

Manual de Usuario Dataset.

Creación de Conjunto de datos.

Existe varias maneras de crear el conjunto de datos para realizar un entrenamiento ya sea para clasificación o detección de objetos, y esta maneja un término específico al que llamaremos “Etiqueta.” Las etiquetas son los nombres de los objetos a ser detectado, por tanto, existirá una etiqueta por cada objeto de estudio. Al hablar de conjunto de datos en el contexto de este proyecto nos referimos a las imágenes que ya cuentan con las etiquetas correspondiente de sus objetos. Este proyecto al buscar el modelo con más alto grado de precisión recurrió a dos métodos de entrenamiento ya explicado anteriormente por tanto tenemos dos diferentes maneras de etiquetar los datos, pero que son igualmente compatibles en ambas plataformas de entrenamiento.

Conjunto de datos para Google Cloud.

Al crear una cuenta de todo el conglomerado de soluciones que ofrece Google como la Gmail, nos abre camino para usar los demás productos, en este caso la solución que más se ajusta nuestras necesidades es la de Google Cloud, este trabaja en conjunto con otros servicios como la de Drive, Storage, Engine. Google Cloud es la plataforma principal de servicios en la nube, en donde se puede alojar aplicaciones, lo más importante es que la plataforma ofrece tres meses gratis y un saldo de 300 dólares para su uso.

DETECCIÓN DE OBJETOS AL GUARANÍ

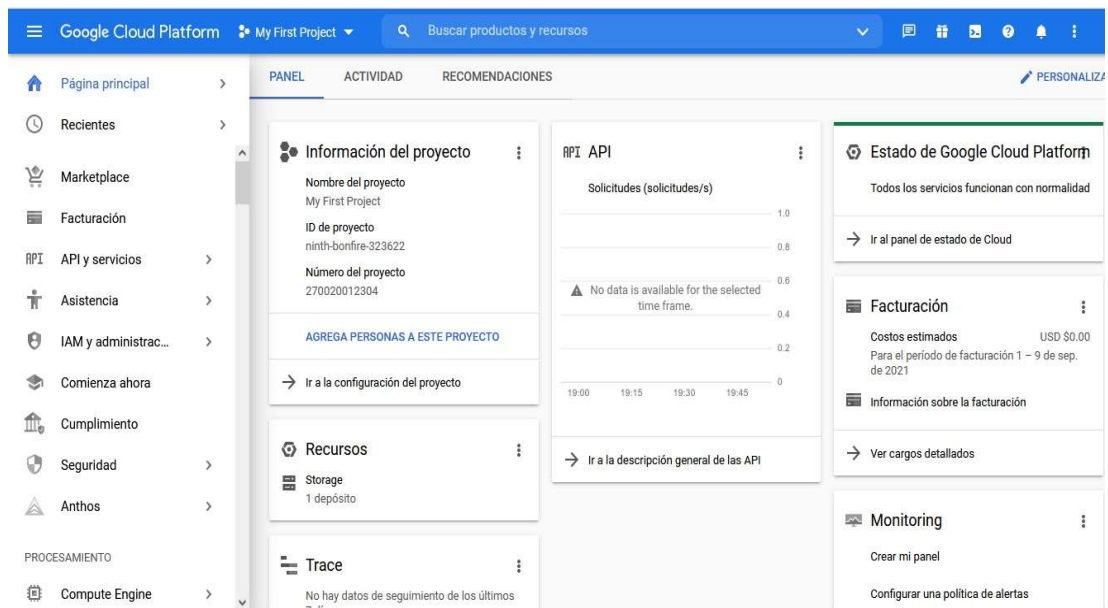


Figura. 2 Vista general Consola de Google Cloud Fuente: Google Cloud Plataform.

En la Figura 2 podemos ver la vista general que nos ofrece la plataforma para la administración de todos los recursos que está habilitada para usar en la cuenta, el panel incluye una barra de búsquedas para encontrar aplicativos que no están a la vista.

Una vez completada el proceso de logueo procederemos a buscar el aplicativo AutoML Vision.

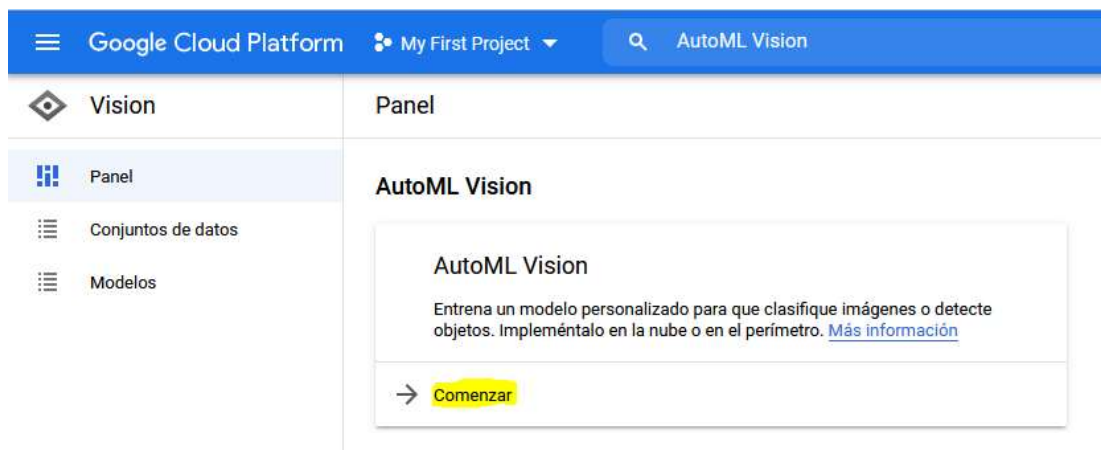


Figura 3. Resultado de búsquedas de aplicativos. Fuente: Google Cloud Plataform.

A continuación, podemos ver el resultado de nuestra búsqueda Figura 3, es esta sección pulsaremos la opción de comenzar, resaltada en amarillo.

DETECCIÓN DE OBJETOS AL GUARANÍ

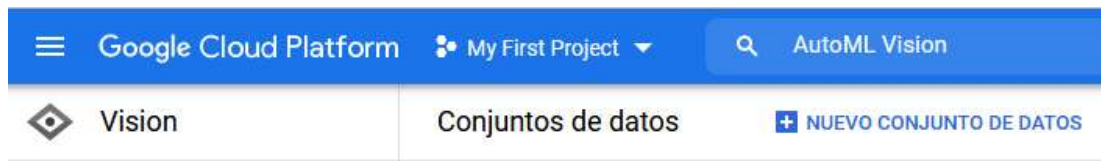


Figura 4 Panel de Creación de Datos Fuente: Google Cloud Plataform.

Seguidamente debemos elegir crear un nuevo conjunto de datos como se muestra en la Figura 4, debemos asignarle un nombre ya que esta será la carpeta que contendrá todas nuestras imágenes que deberán ser entrenadas.

Crear nuevo conjunto de datos

Nombre del conjunto de datos *

Test

Usa un máximo de 32 caracteres, que pueden incluir letras, números y guiones bajos.

Selecciona el objetivo de tu modelo

☐ Clasificación con una sola etiqueta

Realizar la predicción de la etiqueta correcta que deseas asignar a una imagen.

☐ Clasificación con varias etiquetas

Realizar la predicción de las etiquetas correctas que deseas asignar a una imagen.

☒ Detección de objetos

Realiza la predicción de las ubicaciones de los objetos en los que tienes interés.

CANCELAR CREAR CONJUNTO DE DATOS

Figura 5. Opciones de creación de conjunto de datos Fuente: Google Cloud Plataform.

La Figura 5 representa el punto medio del entrenamiento a través del método en la nube, después de asignar un nombre al conjunto de datos debemos elegir cuidadosamente el objetivo de nuestro modelo entrenar, la opción elegida deber estar acorde a los propósitos generales del proyecto y con sus objetivos. La opción es la de Detección de Objetos que nos permitirá realizar la predicción de las ubicaciones de los objetos de nuestro interés.

Es importante mencionar que este conjunto de datos solo servirá para propósitos de predicciones específicas de objetos debido a la manera en que serán etiquetadas cada una de las imágenes que conforma todo el conjunto de datos.

DETECCIÓN DE OBJETOS AL GUARANÍ

The screenshot shows the 'guaraniaobject' project page in Google Cloud AutoML Vision. At the top, there are navigation links: a back arrow, 'guaraniaobject', 'ESTADÍSTICAS DE ETIQUETAS' (with a bar chart icon), and 'EXPORTAR DATOS' (with an upload icon). Below these are tabs: 'IMPORTAR' (selected), 'IMÁGENES', 'ENTRENAR', 'EVALUAR', and 'PROBAR Y USAR'. The main heading is 'Selecciona archivos para importar'. A paragraph explains that AutoML Vision uses images to train a model and advises on labeling and image quantity. A bulleted list provides guidelines: each image must have a label, CSV files can be used for labels, and 100 images per label are recommended. Two radio buttons allow choosing between 'Subir imágenes desde tu computadora' (selected) and 'Seleccionar un archivo CSV en Cloud Storage'. Below this is the section 'Subir imágenes desde tu computadora', which lists supported formats (JPG, PNG, GIF, BMP, ICO) and a 500-file limit. A 'SELECCIONAR ARCHIVOS' button is present. At the bottom, a text input field shows 'gs:// Destino en Cloud Storage' with a 'BROWSE' button to its right.

