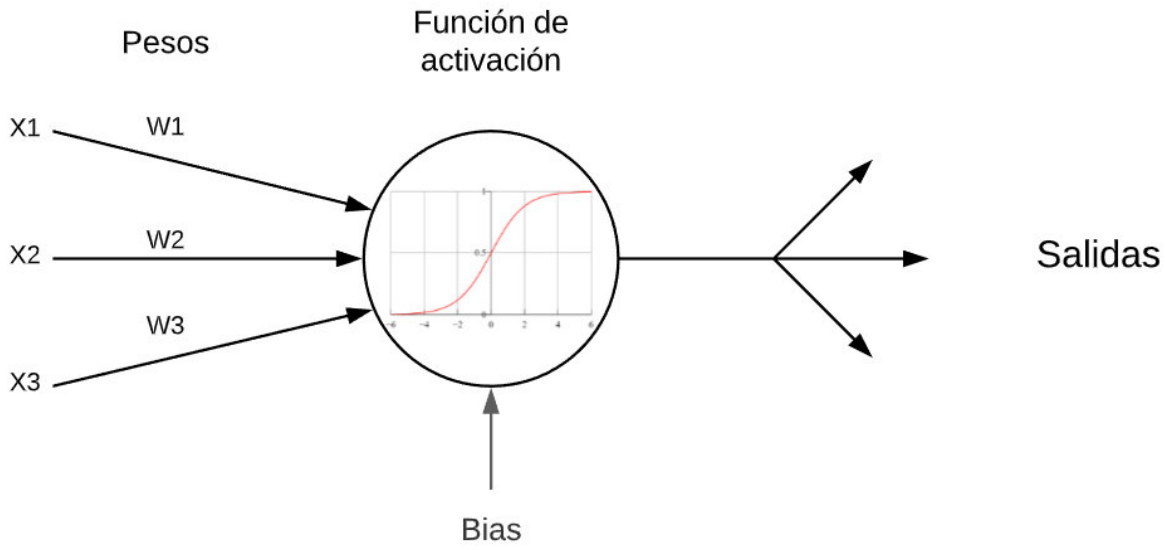


Figura 3. Arquitectura de una neurona

- Capa: es una representación que está compuesta por un conjunto de neuronas ordenadas, cada una de ellas está interconectada con la siguiente capa en la arquitectura. Por lo tanto, los datos de salida de una capa son los datos de entrada de otra, con excepción de la primera que recibe la información inicial. Según la funcionalidad de cada una se puede categorizar en tres:
    - Capa de entrada: Está compuesto por las neuronas que reciben los datos de entrada.
    - Capa de salida: Es el último elemento de la jerarquía, es el encargado de realizar alguna conclusión o predicción aportando datos de salida.

- Capa oculta: Es donde se realiza el procesamiento de información y se hacen los cálculos intermedios. No tiene comunicación con el "exterior" por lo que se utiliza para representaciones internas. Cuantas más neuronas existan en esta capa, más complejos son los cálculos que se efectúan. Las primeras capas ocultas de una red suelen detectar patrones simples, en cambio las últimas capas del modelo suelen detectar los patrones más complejos o los más detallados. En la figura 2 se ejemplificó con dos capas ocultas, pero en los modelos reales de deep learning suele haber más.

2.2 Redes Neuronales Convolucionales (CNN)

Una Red Neuronal Convolucional, más conocida como CNN debido a sus siglas en inglés, es una Red Neuronal Artificial con aprendizaje supervisado que aplica al menos una capa convolucional y es la principal arquitectura utilizada para el procesamiento de imágenes debido a la forma de matriz de las mismas.

Las CNN tienen el mismo objetivo que las redes neuronales clásicas pero lo realizan de forma diferente y de esta manera pueden ser más eficientes en velocidad para resolver un determinado grupo de problemas. Estas redes buscan actuar de manera similar a la corteza visual del cerebro humano debido a que nosotros, como personas, somos clasificadores eficaces y podemos distinguir entre distintos objetos que tengan características similares.

Están formadas por múltiples capas y filtros de distintas dimensiones, (no son sólo en 2D, como las imágenes), razón por la cual se implementan en este proyecto. Las mismas se dividen en dos importantes fases:

- Primera fase: se extraen las principales características que conforman al objeto en si, como pueden ser el color, los bordes y las sombras. En este tipo de redes se tienen varias capas convolucionales. A medida que se avanza en la red neuronal, cada capa reacciona menos que la anterior a los valores de entrada, de esta forma se logra una mayor abstracción.
- En la segunda fase ocurre la clasificación o regresión, se utilizan las denominadas capas densas, que se encarga de unir estas características recopiladas para determinar un resultado final.

2.2.1 Capas Convolucionales

Estas capas se denominan así debido a que operan sobre los datos de entrada mediante el cálculo de convoluciones discretas con bancos de filtros finitos. Estas buscan relaciones entre los pixeles de una imagen, al contrario de las redes tradicionales que buscan pixel por pixel. Esto permite tener un contexto más general y obtener resultados mucho mejores.

Un elemento fundamental en la capa convolucional es el filtro, el cual es una matriz completada con valores que se aprenden a través del entrenamiento. Los filtros aumentan la complejidad, por lo tanto, cuanta mayor cantidad de filtros se generen, mayor será la profundidad en la capa.

El proceso de convolución se explica a continuación presentando un breve ejemplo. Está conformado por distintos elementos, los cuales son:

- Una capa de entrada de 6x6 pixeles que por simplicidad está conformada por ceros y unos.
- Una ventana o cuadrado de 3x3 posicionado sobre la capa de entrada con un valor de deslizamiento igual a 1. (Como se analizará posteriormente en este informe, el deslizamiento puede ser mayor a 1.)
- Un filtro inicial de 3x3 con distintos valores.
- La capa convolucionada, la cual se está completando con la información que se obtiene en la convolución.

En la figura 4, se le aplica el filtro a la ventana verde situada sobre la capa de entrada. En esta iteración se multiplica cada valor de la ventana con el valor correspondiente en el filtro; luego, a estos valores, se los suma y se obtiene un resultado final que es el que se establece en la capa convolucionada. En este ejemplo, el resultado de esa sumatoria es 1, ya que 1*0+0*1+1*0+0*1+0*0+0*1+1*0+1*1+1*0 = 1.

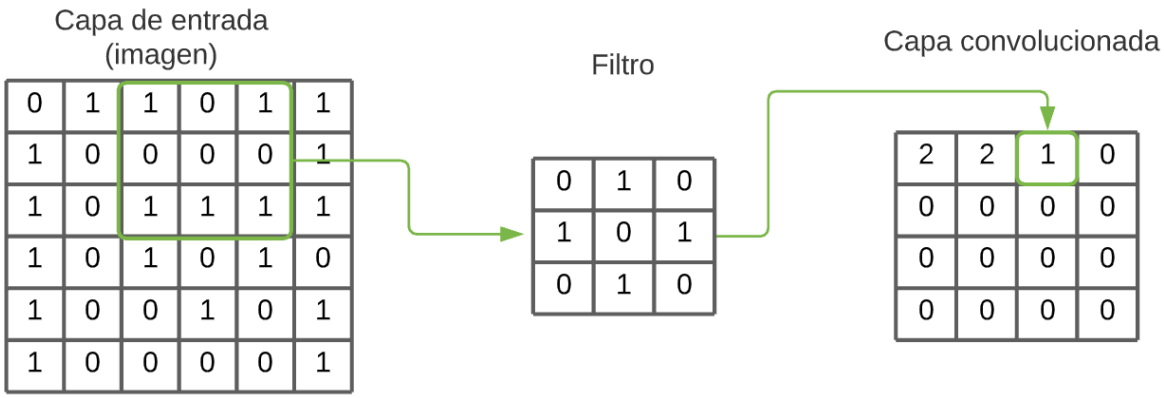


Figura 4. Proceso simple de una convolución

En la figura 5, la ventana se desplaza 1 pixel hacia la derecha, y se aplica la ecuación nuevamente. En este caso el resultado es 3. Una vez conseguido este valor, la ventana vuelve a desplazarse 1 pixel, hasta completar esta nueva matriz que contendrá las principales características de la capa de entrada.

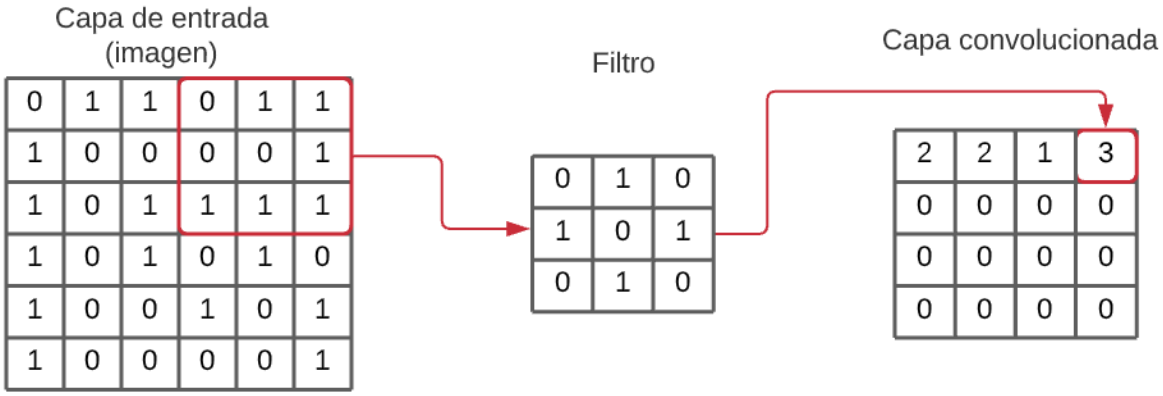


Figura 5. Desplazamiento de un pixel en una convolución

3 CASO DE ESTUDIO: MALEZAS

Como ya se ha mencionado anteriormente, el ML puede ser aplicado a casi cualquier campo de investigación para diferentes usos, tales como predecir información o detectar objetos. En este caso, este trabajo se enfoca en la detección de malezas comunes en Argentina, específicamente en la región pampeana.

3.1 LA AGRICULTURA Y LA TECNOLOGÍA

La agricultura no es un sector que se haya quedado atrás en la implementación de tecnologías para mejorar sus procesos. Se puede observar desde regadores automáticos en huertas, hasta drones para el monitoreo de cultivo en grandes terrenos. Estas tecnologías pueden describirse como “tangibles”; pero existen otras innovaciones que están basadas en el análisis de miles de datos ya existentes, que aunque no puedan verse, son muy eficientes para la optimización de procesos.

Técnicas de Machine Learning (ML) es una de ellas. Actualmente, ML es utilizado, junto con otras tecnologías, para la optimización de la producción y comercialización de cultivos de distintas maneras. Existen sistemas que permiten la personalización de técnicas de cultivo basadas en información en tiempo real, predicción sobre qué producto plantar, tendencias de consumo, mapeo de terreno fértil, entre otras. Estas distintas aplicaciones tienen un objetivo en común: mejorar los procesos de negocios para aumentar la eficiencia y conseguir la mayor cantidad de productos con el menor gasto posible.

3.2 LA PROBLEMÁTICA DE LAS MALEZAS

Se encuentran distintas amenazas que pueden afectar a los cultivos como por ejemplo: las plagas, los incendios, las inundaciones o los animales silvestres que ingieren los mismos. Esta investigación se centra en una amenaza específica, la aparición de distintas malezas, hierbas malas, o yuyos que afectan a la producción agrícola y a los procesos de cosecha, ya que en algunos casos esto es un impedimento para acceder a los cultivos.

Un ejemplo de esto es la maleza vulgarmente conocida como “yuyo colorado”. Según la Red de Manejo de Plagas de la Asociación Argentina de Productores en Siembra Directa (Aapresid), a finales del año pasado, en la provincia de Córdoba,

el “yuyo colorado” ocupaba 18.2 millones de hectáreas, este número es mayor a las hectáreas de cultivo de soja y maíz[1].

Existe una gran diversidad de malezas que surgen dependiendo principalmente de la zona geográfica y de la estación del año. La aparición y expansión de estos yuyos se debe a la resistencia a herbicidas que han desarrollado con el paso del tiempo, dificultando así el control de estos. En Argentina existen actualmente 15 malezas registradas como resistentes a herbicidas (glifosato) según un listado del Servicio Nacional de Sanidad y Calidad Agroalimentaria[2]. Por lo tanto, es importante detectar rápidamente la manifestación de estas hierbas indeseables, para evitar su propagación y reducir su banco de semillas.

En esta lista se encuentran dos de las tres malezas que se seleccionaron para el desarrollo de este proyecto, y ellas son: el Yuyo Colorado y la Rama Negra. Estas son comunes en las plantaciones de soja, uno de los cultivos que más se produce en el país. La tercer maleza restante es la Roseta, la cual es común en huertas y tiene la característica de ser “muy molesta” debido a que sus flores tienen pequeñas púas, lo que hace que en algunas regiones también perjudique al turismo.

---

[1] Asociación Argentina de Productores en Siembra Directa http://agrovoz.lavoz.com.ar/agricultura/en-cordoba-yuyo-colorado-ya-ocupa-mas-hectareas-que-soja-y-maiz

[2] Listado de malezas resistentes en Argentina http://www.senasa.gob.ar/casos-confirmados-de-malezas-resistentes-en-argentina

A continuación se describirá brevemente cada maleza, resaltando las características que diferencian unas de otras, información que el sistema desarrollado deberá detectar automáticamente.

- **Yuyo colorado** (Amaranthus quitensis): Esta especie posee, entre otras características, atributos biológicos que la convierten en una maleza agresiva y muy difícil de manejar eficazmente. Por lo tanto, el INTA la definió textualmente como “una amenaza permanente”[3]. Es originaria de América en las zonas cálidas y templadas. Es una de las malezas más importantes de los cultivos extensivos de verano, incluso puede afectar los de invierno en sus últimas etapas. Prefiere suelos fértiles y algo arenosos, aunque es muy plástica y prospera también en otros tipos de suelos.

[3] El yuyo colorado denominado como una maleza resistente https://inta.gob.ar/documentos/yuyo-colorado-una-amenaza-permanente

Figura 6. Yuyo colorado (Amaranthus quitensis)

- **Rama negra** (Conyza bonariensis): Es una especie que se encuentra en Bolivia, Brasil, Paraguay, Uruguay, Chile, además del norte y centro de Argentina hasta Río Negro. Es muy común como ruderal, es decir, se la encuentra en las esquinas de los caminos y espacios transitados por personas, como maleza de huertas y jardines. En cultivos extensivos es una de las malezas más importantes en la finalización tanto del ciclo de cultivos de invierno como en los de verano y en pasturas. También es utilizada en medicina popular. Esta especie no es una plaga, pero si está catalogada como resistente a herbicidas.

Figura 7. Rama negra (Conyza bonariensis) en un lote de trigo

- **Roseta o cardillo chico (Cenchrus insertus - Cenchrus pauciflorus**): Es una especie autóctona, del norte y centro de la Argentina. Es una maleza importante de los cultivos extensivos de verano en los sitios secos y arenosos de la región pampeana. Es frecuente como planta ruderal y es una maleza muy molesta ya que posee pequeñas espinas en sus flores. Esta especie no es una plaga y no está catalogada como resistente a herbicidas.