← guaraniaobject ESTADÍSTICAS DE ETIQUETAS EXPORTAR DATOS

IMPORTAR IMÁGENES ENTRENAR EVALUAR PROBAR Y USAR

Selecciona archivos para importar

AutoML Vision usa tus imágenes para entrenar un modelo de aprendizaje automático personalizado. Antes de la importación, asegúrate de que tu conjunto de imágenes tenga las etiquetas adecuadas y contenga suficientes imágenes para el resultado que desees obtener.

- Cada cuadro de límite debe tener una etiqueta asignada.
- Puedes incluir los cuadros de límite y las etiquetas en tu archivo CSV o luego de la importación
- Se recomienda que proporciones 100 cuadros de límite por etiqueta

☒ Subir imágenes desde tu computadora

☐ Seleccionar un archivo CSV en Cloud Storage

Subir imágenes desde tu computadora

Admite JPG, PNG, GIF, BMP e ICO. Un máximo de 500 archivos por carga. Los archivos subidos se almacenarán en Cloud Storage.

SELECCIONAR ARCHIVOS

gs:// Destino en Cloud Storage BROWSE

Figura 6. Pantalla de para importar datos Fuente: Google Cloud Platform.

Antes del comienzo del etiquetado de datos debemos cargar todas las imágenes en la Cloud. En la Figura 6 vemos las opciones para importar el conjunto de datos, todos los datos importados serán almacenados en Cloud Storage, que este caso será nuestro repositorio de conjunto de datos.

Culminada la importación, debemos crear las etiquetas que serán asignadas a las imágenes, es importante mencionar que una sola imagen puede contener hasta diez etiquetas que pueden ser todas de un mismo objeto o diferentes, usando este método podemos llegar a reducir el tamaño del conjunto de datos y también requeriremos menos capacidad de computación al momento de realizar aprendizaje automático.

DETECCIÓN DE OBJETOS AL GUARANÍ

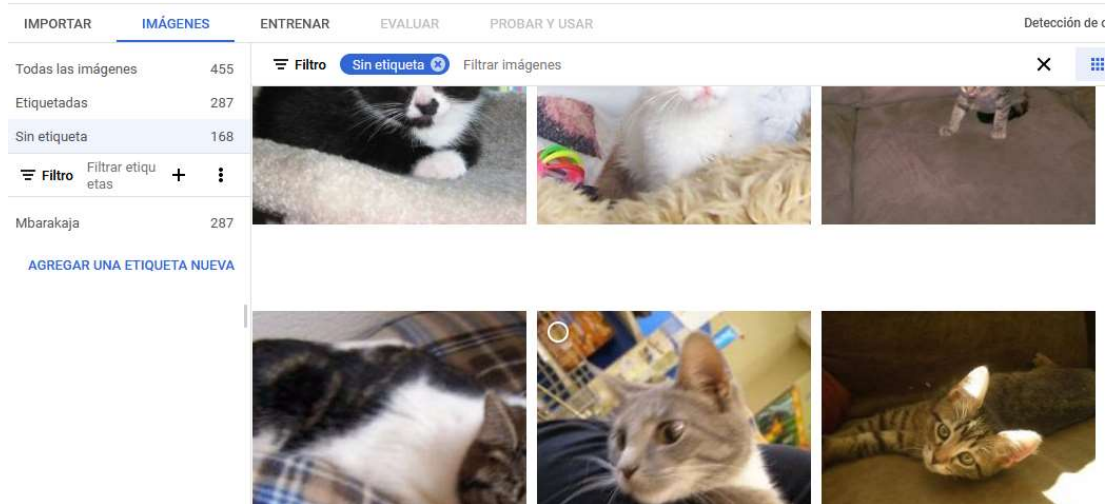


Figura 7. Vista del conjunto de datos Fuente: Google Cloud Plataform.

En la Figura 7 podemos apreciar la vista previa de los datos cargados, en la misma vista debemos cargar las etiquetas necesarias para el conjunto de datos.

Agregar una etiqueta nueva

The form consists of a text input field with a blue border. Above the field is a label 'Etiqueta'. Inside the field, the text 'Kyse' is entered. To the right of the input field are two buttons: 'CANCEL' and 'LISTO'.

Figura 8. Creación de nuevas etiquetas Fuente: Google Cloud Plataform.

Comenzaremos la asignación de las etiquetas, nos posicionamos sobre cualquier imagen de las vistas previas hacemos un click y automáticamente nos abrirá la imagen completa, al posicionar el cursor del mouse en cualquier ubicación sobre la imagen podremos marcar nuestro objeto en cuestión dentro del mismo, así como lo muestra la Figura 9.

DETECCIÓN DE OBJETOS AL GUARANÍ



Figura 9. Marcación de objetos Fuente: Google Cloud Platform.

Identificado y marcado nuestro objeto nos habilitará para asignarle un nombre. En margen izquierdo tendremos todas las opciones previamente cargadas.

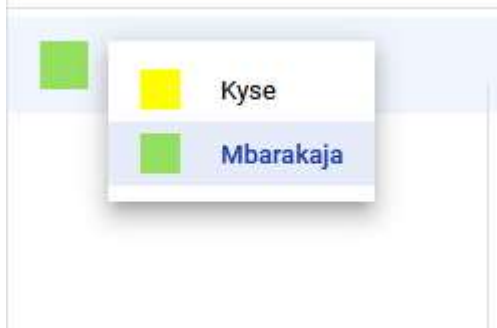


Figura 10. Listas de nombre de objetos cargados Fuente: Google Cloud Platform.

Al seleccionar uno de los nombres, esta será asignada al recuadro dentro de la imagen, de esta manera completamos el proceso que llamamos etiquetado de datos.

DETECCIÓN DE OBJETOS AL GUARANÍ

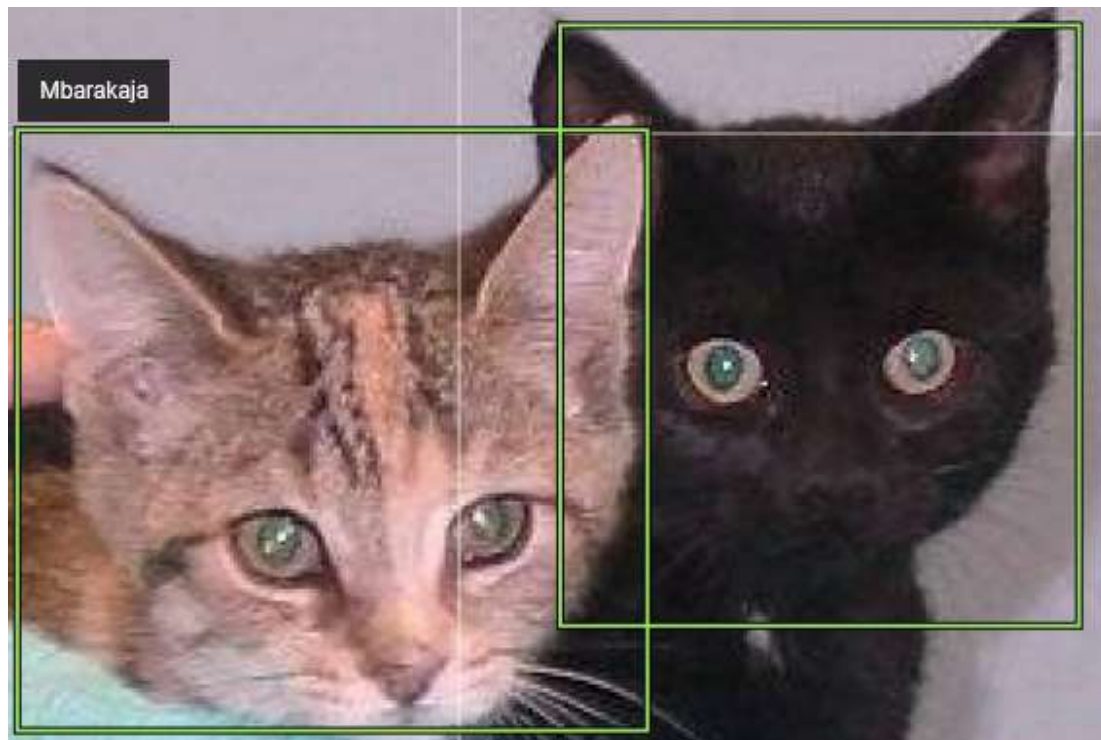


Figura 11. Vista de etiquetado completo Fuente: Google Cloud Plataform.

Podemos observar cómo queda la imagen después de realizar el proceso de etiquetado, como lo muestra la Figura 11, además podemos notar que una sola imagen tenemos dos regiones de interés.

De esa manera se va generando todo el conjunto de datos, esto conlleva muchísimo tiempo, al designar cada nombre de etiqueta por cada imagen, para el desarrollo de este proyecto se realizó el etiquetado de aproximadamente cinco mil imágenes.