Figura 8. Roseta (Cenchrus insertus- Cenchrus pauciflorus)

4 DESARROLLO

4.1 HERRAMIENTAS Y ESPACIO DE TRABAJO

En esta sección se describirán brevemente las tecnologías utilizadas para el presente proyecto y posteriormente algunas librerías importantes para el desarrollo de la aplicación.

- Windows 10: Sistema Operativo en donde se estableció el espacio de trabajo.

- PyCharm Community: IDE (Entorno de desarrollo integrado) utilizado para el desarrollo de toda la aplicación. La versión community permite depurar y programar cómodamente, además la consola que proporciona ha sido de mucha utilidad para la instalación de las librerías y paquetes necesarios.
- Python 3.6[4]: Lenguaje de programación de alto nivel utilizado para la construcción del modelo y el entrenamiento del mismo. Se hace uso de la versión 3.6 debido a que es la versión más nueva que Tensorflow2 soporta.
- PIP: Es un manejador de paquetes para python que permite instalar otras librerías y dependencias fácilmente.
- Tensorflow 2[5]: Es una biblioteca open source desarrollada por Google que facilita el desarrollo de sistemas de Deep Learning proporcionando APIs de alto y bajo nivel. Inicialmente se usó la versión 2.2.0, pero dado que una nueva actualización facilitaba el desarrollo del proyecto se decidió actualizar el mismo a la versión 2.3.1. Esta herramienta ha sido la base del proyecto ya que permite construir modelos de ML de forma sencilla con su integración con la API Keras.
- Colaboratory: Es una herramienta web de Google que permite realizar y compartir documentos ejecutables dentro de Google Drive. El mismo permite

[4] Página oficial de Python https://www.python.org/

[5] Página oficial de Tensorflow https://www.tensorflow.org/

ejecutar código escrito en python sin la necesidad de realizar configuraciones de ambiente en la maquina con la que se está utilizando la herramienta. La principal ventaja de esta aplicación es que otorga un acceso gratuito a GPUs, por lo tanto permite realizar pruebas que requieren mucho procesamiento.

- GitLab: es una plataforma para la gestión de versiones y repositorios, basado en git. Si bien esta aplicación se desarrolló de forma individual, el control de versiones es muy importante para no perder información.

Entre las principales librerías utilizadas en el ambiente desarrollado en Python se encuentran:

- Keras: Es uno de los componentes más importantes de Tensorflow 2.0, se trata de una API de alto nivel para construir y entrenar modelos. Al ser de alto nivel, se sacrifica algo de flexibilidad y performance a cambio de la facilidad de uso que otorga. Además, es una herramienta que sirve para importar las imágenes para el entrenamiento y la prueba, sin necesidad de instalar otra librería.
- Matplotlib: una librería para python utilizada para crear visualizaciones, mostrar imágenes o crear gráficos. Muy útil para verificar que las imágenes se estén leyendo correctamente y tener una mejor visualización de la tasa de éxito.
- Pillow: Es una bifurcación de PIL (Python Image Library) que ha agregado más características y permite abrir, editar y crear imágenes. Actualmente es una de las librerías de manipulación de imágenes más populares.

- NumPy[6]: Es un paquete muy popular para la ciencia computacional. NumPy nos permite, entre otras cosas, manejar de forma eficiente matrices complejas, como lo es una imagen.

4.2 SELECCIÓN Y EDICIÓN DE IMÁGENES

Se realizó una cuidadosa recopilación de imágenes sobre las tres malezas seleccionadas. Es importante que cada una sea representativa de la maleza, en donde la misma destaque del fondo para que la red neuronal "entienda" qué es lo que está clasificando. En la figura 9 se muestran algunas de las imágenes que se ingresaron para entrenar la red neuronal. A este conjunto donde la información es la imagen y la categorización que le corresponde se lo conoce como **dataset.**

[6] Página oficial de NumPy https://numpy.org/

Figura 9. Dataset de malezas

Firma Estudiante:	Firma Docente Supervisor:	Firma docente tutor TAPTA:	Firma tutor Organizacional:

Inicialmente se entrenó la red neuronal con 15 imágenes por maleza, y 5 para realizar la validación de la misma. Estas tienen una dimensión de 400x400 pixeles y se encontraban distribuidas en dos carpetas, una llamada “train” (entrenamiento) y la otra “test” (validación o pruebas.)

Los nombres de los archivos son importantes y caracterizan a cada una, los cuales estarán compuestos de la siguiente forma:

1. Un valor entero y consecutivo iniciando con el número 1 por cada maleza.
2. Un guión medio (-).
3. Un número que caracterizará la categoría a la que pertenece.

Este último se especifica en la tabla que se muestra a continuación. Por ejemplo, un nombre válido seria el “13-1”, que significaría que es la imagen número 13 del yuyo colorado. De esta forma, era sencillo para el programa categorizar una imagen.

Maleza	Número asociado
Yuyo colorado	1
Rama Negra	2
Roseta	3

Tabla 1. Número asociado a cada maleza.

En medio del desarrollo, hubo una actualización de la herramienta TensorFlow 2.2.0 a 2.3.1, la cual contenía importantes mejoras en la carga de imágenes, por lo que

se cambió este modelo, en detrimento de otro basado en directorios, en donde cada carpeta representaba un tipo de maleza, y únicamente contenía las imágenes de esa misma, tal como se muestra en la Figura 10.

El dataset original se siguió manteniendo con un total de 60 imágenes del cual el 10% se utilizará para validación. Además ahora la aplicación dividirá las imágenes para entrenamiento y para validación según el porcentaje proporcionado, es decir que ya no existen las carpetas “train” y “test” del dataset anterior.

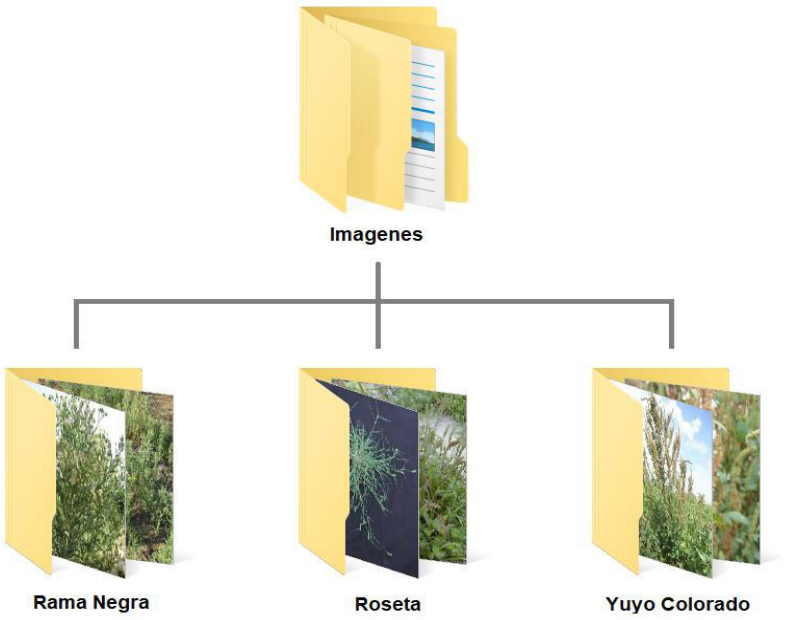


Figura 10. Esquema de directorios

Adicionalmente, para organizar el dataset se hizo uso de una aplicación web llamada Roboflow[7], una herramienta muy útil para almacenar y compartir datasets en un equipo. En la figura 11 se muestra un ejemplo de la administración del dataset, en donde se especifican las imágenes para entrenamiento, pruebas y validaciones.

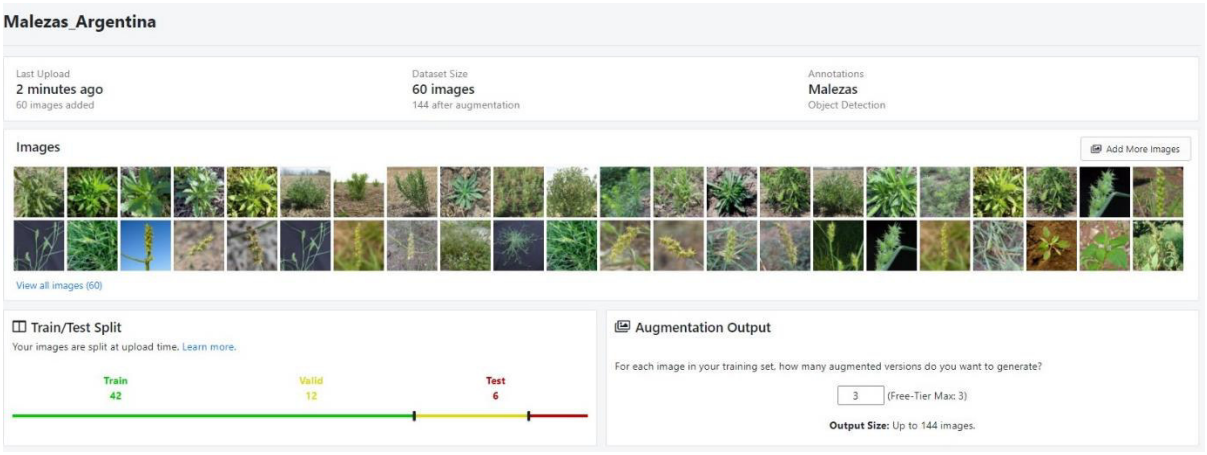


Figura 11. Pantalla de administración del dataset Roboflow

4.3 EL MODELO DE CATEGORIZACIÓN

En esta sección se desarrollará sobre el modelo final, debido a que el mismo se fue adaptando y modificando para poder lograr mejores resultados. El mismo se encuentra resumido en la tabla 2 y está compuesto por 17 capas, cada una con funciones diferentes y parámetros distintos. El modelo está basado en la arquitectura VGG16 [6], el cual consiste en utilizar bancos de capas convolucionales

[7] https://roboflow.com/

seguidas de una capa de max-pooling. Entre sus principales desventajas se encuentra que se requiere mucho más procesamiento que otras arquitecturas debido a que al usar bancos o lotes de capas agrupadas ocasiona que la red sea más compleja.

N°	Nombre	Tipo de capa	Forma de salida	Parámetros
1	random_flip	RandomFlip	(None, 400, 400, 3)	0
2	random_rotation	RandomRotation	(None, 400, 400, 3)	0
3	random_zoom	RandomZoom	(None, 400, 400, 3)	0
4	rescaling	Rescaling	(None, 400, 400, 3)	0
5	conv2d	Conv2D	(None, 398, 398, 16)	448
6	conv2d_1	Conv2D	(None, 396, 396, 16)	2320
7	max_pooling2d	MaxPooling2D	(None, 198, 198, 16)	0
8	conv2d_2	Conv2D	(None, 196, 196, 32)	4640
9	conv2d_3	Conv2D	(None, 194, 194, 32)	9248
10	max_pooling2d_1	MaxPooling2D	(None, 97, 97, 32)	0
11	conv2d_4	Conv2D	(None, 95, 95, 64)	18496
12	conv2d_5	Conv2D	(None, 93, 93, 64)	36928
13	max_pooling2d_2	MaxPooling2D	(None, 46, 46, 64)	0
14	flatten	Flatten	(None, 135424)	0
15	dense	Dense	(None, 128)	17334400
16	dropout	Dropout	(None, 128)	0
17	dense_1	Dense	(None, 3)	387

Tabla 2. Resumen del modelo implementado

A continuación se explicará capa por capa implementada:

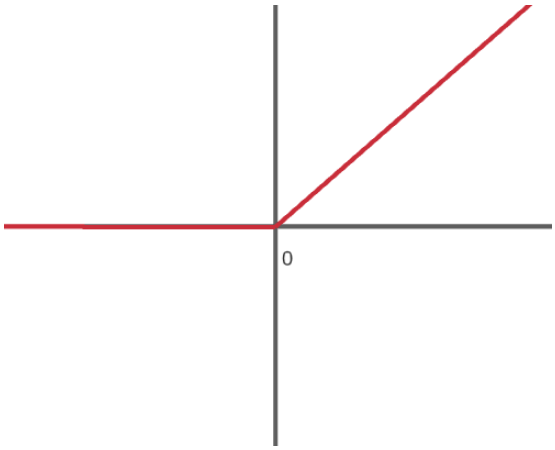
- Capas de Data Augmentation: Como se ha mencionado antes, el dataset que se dispuso para trabajar es muy acotado, ya que sólo tiene 60 imágenes en

total, y por lo general se suele entrenar redes convolucionales con más de mil imágenes. Una solución que se aplicó para contrarrestar este problema es la de desarrollar tres capas de Data Augmentation (random_flip, random_rotation y random_zoom), las cuales consisten en aumentar la cantidad de imágenes que se disponen aplicando en ellas pequeños cambios, como rotaciones o zoom. De esta forma se amplifica el dataset ingresado con varias imágenes tomadas desde ángulos diferentes.

- Capa de pre procesamiento (Rescaling): Se añade una capa de pre procesamiento en donde la matriz que representa a la imagen se edita para que en vez de ser valores de entre 0 y 255 (según el RGB), sean valores de entre 0 y 1. La misma se encuentra nombrada en la tabla 2 como capa de rescaling.
- Capas de convolución (Conv2D): Se añadieron seis capas de convolución en 2D, dos por cada banco de capas. A continuación se explican los parámetros establecidos en cada una:
    - Función de Activación: es la encargada de devolver una salida. Existen múltiples funciones de activación, cada una se adecua más a un problema. En el caso de las Redes Neuronales Convolucionales, la función más utilizada es la función de activación de unidad lineal rectificada, o más conocida como ReLU por sus siglas en inglés. Es la más popular y utilizada debido a que se ha demostrado de forma experimental que es una de las más eficaces para desarrollar redes profundas, tanto como por su tasa de éxito como por su eficiencia computacional, ya que no utiliza funciones complejas al ser una

función lineal a partir de 0. Su principal característica es que a diferencia de otras funciones, no está acotada a un rango de números, en la ecuación 1 se visualiza la función de activación ReLu. Ésta es la función que más se adecua al problema establecido, y por lo tanto la que se seleccionó para cada capa convolucional del modelo desarrollado.

$$f(x) = max(0, x) = \begin{cases} 0 = x < 0 \\ x = x \geq 0 \end{cases}$$



Ecuación 1. Función de activación ReLU

- Filtros: Es un valor numérico entero que representa los canales de convolución o el conjunto de kernels. Una buena práctica es que el valor de los filtros de las primeras capas de la red sea menor que las últimas y que además sean valores iguales a las potencias de 2. Esto significa que a medida que el volumen de información de las capas de

salida va disminuyendo, la cantidad de filtros va aumentando, algo que es típico de las redes neuronales convolucionales. La principal razón de esta práctica es que en las primeras capas se obtienen características sencillas, como los bordes o el color, en cambio en las capas finales, las características que se obtienen son más complejas, por lo cual es necesario un mayor número de filtros. Debido a esto, en el modelo, se implementó lo siguiente:

- Primer lote de capas: dos capas de 16 filtros. Representadas en la tabla 2 como conv2d y conv2d_1.
- Segundo lote de capas: dos capas de 32 filtros. Representadas en la tabla 2 como conv2d_2 y conv2d_3.
- Tercer lote de capas: dos capas de 64 filtros. Representadas en la tabla 2 como conv2d_4 y conv2d_5.