Formato de anotaciones.

Las anotaciones en este contenido del trabajo hacen referencia a las etiquetas que fueron asignadas al conjunto de datos, el archivo que será exportada en formato de texto o CSV, contiene los siguientes parámetros.

TRAIN: imágenes para entrenamiento.

VALIDATION: imágenes de prueba antes de terminar entrenamiento.

TEST: imágenes para evaluación de modelo final.

PATH: ubicación del conjunto de datos.

NAME: nombre del conjunto de datos.

DETECCIÓN DE OBJETOS AL GUARANÍ

TAGS: coordenadas de las regiones de interés de la imagen, son los recuadros asignados a cada objeto dentro de la imagen.

```
TRAIN:/content/images/dato1.jpg;Ryguasu;0.17196262;0.40356082;0.34579438;0.40356082;0.34579438;0.7448071;0.17196262;0.7448071
TRAIN:/content/images/dato2.jpg;Ryguasu;0.17196262;0.40356082;0.34579438;0.40356082;0.34579438;0.7448071;0.17196262;0.7448071
TRAIN:/content/images/dato3.jpg;Ryguasu;0.50666666;0.69436204;0.98222222;0.69436204;0.98222222;0.9525223;0.50666666;0.9525223
```

Figura 12. Vista de archivo de entrenamiento Fuente: Elaboración Propia.

Manual de Usuario entrenamiento Google Cloud.

Una vez etiquetado todos los datos podemos pasar directamente a la sección de entrenamiento de Google Cloud, es importante resaltar que esta actividad será realizada en modo gratuito, por lo tanto, las horas de entrenamiento estarán sujetas a este plan.

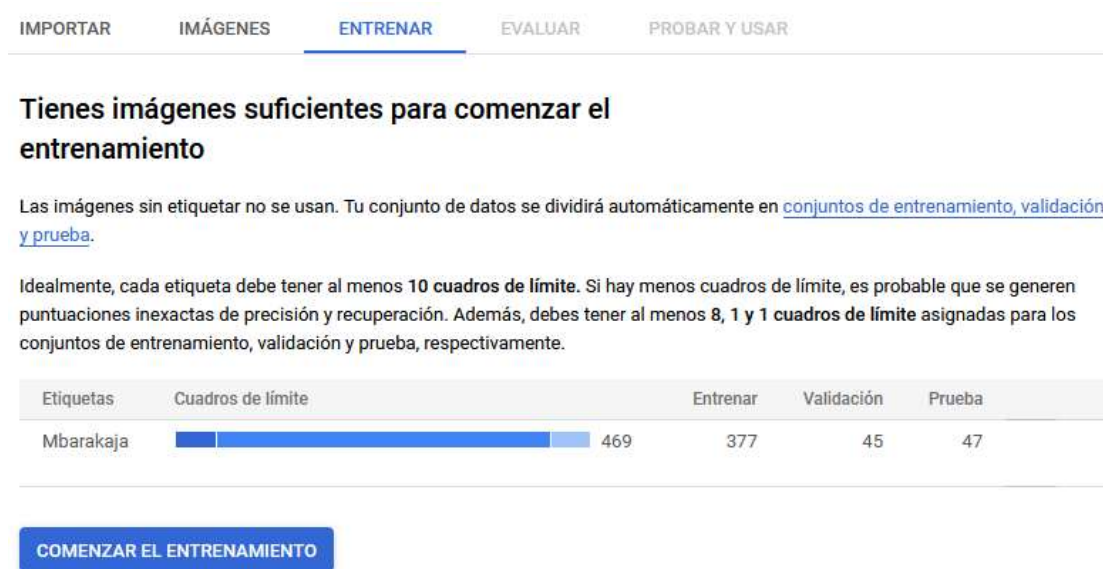
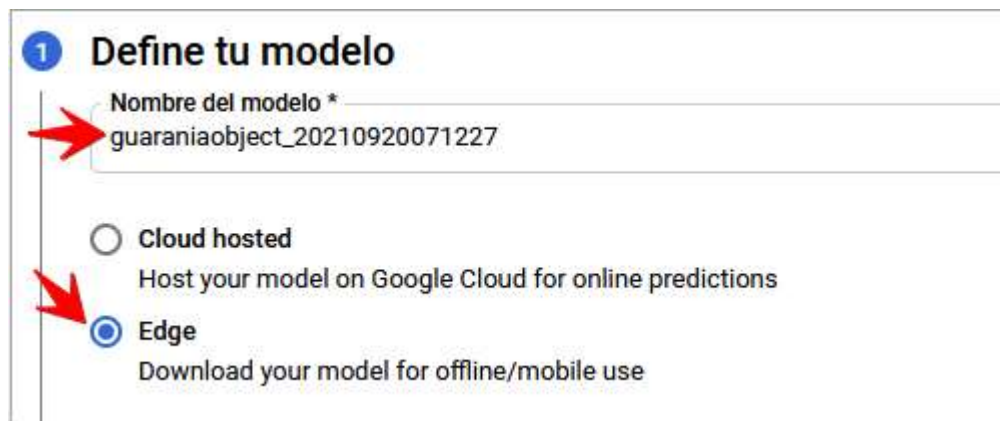


Figura 13. Vista del conjunto antes de entrenamiento Fuente: Google Cloud Plataforma.

En la Figura 13 podemos apreciar la vista de la pestaña Entrenar, en donde la plataforma no indica las etiquetas que existentes, cantidad de imágenes que lo conformó y las divisiones para cada etapa del entrenamiento, las agrupaciones son generadas en forma automáticas por la plataforma en donde aplica una métrica de 80% del conjunto para entrenamiento, 10% para validación y 10% para pruebas.

Al ingresar a la opción de comenzar entrenamiento nos desplegará una nueva sección, aquí debemos de asignarle un nombre al modelo como se muestra en la Figura 14.

DETECCIÓN DE OBJETOS AL GUARANÍ



1 Define tu modelo

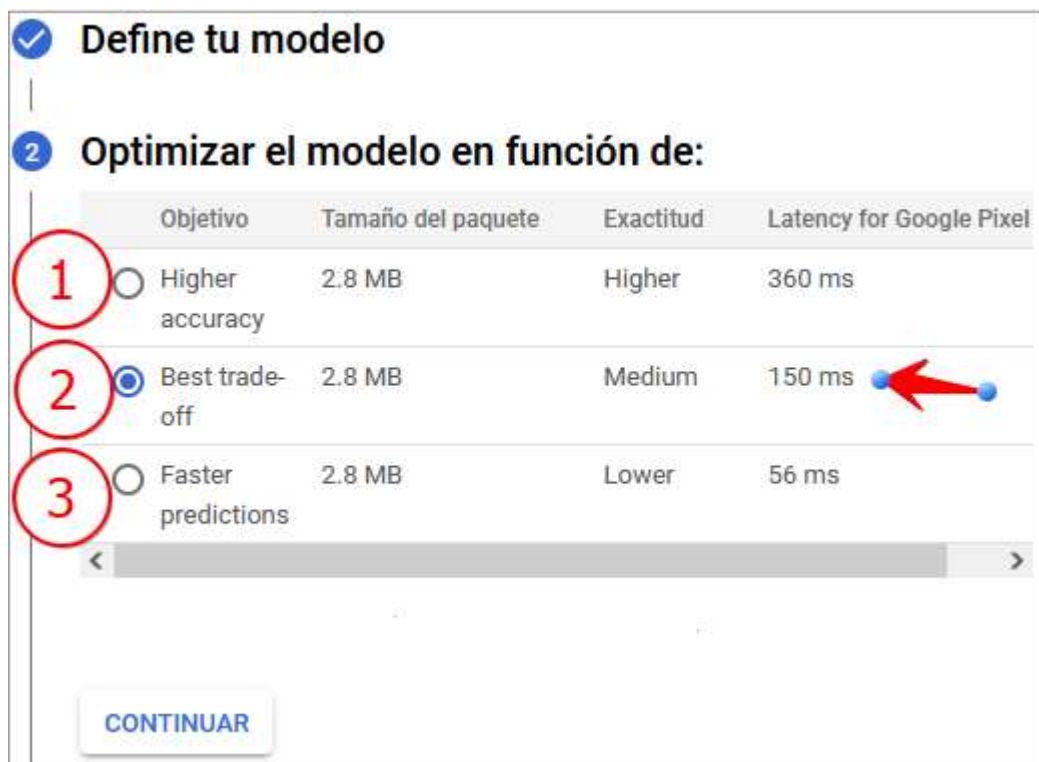
Nombre del modelo *
guaraniaobject_20210920071227

☐ Cloud hosted
Host your model on Google Cloud for online predictions

☒ Edge
Download your model for offline/mobile use

Figura 14. Definición del nombre del modelo y método de entrenamiento. Fuente: Google Cloud Platform.