- Kernel size: es el tamaño del filtro que se empleará. El tamaño depende de la dimensión de la capa, por ejemplo, en una convolución 1D, el tamaño del kernel es una tupla; en el caso de una 2D (como se trata en este proyecto), el tamaño del kernel es de dos dimensiones, típicamente cuadrado aunque pueden existir filtros asimétricos. En este caso se optó por usar un cuadradado de 3x3 pixeles.
- Stride: Es el deslizamiento de ventana empleado, por defecto el valor es de 1, y el mismo se mantuvo ya que esto causa que haya solapamiento entre filtrado al moverse pixel por pixel.

  - Padding: Hace referencia a si se tendrán en cuenta o no los bordes de las imágenes ingresadas. En este caso, los bordes de la imagen se tienen en cuenta en esta red neuronal debido a que las mismas fueron previamente procesadas de forma manual.

- Capas de Max-Pooling (MaxPooling2D): Antes de explicar el porqué de estas capas, es necesario entender una problemática. Cada imagen ingresada es a color (RGB) y tiene un tamaño de 400x400 pixeles. Esto implica que en la capa de entrada del modelo existan un total de 480000 neuronas (400x400x3), luego de la primer convolución, la cantidad de neuronas incrementa a 768000000, ya que se agregan los 16 filtros o mapas. Este número es demasiado grande y volverá a incrementar en la próxima convolución, lo que además implica más procesamiento. Estas capas son fundamentales, ya que sin ellas no se puede crear el modelo con los recursos que se disponen, debido a que al compilar la aplicación se obtiene un error de “recursos insuficientes” en la consola, y finaliza el proceso. Para solucionar este problema, se agrega luego de cada capa de convolución una de Max-Pooling (capas max_pooling2d, max_pooling2d_1 y max_pooling2d_2 en la tabla 2), cuya función es “quedarse” con las características más importantes de cada filtro y reducir la salida a la mitad de información. Para poder visualizar de forma práctica esta diferencia de parámetros, se realizaron dos pruebas en el modelo plasmadas en la tabla 3. En el primer caso, el modelo cuenta con una sola capa de Max-Pooling que se encuentra ubicada seguida de la segunda capa convolucional. El segundo caso es el que se emplea en el modelo, que es de una capa de Max-

Pooling luego de cada capa convolucional. La diferencia entre ambos modelos es notoria, ya que el primer caso tiene una cantidad de parámetros 16 veces mayor que el segundo, un hecho que no solo se nota analizando los números, sino que también se nota en la velocidad de compilación del modelo, el cual se ve seriamente afectado por la cantidad de parámetros.

Nombre de capa	#Parámetros con 1 capa de m-p	#Parámetros con 1 capa de m-p luego de cada capa convolucional
Convolucional 2D 1	448	448
Convolucional 2D 2	4.640	4.640
Convolucional 2D 3	18.496	18.496
Densa (final)	314.701.000	18.874.496

Tabla 3. Diferenciación entre la cantidad de parámetros al implementar modelos con distintas cantidades de capas de Max-Pooling

El parámetro más importante de este tipo de capa es el del tamaño de la piscina o ventana, conocido en inglés como pool size, para esta implementación se escogió el tamaño de 2x2, que es el valor por defecto y reduce a la mitad la información, en la figura 12 esta ventana está representada por los distintas zonas de colores de tamaño de 2x2. Si bien el Max-Pooling es beneficioso para la red, ya que se queda con las características más representativas y reduce la información, no hay que

abusar del mismo, ya que se pueden perder características importantes si por ejemplo se aumenta la ventana al tamaño de 3x3. Cuanto mayor sea la ventana, mas características se perderán. En la figura 12 se muestra un ejemplo del Max-Pooling 2x2 en donde en la salida se devuelven los valores máximos de la entrada.

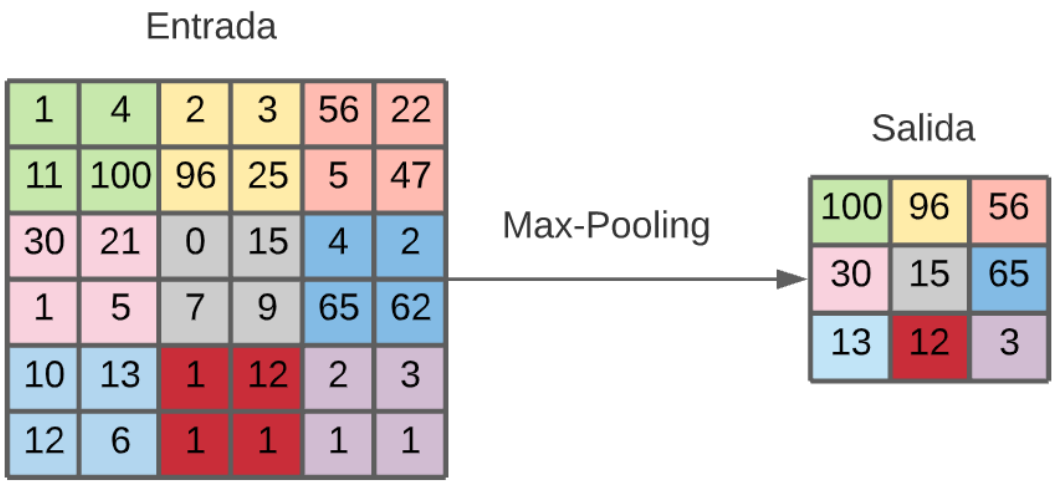


Figura 12. Ejemplificación del funcionamiento de la capa de Max-Pooling

- Capas densas (Dense): hace referencia a las capas clásicas. Se sitúa normalmente al final de la red convolucional y hace la labor de relacionar todas las características y todos los datos que se procesaron previamente para obtener una clasificación o una regresión final. En este modelo, se añadieron dos capas densas, ambas ubicadas al final de la red, en donde la última capa es la responsable de dar el resultado final. Estas capas se encuentran nombradas en la tabla 2 como dense y dense_1. Los parámetros de la misma son las siguientes:

- Unidades: la cantidad de neuronas clásicas que va a tener la capa densa. Para la primera capa se agregaron 128 neuronas, sin embargo para la segunda capa se agregó la cantidad total de clases utilizadas, que en este caso es de tres clases (yuyo colorado, rama negra y roseta).
- Función de activación: Al igual que en las capas de convolución, la función de activación es la encargada de devolver una salida. En esta capa también se optó por la función de unidad lineal rectificada, conocida como ReLu.

- Capa de Dropout: Es una capa reguladora que actúa comúnmente sobre las capas densas, la misma no es obligatoria en los modelos de CNN pero si es muy útil para atenuar una problemática muy conocida en Machine Learning que se denomina como overfitting. Esta capa realiza su labor únicamente en el entrenamiento, y se encarga de eliminar contribuciones de las neuronas de forma aleatoria en cada iteración, tal como se muestra en la figura 13. El objetivo de la eliminación es que las neuronas no dependan unas de otras y puedan “aprender” en conjunto. El parámetro más importante en esta capa es el porcentaje de eliminación que suele oscilar entre el 20% y el 60%, un porcentaje mayor al 60% ocasionaría el problema opuesto al overfitting, el cual es el underfitting, que significaría que la red no está aprendiendo lo suficiente. Para este modelo se optó por un porcentaje del 30%. En la tabla 2 se visualiza que esta capa se encuentra seguida de la capa “dense”, a la cual le aplica el dropout.