En la misma sección debemos de elegir también el método de entrenamiento, para las finalidades prácticas de este proyecto usaremos la opción de Edge que se encuentra indicada en la Figura 14 la elección de esta opción nos permitirá implementar el modelo resultante en dispositivos móviles para realizar una detección fuera de línea, es decir, el modelo predictivo recurrirá a los recursos de hardware del dispositivo móvil para realizar las predicciones que están entrenadas en el modelo implementado.



✓ Define tu modelo

2 Optimizar el modelo en función de:

	Objetivo	Tamaño del paquete	Exactitud	Latency for Google Pixel
1	<input type="radio"/> Higher accuracy	2.8 MB	Higher	360 ms
2	<input checked="" type="radio"/> Best trade-off	2.8 MB	Medium	150 ms
3	<input type="radio"/> Faster predictions	2.8 MB	Lower	56 ms

< >

CONTINUAR

DETECCIÓN DE OBJETOS AL GUARANÍ

Figura 15. Elegir grado de exactitud Fuente: Google Cloud Plataform

En la Figura 15 podemos notar las diferentes opciones de exactitud que está disponible actualmente, en donde todo relacionado con rapidez de procesamiento y exactitud de predicción.

La opción uno en la misma gráfica vemos que la prioridad es la de realizar una detección con un alto grado de fiabilidad, en contrapartida esto requiere mayor uso de las capacidades de procesamiento del dispositivo y por lo tanto la visualización de los resultados se procesará con una mayor latencia.

La segunda opción que es método elegido para este proyecto, se presenta con una latencia media para mostrar los resultados, los cuales presentan una certeza del 50% en promedio con la capacidad del modelo entrenado.

La tercera usa la premisa de realizar una rápida detección del cualquier objeto dentro del modelo, con una baja tasa de uso de las capacidades del hardware, sin embargo, recurrir a esto significa sacrificar la exactitud de las predicciones.

✓ Define tu modelo

✓ Optimizar el modelo en función de:

3 Set a node hour budget

Ingresa el número máximo de horas de procesamiento de nodo que quieres dedicar al entrenamiento de tu modelo.

We recommend using [24 node hours](#) for your dataset. However, you can train for as little as 1 node hours. You may also eligible to train with free node hours. [Guía de precios](#)

Establece tu presupuesto * 24 node hours

Estimated completion date: sep. 21, 2021 9 p.m. GMT-4

COMENZAR EL ENTRENAMIENTO CANCELAR

Figura 16. Definir horas de entrenamiento Fuente: Google Cloud Plataform.

Como último paso del método de entrenamiento en la nube de Google, dependiendo de la cantidad de imágenes que conforma el conjunto de datos la herramienta nos recomendará las horas de entrenamiento necesarias para lograr un modelo optimizado de acuerdo a las previas elecciones hechas, está también estará ligada a un costo si supera las horas que incluye el vale otorgado en un principio. Definido todos los parámetros procedemos a realizar el entrenamiento según indica la Figura 16, la plataforma se encarga de la notificación cuando concluyan las pruebas, para poder descargar el modelo generado y empezar a utilizarla.

Manual para entrenamiento en maquina local.

El principal requisito para realizar este entrenamiento es que la máquina debe contar con un acelerador gráfico de última generación compatibles con las herramientas de desarrollo CUDA (Developer, Nvidia, 2021), los dispositivos que cumplen estos requisitos están preparados para soportar entrenamiento de aprendizaje profundo y aprendizaje de máquinas.

Esta sección del manual será explicada de manera muy detallada, para ajustarte a las capacidades técnicas de personas que quieran colaborar con este proyecto a futuro, o para usarla como base para nuevas ideas de desarrollo.

Para dar un inicio rápido a este manual el entorno del sistema operativo usado será la Microsoft Windows 10 en cualquiera de sus variantes, pero con una capacidad de 64bits, esto debido en que la cantidad de memoria de acceso aleatoria necesario para la extracción de características únicas de cada objeto es alta, como también las cantidades de cálculos a ser realizados requieren mucha potencia y ciclos del procesador.

Todos los pasos que serán descritos a continuación serán válidos para equipos que cumplan con las especificaciones y que tengan instalado una distribución del ambiente de Linux.

Entorno de Entrenamiento.

Para evitar entrar en conflictos con programas actualmente instalados en el equipo local recurriremos al uso de un software especialmente diseñado para estos casos, este aplicativo usa un método de trabajo que se denomina: entorno, podremos crear varios entornos, y cada una de ellas pueden tener diferentes paquetes de softwares instalados y estas no entraran en conflicto con las otras.

El primer paso será descargar la herramienta, la cual podemos obtener de manera gratuita de la página oficial: <https://www.anaconda.com/products/individual>

Con la descarga completa procederemos a realizar la instalación.

En la barra de buscadores debemos buscar el aplicativo llamado: Anaconda Prompt, al ejecutarlo nos abrirá una ventana de línea de comandos.

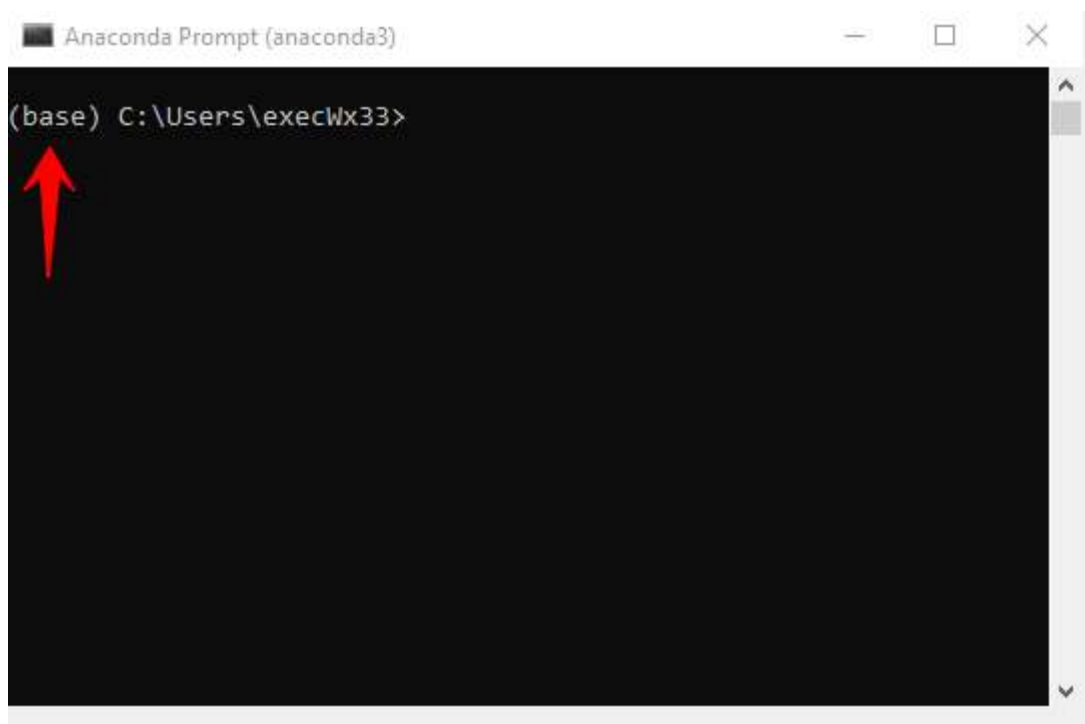


Figura 17. Inicio de consola Anaconda. Fuente: Elaboración Propia.

DETECCIÓN DE OBJETOS AL GUARANÍ

En la Figura 17 podemos notar que en la consola antecede la palabra base dentro de paréntesis, este indicativo nos muestra que se instaló la herramienta de manera correcta y ya esta lista para la creación de un ambiente de trabajo.

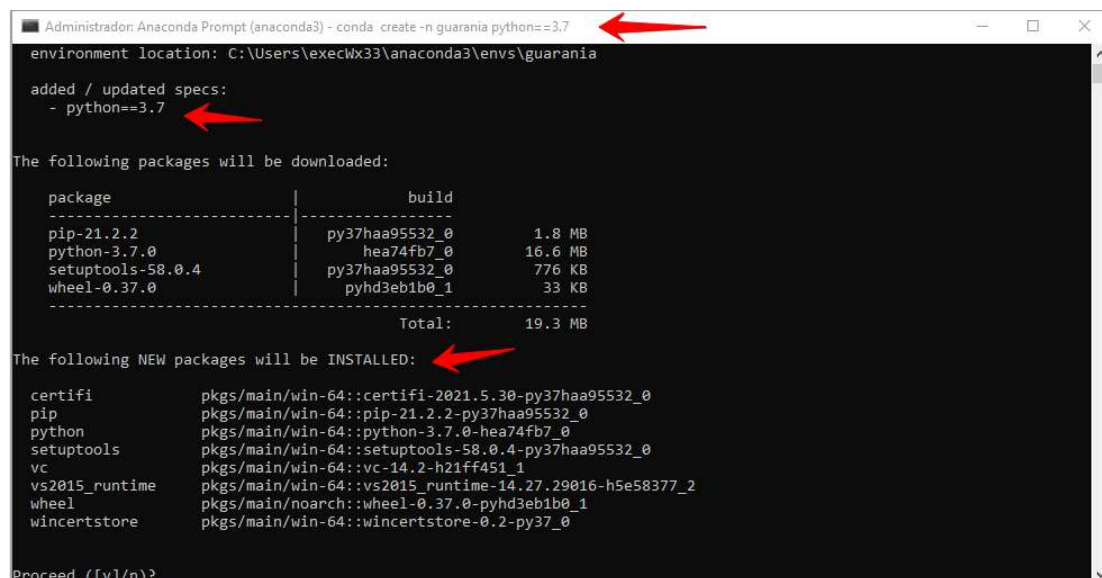
En la consola abierta procederemos a ejecutar comandos para lograr crear un entorno personalizado, para este trabajo solo veremos aquellos que nos ayuden al desarrollo del entrenamiento, en caso de querer profundizar más sobre ella recomendamos consultar las documentaciones disponibles en página web del editor.

Nos reubicamos de vuelta en el apartado del Gráfica 27 en donde ejecutaremos el siguiente comando: `conda create -n guarania python==3.7`

`conda create` : llamada de comando para creación de un nuevo entorno de trabajo

`-n guarania`: asignamos un nombre para identificar el entorno que debemos crear, es este caso el asignado es “guarania”.

`python==3.7`: con esta sentencia indicamos la versión de Python que será utilizado en el entorno de manera predeterminada, es decir cualquier paquete que necesitamos instalar será descargada con la versión compatible con esta versión del lenguaje, esta versión es elegida debido que actualmente cuenta con las bibliotecas de repositorios más amplias, estables y con la mayor documentación necesarias para resolver cualquier inconveniente que puede surgir.



```
Administrator: Anaconda Prompt (anaconda3) - conda create -n guarania python==3.7
environment location: C:\Users\execWx33\anaconda3\envs\guarania

added / updated specs:
- python==3.7

The following packages will be downloaded:

package | build | size
-----|-----|-----
pip-21.2.2 | py37haa95532_0 | 1.8 MB
python-3.7.0 | hea74fb7_0 | 16.6 MB
setuptools-58.0.4 | py37haa95532_0 | 776 KB
wheel-0.37.0 | pyhd3eb1b0_1 | 33 KB
-----|-----|-----
Total: | 19.3 MB

The following NEW packages will be INSTALLED:

certifi | pkgs/main/win-64::certifi-2021.5.30-py37haa95532_0
pip | pkgs/main/win-64::pip-21.2.2-py37haa95532_0
python | pkgs/main/win-64::python-3.7.0-hea74fb7_0
setuptools | pkgs/main/win-64::setuptools-58.0.4-py37haa95532_0
vc | pkgs/main/win-64::vc-14.2-h21ff451_1
vs2015_runtime | pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2
wheel | pkgs/main/noarch::wheel-0.37.0-pyhd3eb1b0_1
wincertstore | pkgs/main/win-64::wincertstore-0.2-py37_0

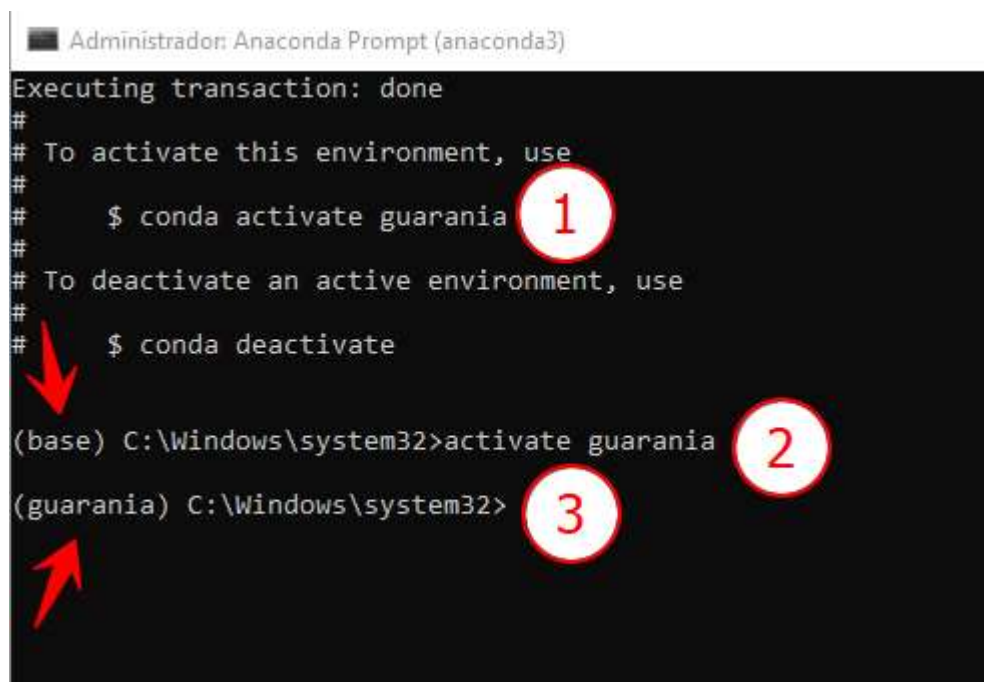
Proceed ([y]/n)?
```

Figura 18. Creación de entorno de trabajo. Fuente: Elaboración Propia.

DETECCIÓN DE OBJETOS AL GUARANÍ

En la Figura 18 podemos ver las salidas que se generan al ingresar los comando para la creación del entorno, como podemos apreciar con los indicadores resaltados en el gráfico, la ejecución descargará los paquetes de intérpretes necesarios para que el entorno funcione de manera individual sin afectar al sistema operativo huésped.

Al momento de terminar las instalaciones la consola de línea de comando arrojará un mensaje con las instrucciones para activar el entorno creado.



```
Administrador: Anaconda Prompt (anaconda3)
Executing transaction: done
#
# To activate this environment, use
#
#   $ conda activate guarania 1
#
# To deactivate an active environment, use
#
#   $ conda deactivate
#
(base) C:\Windows\system32>activate guarania 2
(guarania) C:\Windows\system32> 3
```

The screenshot shows a Windows command prompt window titled 'Administrador: Anaconda Prompt (anaconda3)'. The output shows the successful execution of a transaction and instructions for activating and deactivating the 'guarania' environment. Three red circles with numbers 1, 2, and 3 are overlaid on the text. Circle 1 highlights the command '\$ conda activate guarania'. Circle 2 highlights the command '(base) C:\Windows\system32>activate guarania'. Circle 3 highlights the prompt '(guarania) C:\Windows\system32>'. Red arrows point to the prompt changes: one from the first instruction to the second prompt, and another from the second instruction to the third prompt.

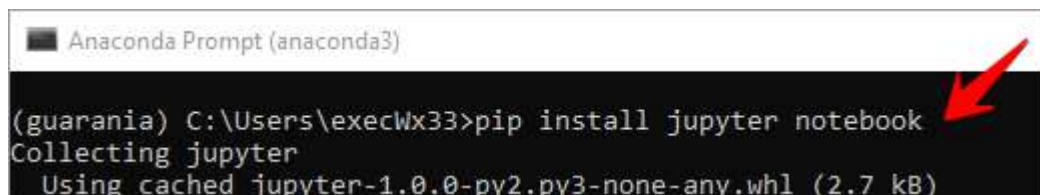
Figura 19. Activación de entorno de trabajo. Fuente: Elaboración Propia.

En el indicador uno de la Figura 19 podemos apreciar los comandos para la activación del entorno del trabajo, el segundo indicador podemos ver que estamos ubicados en la raíz de la instalación que es la base al ingresar la sentencia descrita en el punto número uno, podemos ver el cambio y se queda con el entorno que habíamos creado (guarania) tal y como podemos ver el indicador número tres.

DETECCIÓN DE OBJETOS AL GUARANÍ

En este punto podemos empezar a utilizar las bondades de esta herramienta ya como se mencionó anteriormente podemos instalar todos programas que necesitemos para poder trabajar sin que esto afecte a otros entornos creados, al sistema operativo ya sea de Windows o cualquier distro de Linux.

La primera herramienta a instalar será Jupyter Notebook, esta nos permitirá editar los códigos de manera interpretada junto con Python para poder realizar los entrenamientos.



```
Anaconda Prompt (anaconda3)

(guarania) C:\Users\execWx33>pip install jupyter notebook
Collecting jupyter
Using cached jupyter-1.0.0-py2.py3-none-any.whl (2.7 kB)
```

Figura 20. Comando de instalación de editor de códigos. Fuente: Elaboración Propia.

La Figura 20 nos muestra las indicaciones de la instalación del aplicativo, esta al ejecutarse descargara sus propios paquetes para poder iniciarse sin inconveniente.

El uso de esta es muy sencilla, para empezar siempre debemos estar posicionados en el entorno adecuados, y ejecutar lo siguiente jupyter notebook , con esto realizará un llamado al navegador de internet predeterminado para ejecutarse .

La configuración de este proyecto incluye la carpeta con todos los datos necesarios para poder realizar el entrenamiento, para lo cual antes de iniciar el editor de códigos debemos de posicionarnos en la ruta en donde se encuentra dichos archivos.