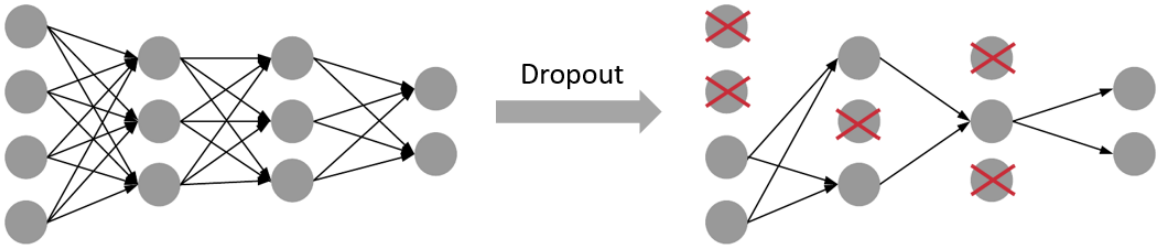


Figura 13. Funcionamiento del dropout

4.4 RESULTADOS DE LA CATEGORIZACIÓN

A continuación se presentarán los distintos resultados conseguidos utilizando el modelo planteado y los principales hallazgos.

El modelo se ejecutó un total de 30 épocas, con una duración de 4 minutos; en la figura 14 se puede visualizar la precisión del entrenamiento (representado con color naranja) y de la validación (representado con color azul). Se puede deducir que el modelo finalizó con un *overfitting* bastante considerable, ya que el valor de precisión de entrenamiento fue de 0.78 y la de validación fue de 0.63, dando así una diferencia de 0.15. Esto se traduce como que el modelo falla al reconocer ciertos tipos de maleza, debido a que las mismas no tienen exactamente los mismos valores que las imágenes con las que se entrena la red. Lo ideal para una red convolucional es que la precisión del entrenamiento sea mayor a la precisión de la validación pero con una diferencia mínima.

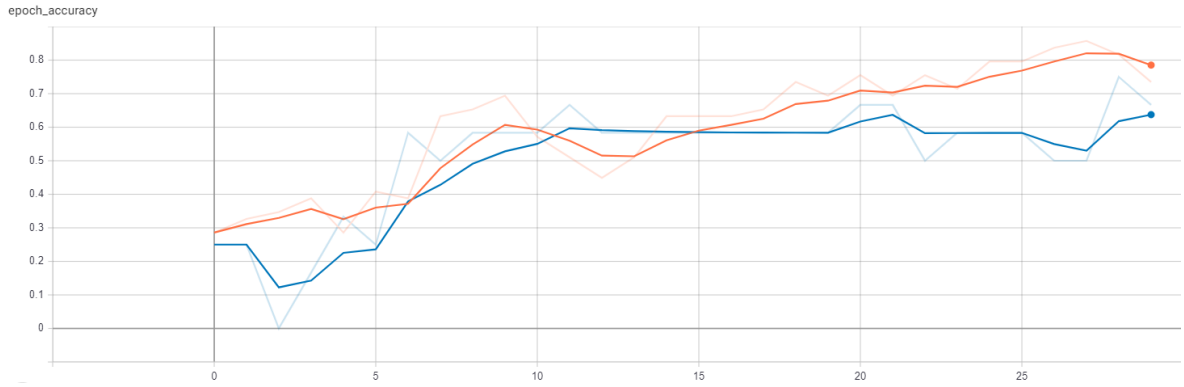


Figura 14. Gráfico de precisión de la red, representado en naranja el entrenamiento y en azul la validación

Una vez entrenada la red, se procede a realizar pruebas con imágenes que nunca hayan sido vistas por el modelo, por lo tanto, se seleccionaron 10 imágenes de cada maleza que sean completamente nuevas para la red, y se desarrollaron las pruebas pertinentes. En la siguiente figura (15) se representan los hallazgos encontrados.

		Predicciones: Yuyo colorado	Rama negra	Roseta
Reales	Yuyo colorado	3	3	4
	Rama negra	1	8	1
	Roseta	10	0	0

Figura 15. Matriz de los resultados generales

Firma Estudiante:	Firma Docente Supervisor:	Firma docente tutor TAPTA:	Firma tutor Organizacional:

Como primera conclusión, se visualiza que hay dos malezas que se destacaron; en primer lugar, la rama negra por su éxito en las pruebas y en segundo lugar, la roseta por el fracaso en las mismas. Sobre ambas se explayará más detalladamente en las respectivas secciones de los resultados.

4.4.1 Resultados de las malezas: Rama negra

De las 30 imágenes que se probaron, únicamente cinco fallaron, otorgando una tasa de éxito del 83.33% de éxito. Los resultados se diagramaron en una matriz de confusión (tal como se muestra en la figura 16). De esta forma, es sencillo visualizar en dónde se cometieron la mayor cantidad de errores.

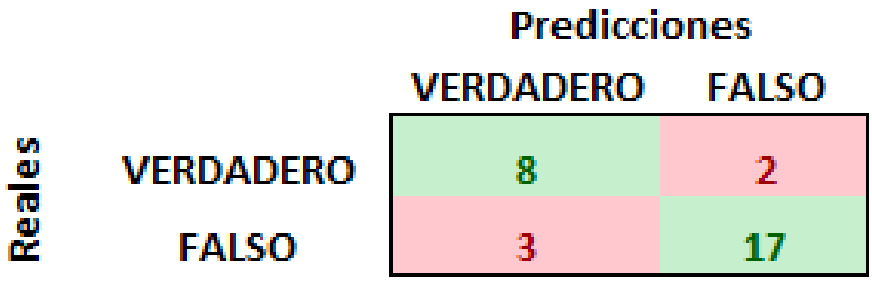


Figura 16. Matriz de confusión: Rama negra

En la matriz presentada se visualiza que tres ilustraciones dieron un falso positivo, y dos dieron un falso negativo.

A continuación se intentará evaluar alguno de estos errores, y otras pruebas de interés. En la figura 17, se muestra una prueba fallida de un falso positivo, en donde

un lote de rama negra en un campo de trigo fue diagnosticado como un yuyo colorado con un total de 83.42% de probabilidad, el cual es bastante alto, quedando en segundo lugar la rama negra con un 16.58% de probabilidad. Una confusión que parece repetirse en varias pruebas, pero se explayará en la sección correspondiente al yuyo colorado

Figura 17.Rama negra diagnosticada como Yuyo colorado.

Figura 18. Una predicción exitosa de la rama negra, con un 30% de probabilidad de que sea un yuyo colorado

Como se detalló en la matriz de confusión (figura 16), también existieron falsos positivos, como es el caso de la figura 19, en donde claramente se distingue en la imagen que es un yuyo colorado; sin embargo, el modelo predijo que era una rama negra con un 91% de precisión, el cual es alto.

Figura 19. Predicción de rama negra cuando la maleza es un yuyo colorado. Falso positivo

4.4.2 Resultados de las malezas: Yuyo Colorado

En el análisis de la rama negra se hizo mención sobre la gran confusión que género el yuyo colorado, debido a que varias imágenes, en especial las que eran de la

roseta, se predijeron como yuyos colorados, cuando en la realidad no era así. En la figura 20 se detalla el problema planteado en una matriz de confusión.

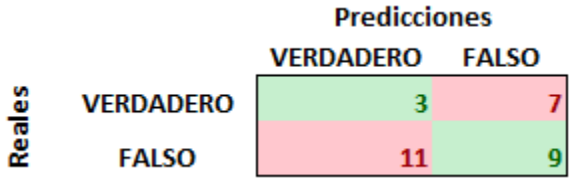


Figura 20. Matriz de confusión: Yuyo colorado

Como se puede visualizar, la precisión final del yuyo colorado fue el más bajo con 40%, habiendo obtenido solo tres verdaderos positivos y una gran cantidad de falsos positivos. En la figura 17 se visualiza este hecho, dado que el modelo predijo que una rama negra era un yuyo colorado. Pero en donde más se repitió este error fue en la roseta (tal como se demuestra en la figura 21) en donde se predice con un 99.98% de acierto que la imagen es un yuyo colorado, un porcentaje preocupantemente alto para ser un error.

Figura 21. Roseta diagnosticada como Yuyo colorado

Este error puede darse por varios motivos, pero principalmente se destaca que faltaron imágenes de la roseta y del yuyo colorado para que el sistema aprendiera a diferenciarlos a ambos, ya que también existieron cuatro casos en donde el yuyo

colorado fue confundido con una roseta, aunque no con porcentajes tan altos, tal como se visualiza en la figura 22.

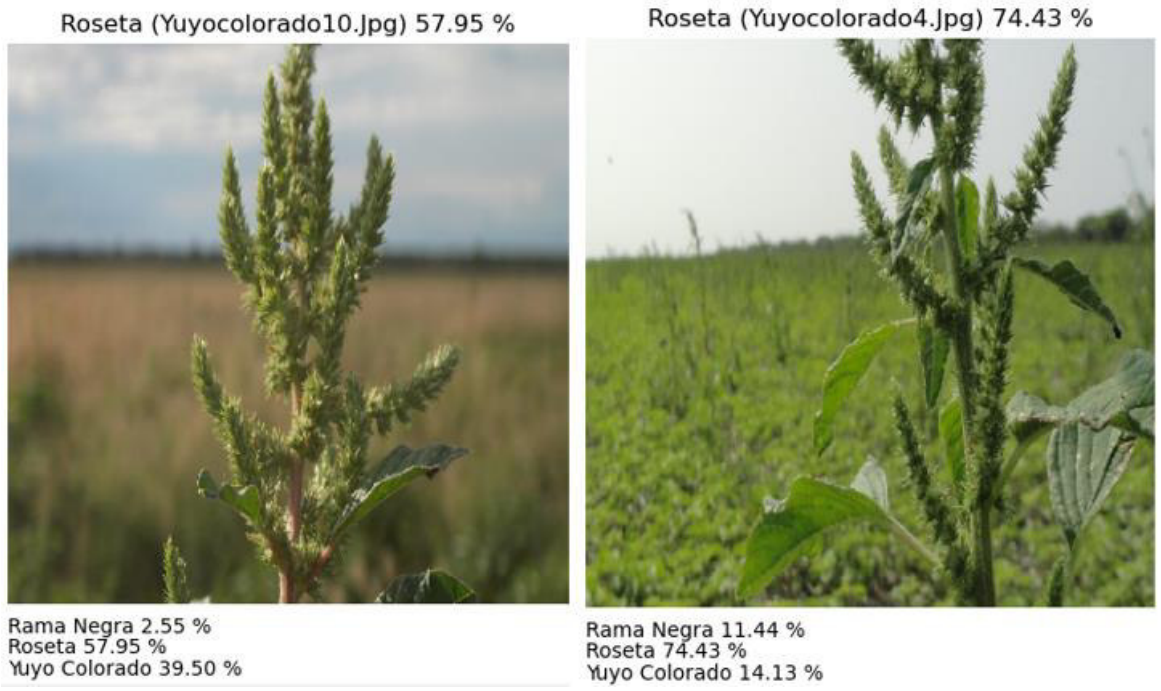


Figura 22. Yuyo colorado diagnosticado como roseta

El sistema igualmente identificó casos correctos en donde la maleza fue bien catalogada y con un alto porcentaje de precisión, como es en el caso de la figura 23.

Figura 23. Yuyo colorado correctamente predecido

4.4.3 Resultados de las malezas: Roseta

Como se mencionó anteriormente, la roseta fue la maleza con los peores resultados finales, finalizando con una presición del 50% y sin haber logrado ninguna

predicción verdadera y positiva, lo cual permite analizar distintos problemas en el presente modelo. En la figura 24 se muestra la matriz de confusión para la roseta

		Predicciones	
		VERDADERO	FALSO
Reales	VERDADERO	0	10
	FALSO	5	15

Figura 24. Matriz de confusión: Roseta

Dada la matriz de confusión se identifica que ninguna imagen de la roseta se detectó como tal, algo que bajo significativamente la tasa de éxito tanto como el de la roseta como el del yuyo colorado. Como primera observación, es evidente que al sistema le faltó información sobre la roseta para poder realizar predicciones correctas, ya que el mismo lo confundió completamente por el yuyo colorado, tal como se demostró en la matriz de resultados generales (figura 15) y se mostró de forma práctica en las figuras 21 y 22.

4.4.4 Conclusiones de los resultados

Se concluye que el modelo final implementado “aprendió” a categorizar correctamente a la Rama Negra con una precisión general de 83.33%, es decir, puede diferenciar correctamente que una maleza es una rama negra y no una roseta o un yuyo colorado. Sin embargo, estos buenos resultados no se reflejaron en las dos malezas restantes. Ambos tuvieron una tasa de éxito muy baja y afectaron de forma negativa a los resultados finales. Esta falla se debe a dos motivos principales:

- La escasez de imágenes. Este es el principal problema del sistema en general, el cual no le permite al modelo 'aprender' lo suficientemente bien, ya que no ha visualizado una cantidad significativa de malezas.
- La roseta y el yuyo colorado son estéticamente parecidos. Si uno visualiza únicamente las flores, ambos tienen un aspecto como puntiagudo y verde, además, sus tallos también son verdes, aunque el del yuyo colorado es más grueso que el de la roseta. Esta comparación entre ambas malezas se puede visualizar con mayor detalle en la figura 25.

Figura 25. Comparación visual entre una roseta (a la izquierda) y un yuyo colorado (a la derecha)

4.5 TRABAJO FUTURO

Ningún sistema es perfecto, por lo que siempre pueden aplicarse mejoras y nuevas funcionalidades para enriquecer al mismo. En la aplicación que aquí se presenta, se detectaron las siguientes mejoras aplicables que harían un cambio significativo a la funcionalidad del mismo.

- Probar el sistema en un ambiente real, es decir, en una huerta que presente algún tipo de maleza. Esta mejora proporcionaría datos reales a la aplicación, y sería el caso de *prueba ideal.* Cabe aclarar que este caso de prueba se vio directamente afectado por la cuarentena ocasionada por la pandemia del COVID 19, que impidió la movilidad.
- Diseñar y desarrollar una pequeña aplicación web multi-plataforma en donde se puedan cargar imágenes en tiempo real para ser procesadas y obtener una respuesta que sea correcta. La respuesta debería ser un “Si, hay maleza” o “No, no hay maleza”. Esta nueva funcionalidad le agregaría un nivel de utilidad mayor al que hoy se presenta.
- Una de las mejoras más importantes es la de conseguir más imágenes de malezas, ya que esto le permitiría a la aplicación hacer una categorización más certera y obtener una mayor tasa de éxito. Este problema se intentó disminuir aplicando la técnica de data augmentation. Sin embargo, la cantidad de imágenes sigue siendo insuficiente.
- Implementación de Transfer Learning (Transferencia de Aprendizaje, en español). Es una técnica muy conocida en el deep learning que permite

utilizar información ya conocida a un problema nuevo. Se aplica reutilizando redes neuronales con información similar ya entrenadas, de esta forma se mejora el rendimiento y la velocidad de entrenamiento.

5 CONCLUSIÓN

El desarrollo del trabajo presentado se basó fundamentalmente en cuatro etapas: 1) investigación sobre la problemática de las malezas en los cultivos de la región; 2) recolección de un conjunto de imágenes que fueron procesadas por modelos de machine learning; 3) estudio de las principales características de diferentes técnicas de Machine Learning; y 4) diseño, desarrollo e implementación de un sistema de detección de malezas en cultivos mediante el procesamiento de imágenes utilizando machine learning.

En la primera etapa se pudo descubrir cómo las malezas afectaban los diversos cultivos de la región. Si bien existen muchos tipos de malezas que provocan esto en mayor o menor medida, se seleccionaron únicamente tres, dependiendo de su significancia en base a distintos parámetros, tal como la resistencia a herbicidas, frecuencia de aparición, las zonas de crecimiento, etcétera.