En el ejemplo utilizaremos la unidad principal del sistema operativo, por lo cual copiaremos la carpeta del proyecto en la Unidad C:, la carpeta puede ser descargada directamente desde el repositorio creado para este proyecto desde el siguiente enlace.

<https://github.com/GuaraniDL/guarani-ia.git>

DETECCIÓN DE OBJETOS AL GUARANÍ

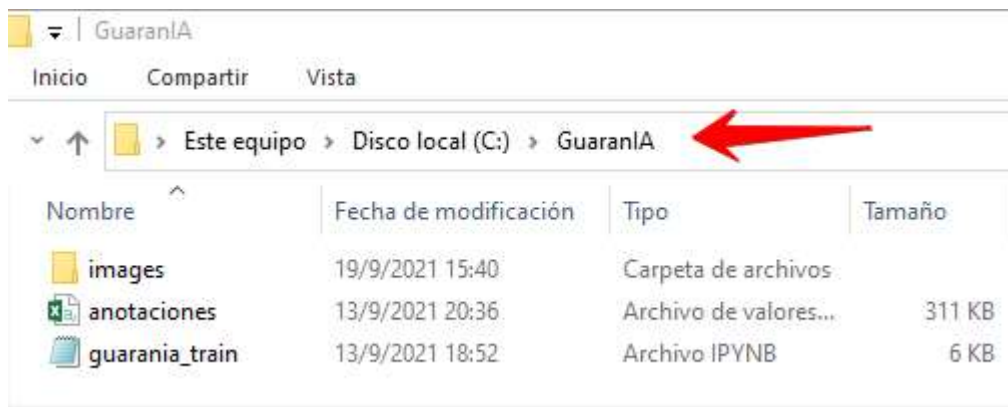


Figura 21. Vista de la carpeta del proyecto. Fuente: Elaboración Propia.

El contenido de la carpeta según la Figura 21 lo conforman:

- images: contiene todas las imágenes que serán utilizadas durante la fase de entrenamiento.

- anotaciones: este archivo en formato CSV separadas por comas, contiene las ubicaciones de las imágenes seguido de la región de interés de cada una de ellas.

Una aclaración importante en este punto, en caso de que la carpeta del proyecto se aloje en otra ubicación, debemos de editar el archivo anotaciones para cambiar las rutas de las imágenes.

Ejemplo 1:

TRAIN,C:\GuaranIA\images\gallina234-2021-08-14T16_58_15.552Z.jpg

Ejemplo 2:

TRAIN,C:\Documentos\GuaranIA\images\gallina234-2021-08-14T16_58_15.552Z.jpg

En el ejemplo 1, podemos apreciar la ubicación de las imágenes que usamos para este contexto, en el ejemplo 2 podemos ver que la carpeta del proyecto está en otra ubicación por tanto se cambia la ruta de acceso a las imágenes en el archivo de anotaciones.

Para el cambio masivo de las rutas podemos usar la función reemplazar de la herramienta Notepad++ atendiendo que son casi tres mil registros a cambiar.

No obstante, otro punto que aclarar suponiendo que estemos en un sistema operativo Linux es la orientación de las divisiones entre las rutas de las carpetas dentro del documento de anotaciones.

DETECCIÓN DE OBJETOS AL GUARANÍ

Ejemplo 2:

TRAIN,C:\GuaranIA\images\gallina234-2021-08-14T16_58_15.552Z.jpg

Ejemplo 3:

TRAIN,/GuaranIA/images/gallina234-2021-08-14T16_58_15.552Z.jpg

En ejemplo 2 podemos ver que la ubicaciones de las carpetas se separaran por símbolo de la barra invertida (\), y justo antes de la ubicación de los archivos que se requiere acentar se usa dos repeticiones seguidas, esto es así solo para los entornos de Microsoft Windows, en cambio para Linux es más sencilla como vemos en el ejemplo 3 , solo requiere del símbolo de la división (/) para realizar la lectura de las imágenes y tampoco es necesario agregar dos secuencias seguidas del símbolo antes de la ubicación del archivo.

guarania_train.ipynb: este último archivo contiene todos los pasos y parámetros y códigos fuentes de para realizar el entrenamientos, está en un formato válido para que pueda ser ejecutado por Jupyter Notebook.

Iniciaremos el proceso de entrenamiento:



```
Anaconda Prompt (anaconda3) jupyter notebook guarania_train.ipynb
(guarania) C:\Users\exel\WX33>cd ..
(guarania) C:\Users>cd..
(guarania) C:\>cd GuaranIA
(guarania) C:\GuaranIA>jupyter notebook guarania_train.ipynb
[I 14:13:13.185 NotebookApp] Serving notebooks from local directory: C:\GuaranIA
[I 14:13:13.185 NotebookApp] Jupyter Notebook 6.4.4 is running at:
[I 14:13:13.185 NotebookApp] http://localhost:8888/?token=31a2e2febad98a84fba0708588076df2f49e0e88f56d2607
[I 14:13:13.185 NotebookApp] or http://127.0.0.1:8888/?token=31a2e2febad98a84fba0708588076df2f49e0e88f56d2607
[I 14:13:13.186 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 14:13:13.324 NotebookApp]
```

Figura 22 Inicio de archivo fuente del entrenamiento . Fuente: Elaboración Propia.

Los indicadores en la Figura 22, podemos ver primeramente que estamos ubicados en el entorno correcto, en el segundo paso nos ubicamos en la carpeta de nuestro proyecto y en el último paso procedemos a ejecutar Jupyter Notebook con el archivo que contiene las fuentes del proyecto de forma inmediata esta invocará una serie de proceso que culminará con la apertura del navegador principal por defecto con el contenido del archivo de códigos fuentes, para ejecutar están instancia como lo indica el paso número tres escribimos lo siguiente en la consola de nuestro entorno: `jupyter notebook guarania_train.ipynb`.

En esta sección explicaremos el uso Jupyter Notebook, para ejecutar una sentencia debemos ubicarnos con el cursor en la celda correspondiente.

DETECCIÓN DE OBJETOS AL GUARANÍ

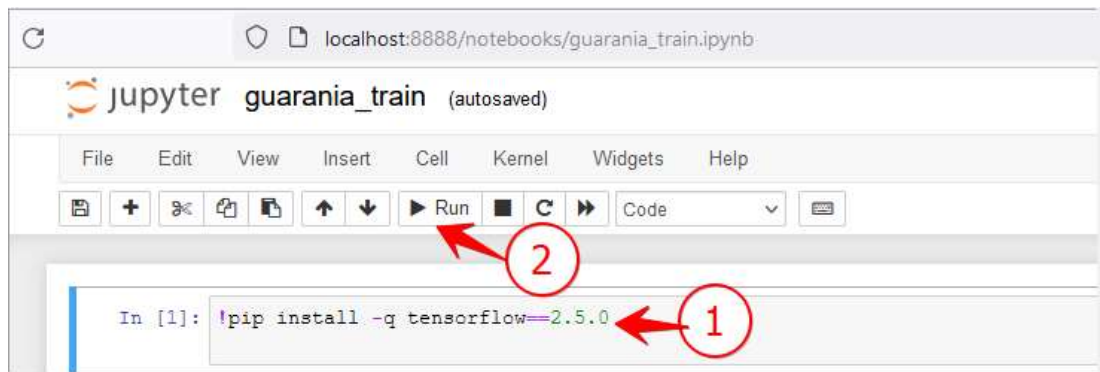


Figura 23. Vista de opciones de ejecución de Jupyter Notebook. Fuente: Elaboración Propia.

Como nos muestra la Figura 23 en el paso número uno, luego aplicamos siguiente paso de ejecutar la orden como nos indica el gráfico, también podemos realizar la ejecución de una sentencia oprimiendo la combinación de las teclas CTRL+ENTER, en la parte superior del paso numero dos podemos apreciar todas las operaciones de ejecución que tenemos disponibles.

Al momento de la ejecución entre los corchetes deberá aparecer un símbolo de asterisco como se ve en la Figura 24, esto es indicativo de que se está realizando acciones tras la ejecución.

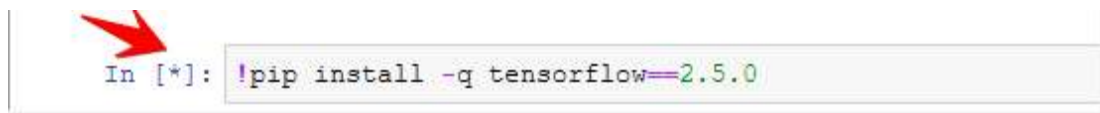


Figura 24. Vista de ejecución de sentencia Fuente: Elaboración Propia.

Cuando todos los procesos desencadenados por la ejecución de la celda culminen, esta quedará como lo muestra la Figura 25, esta vez el distintivo será que entre los corchetes deberá aparecer un número, esta hace referencia a que culmino la ejecución y que fue la primera orden en ser ejecutada en el entorno.