La segunda etapa consistió de la selección de imágenes, presentándose de esta forma la primera gran problemática que se tuvo en el proyecto: es difícil conseguir un dataset tan específico. Motivo por el cual se tuvo que armar uno propio, que puede seguir ampliándose a medida que se hallen más imágenes. Este problema

fue un reto en el desarrollo del proyecto, ya que a partir del mismo surgían otros problemas distintos a los que fueron presentados en el inicio de este trabajo.

En la tercera etapa se logró investigar diferentes técnicas de ML. Por ende, se obtuvieron los conocimientos teóricos que se aplicarían a lo largo del todo proyecto. Esta etapa incluyó el aprendizaje de distintas técnicas, arquitecturas, metodologías, herramientas, entre otros. Los principales conocimientos adquiridos que se pueden destacar son Tensorflow, CNN, Max-Pooling, Transfer Learning y distintas funciones de activación.

La última etapa fue la más larga, y ya involucra el uso de todos los conocimientos obtenidos para desarrollar una aplicación de detección y procesamiento de imágenes que supere las expectativas iniciales con una tasa de éxito mayor al 70%, el cual fue un objetivo cumplido. En el transcurso de esta etapa se crearon varios modelos, algunos con diferencias minúsculas, y cada uno fue testeado para poder lograr el modelo que mejor cumpliera con el objetivo.

Finalmente, se espera continuar con este trabajo en las líneas futuras propuestas para que pueda aplicarse a sistemas reales y ser beneficioso para la región. Por ello, se replantearían varios objetivos acá presentados, como una tasa de éxito mayor al 80%, aumentar la cantidad de malezas identificadas, y seguir estudiando las diferentes arquitecturas y técnicas del Machine Learning para obtener finalmente un modelo mejor.

6 BIBLIOGRAFÍA

1. Arriaga, Ezequiel; López de Sabando, Marcelo (2015), *Malezas problema: yuyo colorado*, Argentina Publicado por Instituto Nacional de Tecnología Agropecuaria, Disponible en https://inta.gob.ar/sites/default/files/malezas_problema_yuyo_colorado.pdf (Consultado el 24/04/2021)
2. Alesanco, Sahagún; Rodríguez, Pablo (2018), *Aplicación de redes neuronales convolucionales y recurrentes al diagnóstico de autismo a partir de resonancias magnéticas funcionales*, España Publicado por Universidad Politécnica de Madrid
3. Chollet, François (2017), *Deep Learning with python*, Estados Unidos Editorial Manning
4. Díaz Panizza, Lucas; Zubizarreta, Lorena (2014), *Guía de reconocimiento de malezas*, Argentina Publicado por SYNGENTA AGRO S.A. Disponible en https://www.syngenta.com.ar/product/crop-protection/herbicida/landing-control-de-no-malezas (Consultado el 24/04/2021)
5. Simonyan, Karen; Zisserman, Andrew (2015), *Very deep convolutional networks for large-scale image recognition*, Oxford Publicado por University of Oxford.
6. Torres, Luz Gloria (1995), *Redes Neuronales y Aproximación de Funciones*, Colombia Publicado por Universidad Nacional de Colombia.

7 ANEXO: CÓDIGO IMPLEMENTADO

En esta sección se mostrará el código elaborado para este trabajo. Hay que recordar que la implementación realizada es una adaptación de la arquitectura VGG16, por lo tanto, las capas pueden ser distintas.

1. Importación: se importan las librerías a utilizar, de las cuales se destacan:
    a. Tensorflow: permite construir el modelo
    b. Keras: brinda facilidad en la construcción del modelo.
    c. Matplotlib: para realizar gráficos.
    d. Data: es una clase propia que se encarga de importar las imágenes que se utilizaran para el entrenamiento.

```
from __future__ import absolute_import, division, print_function, unicode_literals

from classes.Data import Data
import matplotlib.pyplot as plt
from tensorflow.keras import layers, models
import tensorflow as tf
import datetime
```

Anexo - Figura 1. Importación de librerías

2. Set de variables: se configuran las variables iniciales y se importan las imágenes que se utilizaran. Las variables configuradas son:
    a. dir_name: path de las imágenes.
    b. Img*: los tamaños de la imagen, en este caso son de 400x400 px

c. Batch_size: en este caso es un valor bastante bajo debido a que la cantidad de imágenes es baja.
d. Epochs: la cantidad de épocas que se va a iterar, en este caso 30
e. Info: importación de las imágenes
f. className: los nombres de las imágenes (Rama negra, Yuyo colorado y Roseta)

```
dir_name = "../images/images"

img_height = 400
img_width = 400
batch_size = 8
epochs = 30

info = Data(dir_name, img_height=img_height, img_width=img_width)
(train_images, test_images) = info.load_data()
classNames = train_images.class_names
```

Anexo - Figura 2. Set de variables

3. Creación del modelo: es lo más interesante del código ya que en esta sección se crea el modelo que se va a entrenar posteriormente. Aquí no se explicará cada capa implementada puesto que fue explicada anteriormente en la sección de “El modelo de categorización”

```
#START CREACION DEL MODELO#

num_classes = len(classNames)
model = models.Sequential()

#Data augmentation
model.add(layers.experimental.preprocessing.RandomFlip("horizontal",
                                                       input_shape=(img_height,
                                                                    img_width,
                                                                    3)))
model.add(layers.experimental.preprocessing.RandomRotation(0.1))
model.add(layers.experimental.preprocessing.RandomZoom(0.1))

model.add(layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3))) #3 por RGB
model.add(layers.Conv2D(32, 3, padding='valid', activation='relu'))
model.add(layers.Conv2D(32, 3, padding='valid', activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Conv2D(64, 3, padding='valid', activation='relu'))
model.add(layers.Conv2D(64, 3, padding='valid', activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Conv2D(128, 3, padding='valid', activation='relu'))
model.add(layers.Conv2D(128, 3, padding='valid', activation='relu'))
model.add(layers.Conv2D(128, 3, padding='valid', activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Flatten())
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.35))
model.add(layers.Dense(num_classes, activation='softmax'))

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.summary()
#END CREACION DEL MODELO#
```

Anexo - Figura 3. Creación del modelo

4. Entrenamiento del modelo: Seguido del armado del modelo, se lo entrena en la línea 92 del código. Adicionalmente se agregó el guardado del mejor

modelo y de los logs. El primero sirve para guardar la mejor iteración del modelo y usarlo posteriormente para realizar pruebas. El segundo sirve para obtener de forma detallada el entrenamiento realizado y analizarlo.

```
datetime = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
log_dir = "logs/vgg16" + datetime
model_save_dir = '../models/best_models/modelcallback'+datetime+'.h5'
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)
model_callback = tf.keras.callbacks.ModelCheckpoint(
    model_save_dir,
    monitor='val_loss',
    verbose=1,
    save_best_only=False,
    save_weights_only=False,
    mode='auto',
    period=1)

history = model.fit(
    train_images,
    validation_data=test_images,
    epochs=epochs,
    callbacks=[tensorboard_callback, model_callback]
)
```

Anexo - Figura 4. Entrenamiento del modelo

5. Predecir una imagen: se carga el modelo guardado y haciendo uso de keras se predice el tipo de maleza de una imagen.

Firma Estudiante:	Firma Docente Supervisor:	Firma docente tutor TAPTA:	Firma tutor Organizacional:

```
model = tf.keras.models.load_model('../models/definitivos/vgg16-01.h5')
image_dir = '../images/test/ramanegra3.jpg'
img = keras.preprocessing.image.load_img(
    image_dir, target_size=(400,400)
)
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])
```

Anexo - Figura 5. Predecir una imagen