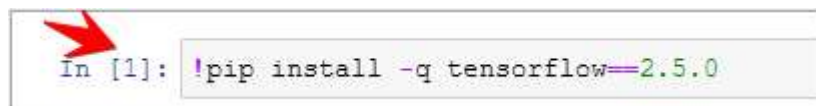
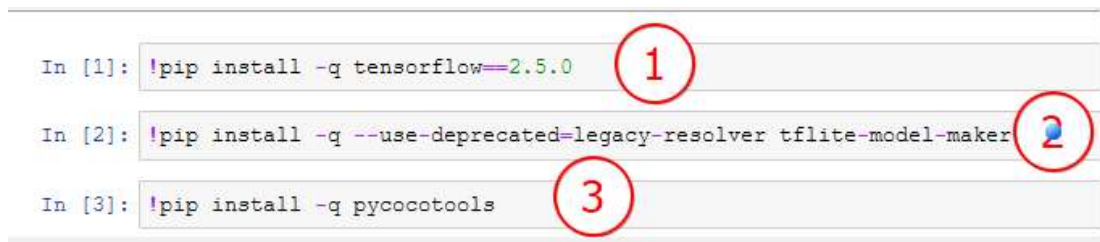


Figura 25. Vista de ejecución terminada. Fuente: Elaboración Propia.

Instalación de librerías

DETECCIÓN DE OBJETOS AL GUARANÍ



```
In [1]: !pip install -q tensorflow==2.5.0
In [2]: !pip install -q --use-deprecated=legacy-resolver tflite-model-maker
In [3]: !pip install -q pycocotools
```

Figura 26. Instalación de librería de ejecución. Fuente: Elaboración Propia.

El paso número uno de la Figura 26 se instala la herramienta principal que se encargada de la extracción de las características de la imagen, luego se descargará las librerías que nos permitirá leer todo el conjunto de datos, y nos permitirá exportar modelo generado para los usos en nuestras instancias, en tercer paso instalaremos la biblioteca que nos permitirá medir el porcentaje de la exactitud del modelo que entrenaremos.

```
|
import numpy as np
import os

from tflite_model_maker.config import ExportFormat
from tflite_model_maker import model_spec
from tflite_model_maker import object_detector

import tensorflow as tf
assert tf.__version__.startswith('2')

tf.get_logger().setLevel('ERROR')
from absl import logging
logging.set_verbosity(logging.ERROR)
```

Figura 27. Importación de controladores Fuente: elaboración Propia.

La importación de objetos y controladores se realiza después de la instalación de las librerías, Figura 27 podemos observar el llamado a la función de la exportación del formato del modelo, también vemos la importación de conjunto que nos permitirá elegir la categoría de entrenamiento y la opción de detección de objetos, y lo más importante el llamado a función principal de TensorFlow.

El llamado de todas estas características nos permitirá declarar y ajustar todas nuestras variables para modelar el entrenamiento.

DETECCIÓN DE OBJETOS AL GUARANÍ

Primera Variable

La primera declaración de variables dentro de los objetivos del proyecto será la de especificar con qué algoritmo de arquitectura de extracción de características será entrenado el modelo, para el efecto podemos ver el en la Tabla 6.

Para lograr el objetivo del proyecto, el conjunto de datos será entrenado en todos los modelos disponibles con las mismas configuraciones de las épocas, para luego comparar los resultados, en la Figura 29 podemos observar la declaración de ejemplo de unas de las arquitecturas de entrenamiento.

```
spec = model_spec.get('efficientdet_lite4')
```

Figura 28. Definición del modelo de arquitectura para entrenamiento. Fuente: Elaboración Propia.

Segunda Variable.

La segunda variable consta de la parametrización del conjunto de datos en el recuadro de la Figura 29 definimos la estructura de procesamiento de los datos que son tres: validación, entrenamiento, prueba, seguidamente hacemos un llamado al conjunto del controlador previamente invocado cuya tarea será la de leer las imágenes del conjunto de datos a través del archivo de anotaciones para el efecto le pasamos como parámetros la ubicación del archivo. En este punto se comprobará de que todas las imágenes existan dentro del conjunto de datos, devolverá un error en caso de que una imagen no se encuentre disponible en el conjunto.

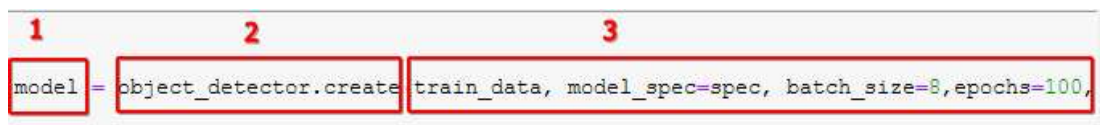
```
train_data, validation_data, test_data = object_detector.DataLoader.from_csv('C:\GuaranIA\anotaciones.csv')
```

Figura 29. Parametrización del conjunto de datos Fuente: Elaboración Propia.

Tercera Variable.

Esta variable contiene sentencias y parámetros bastantes extensos por lo que procederemos a explicarlas brevemente, es aquí donde definimos las instancias del entrenamiento para generar el modelo que estamos buscando.

DETECCIÓN DE OBJETOS AL GUARANÍ

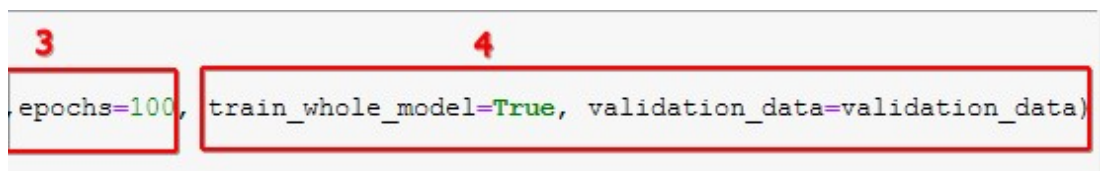


```
1 2 3
model = object_detector.create(train_data, model_spec=spec, batch_size=8, epochs=100,
```

Figura 30. Vista principal de variables de entrenamiento. Fuente: Elaboración Propia.

El primer contenido de la tercera variable es la declaración del modelo como se ve en la Figura 30 en su punto número uno está su vez invoca a la función para la creación de objetos en su paso número dos, esta función en efecto tiene sus propios parámetros en donde llama al conjunto de datos para entrenamiento, invoca al modelo de arquitectura definido previamente, define el tamaño de datos que enviaremos a la CPU/GPU para el entrenamiento, para este trabajo utilizamos un tamaño de 8bytes de datos, con esto se logra que el modelo empiece a converger con todas las neuronas y capas ocultas logrando aprender los rasgos necesarios sin necesidad de ver todo el conjunto de datos.

Las arquitecturas de modelos EfficientDet por defecto esta predefinida para entrenar 50 épocas, como notamos en nuestro caso utilizaremos 100 épocas para entrenar los datos, las épocas hacen referencia a la cantidad de veces que el conjunto de datos será analizado para la extracción de las características.

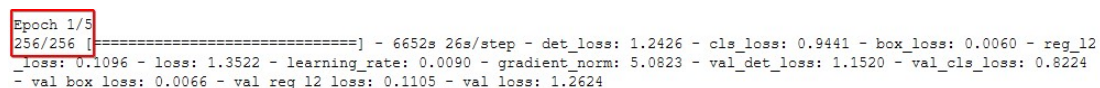


```
3 4
epochs=100, train_whole_model=True, validation_data=validation_data)
```

Figura 31. Vista final de las variables de entrenamiento. Fuente: Elaboración Propia.

Los últimos parámetros de la tercera variable, indicamos al intérprete de Python que entrene el modelo completo; es decir, solo culminará el proceso una vez que se haya completado las 100 épocas de entrenamiento, como último parámetro definimos la validación que también forma parte del conjunto de datos.

Esta parte de la ejecución tarda 18 horas.



```
Epoch 1/5
256/256 [=====] - 6652s 26s/step - det_loss: 1.2426 - cls_loss: 0.9441 - box_loss: 0.0060 - reg_l2
_loss: 0.1096 - loss: 1.3522 - learning_rate: 0.0090 - gradient_norm: 5.0823 - val_det_loss: 1.1520 - val_cls_loss: 0.8224
- val box loss: 0.0066 - val reg l2 loss: 0.1105 - val loss: 1.2624
```

Figura 32. Detalles de entrenamiento. Fuente: Elaboración Propia.

DETECCIÓN DE OBJETOS AL GUARANÍ

La duración del entrenamiento es extensa, esto es debido a que en cada época todo el conjunto de datos deberá ser procesada varias veces, en la Figura 32 podemos observar los detalles del proceso de entrenamiento, la parte resaltada se ve la cantidad de épocas definidas y más abajo la cantidad de pasos (256/256/) necesarios para leer todo el conjunto de datos en una sola época, esta segmentación se realiza a partir del tamaño de datos definidos que son de 8bytes, y divide el conjunto de datos en 256 lotes, para enviarlas a la arquitectura elegida para su procesamiento.

Cuando las lecturas de las épocas culminen procederemos a evaluar el modelo generado. Para realizar esto ejecutamos la sintaxis que vemos en la Figura 33.

Esta etapa del proyecto realiza una evaluación del modelo generado para medir la precisión en la detección de cada objeto y lo realiza comparando con las imágenes que están el conjunto de datos con las etiquetas de prueba, están imágenes son nuevas para el modelo y no formaron parte del entrenamiento, por lo que es importante que estos datos sean únicos y no se encuentren repetidas dentro del conjunto de validación y entrenamiento.

```
model.evaluate(test_data)
```

Figura 33. Evaluación del modelo generado. Fuente: Elaboración Propia.

Los detalles que devuelve esta ejecución podemos apreciarlo en la Figura 34, en donde la parte resaltada muestra los objetos entrenados, con la precisión obtenida usando los datos de pruebas.

DETECCIÓN DE OBJETOS AL GUARANÍ

```
{ 'AP': 0.47524908,  
  'AP50': 0.7578863,  
  'AP75': 0.51540583,  
  'APs': 0.2977908,  
  'APm': 0.36507475,  
  'APl': 0.5377998,  
  'ARmax1': 0.5132024,  
  'ARmax10': 0.649214,  
  'ARmax100': 0.65816087,  
  'ARs': 0.68333334,  
  'ARm': 0.5726726,  
  'ARl': 0.6962452,  
  'AP_/Ryguasu': 0.4268511,  
  'AP_/Mbarakaja': 0.23436964,  
  'AP_/Jagua': 0.37182269,  
  'AP_/Pumbyry': 0.6070896,  
  'AP_/Kavaju': 0.4771644,  
  'AP_/Jetapa': 0.5156747,  
  'AP_/Apyka': 0.65829366,  
  'AP_/Anguja': 0.3465513,  
  'AP_/Kuimbe': 0.5749284,  
  'AP_/Kyse': 0.53974533}
```

Figura 34. Resultados de la evaluación. Fuente: Elaboración Propia.

Exportación de Modulo.

La parte final del entrenamiento es donde generamos el modelo en un archivo con su formato característico, para esto recurriremos al uso de sentencia que vemos en la Figura 35, esta instancia empaquetará el modelo con las etiquetas de cada objeto con sus pesos, los pesos son los binarios que contiene la información para detectar el objetivo en una imagen o en el entorno.

```
model.export(export_dir='.')
```

Figura 35. Comandos para exportación de modelo. Fuente: Elaboración Propia

El modelo será guardado en la carpeta donde el entorno está ubicado ver Figura 21.

DETECCIÓN DE OBJETOS AL GUARANÍ

En la Figura 36 ya vemos el modelo que se exportó como “model.tflite”, también está resaltado la carpeta donde se generó el archivo.

Con esto ya estamos listos para implementarlo en la aplicación por defecto puesto a disposición por Google.

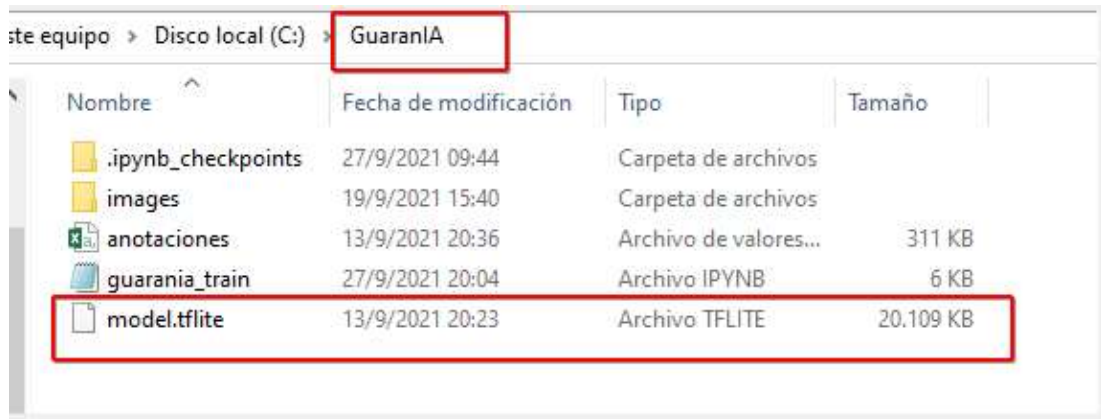


Figura 36. Vista de generación de archivos. Fuente: Elaboración Propia.

Implementación del Modelo.

La última fase consiste en la implementación de modelo, por lo que empezaremos con la instalación de Android Studio.

Luego clonamos el repositorio de TensorFlow, con la cual utilizaremos la aplicación prediseñada para implementar el modelo generado.

Para este punto ejecutamos Android Studio, realizamos la importación del proyecto, debemos esperar a que actualicen las librerías correspondientes.

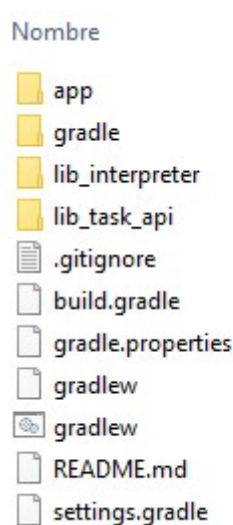


Figura 37. Estructura de archivos de la aplicación. Fuente: Elaboración Propia.

DETECCIÓN DE OBJETOS AL GUARANÍ

La Figura 37 nos muestra las estructuras de los archivos del proyecto de una aplicación destinado para el sistema operativo Android.

Seguidamente compilaremos aplicación en este proceso se descarga desde Google un modelo de detección de objetos de ejemplo, esa se ubicará en una carpeta específica. Identificamos la carpeta y reemplacemos el modelo por el generado por nuestro entrenamiento.

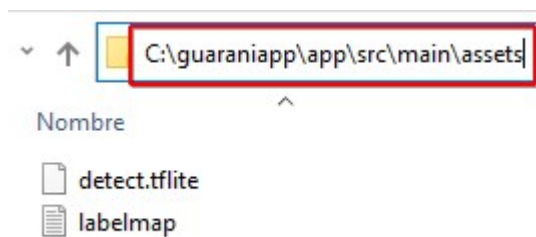


Figura 38. Vista de carpeta de modelos. Fuente: Elaboración Propia.

En la Figura 38, vemos en el recuadro resaltado la ubicación donde se descarga los modelos de prueba que por defecto debe llamarse “detect. tflite”, esto está definido así en la documentación proveída por la empresa Google, ya que esta misma aplicación es usada para implementar modelos de clasificación de imagen entre otros; por tanto, debemos reemplazar nuestro modelo usando el mismo nombre.

El segundo archivo de texto dentro de la carpeta llamada “labelmap” también debemos reemplazarla, su contenido está constituido por las etiquetas que se usaron al momento de crear el conjunto de datos para el entrenamiento.

DETECCIÓN DE OBJETOS AL GUARANÍ



Figura 39. Vista de contenido de archivo de etiquetas. Fuente: Elaboración Propia.

Como podemos notar en la Figura 39, están escritas todos los nombres en guaraní del objeto que forman parte de nuestro estudio, este archivo será usado al momento de ejecutar la aplicación mientras esté realizando las predicciones basados en nuestro modelo, cuando se encuentre las coincidencias necesarias serán mostrados los nombres de los objetos basados en este archivo de texto. Es importante aclarar que dentro del modelo binario del entrenamiento también se encuentran estos mismos nombres por lo que este archivo sirve como validación de las etiquetas que serán mostradas en la pantalla de la aplicación.

Cuando todos los archivos necesarios ya se han copiado procederemos a conectar el dispositivo con a la computadora. Android Studio detectará este nuevo equipo, gracias a los controladores que tienen instalados.

Un método sencillo de comprobar estos es fijarnos en el menú de las herramientas de Android Studio.

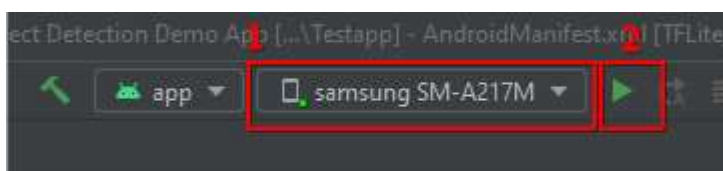


Figura 40. Dispositivos conectados al entorno de Android Studio. Fuente: Elaboración Propia.

DETECCIÓN DE OBJETOS AL GUARANÍ

La Figura 40 nos muestra los detalles de los dispositivos conectados en el recuadro número uno el siguiente objeto resaltado es la del icono de compilación, procederemos a ejecutarla de vuelta, con esto lograremos que nuestro modelo personalizado se cargue en la aplicación y al tener ya conectado nuestro aparato celular esta se instalará de manera automática en ella.

A continuación, veremos algunas capturas de la aplicación en uso.



Como podemos notar en las imágenes las detecciones se realizan asignando una región de interés con el porcentaje certeza que el modelo detecta, esto por supuesto usa como medida todos los conocimientos que se fueron adquiriendo a través de las épocas de entrenamiento.

Estos resultados son variables teniendo en cuenta el entorno en donde realizamos las detecciones, donde son vinculantes la calidad de enfoque de la cámara, apertura de focos, iluminación del ambiente, capacidades de procesamiento del dispositivo móvil y principalmente la calidad promedia de detecciones del modelo, que varían según la versión de arquitectura entrenada.