

# Desarrollo de proyectos de IA

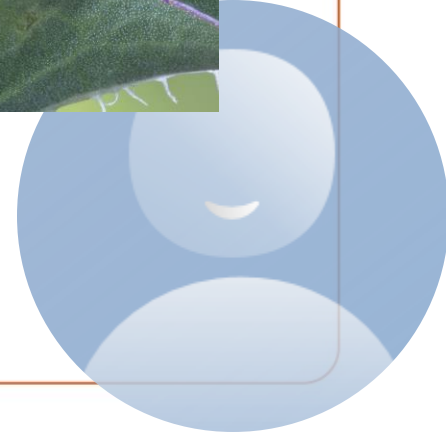
## Proyecto final: Clasificación de insectos

Alejandro Valencia	2244055
Guillermo Umaña Ramírez	2236193
Camilo Farina	2244202



# ¿Cuál es el problema?

Actualmente la crisis ambiental requiere acciones inmediatas para generar conciencia sobre las especies que están en peligro, entre ellas la diversidad de insectos que tenemos en el valle del cauca.





# Equipo de trabajo



## **Business Analyst – Guillermo Umaña**

Su trabajo es entender a fondo el problema de negocio y definir claramente los objetivos que los proyectos de machine learning deben alcanzar.

## **Data Scientist – Guillermo Umaña**

Experto en la interpretación y manejo de datos complejos. Su trabajo implica descubrir patrones, realizar análisis predictivos y prescriptivos, y comunicar descubrimientos clave a través de visualizaciones y reportes, apoyando así la toma de decisiones estratégicas

## **MLOps – Camilo Farina:**

Se centra en todo el ciclo de vida de la implementación y gestión de modelos de aprendizaje automático en producción. Son los responsables de construir, automatizar y monitorizar las canalizaciones que llevan los modelos de aprendizaje automático desde el desarrollo hasta la producción.

## **Data Analyst– Camilo Farina:**

El Analista de Datos examina grandes conjuntos de datos para identificar tendencias, desarrollar gráficos y crear reportes visuales.

## **Desarrollador web full stack (FrontEnd y Backend) - Alejandro Valencia**

Tiene capacidad y experiencia para manejar tanto el front-end como el back-end de aplicaciones web. Esto significa que puede trabajar en el diseño y desarrollo de interfaces de usuario (front-end), creando una experiencia atractiva y funcional para el usuario final, así como también manejar los aspectos del servidor y la base de datos (back-end) para asegurarse de que la lógica de la aplicación, el manejo de datos y la integración de sistemas funcionen sin problemas.

## **MLEngineer – Alejandro Valencia**

Se centra en la construcción, implementación y mantenimiento de modelos de aprendizaje automático. Son los encargados de llevar los prototipos desarrollados por los científicos de datos a un entorno de producción real.

# Publico Objetivo



- Instituciones educativas (escuelas, universidades)
- Organizaciones ambientales y de conservación
- Entomólogos aficionados y entusiastas de la naturaleza
- Investigadores y científicos en el campo de la entomología ciudadana.



# Descripción de su solución

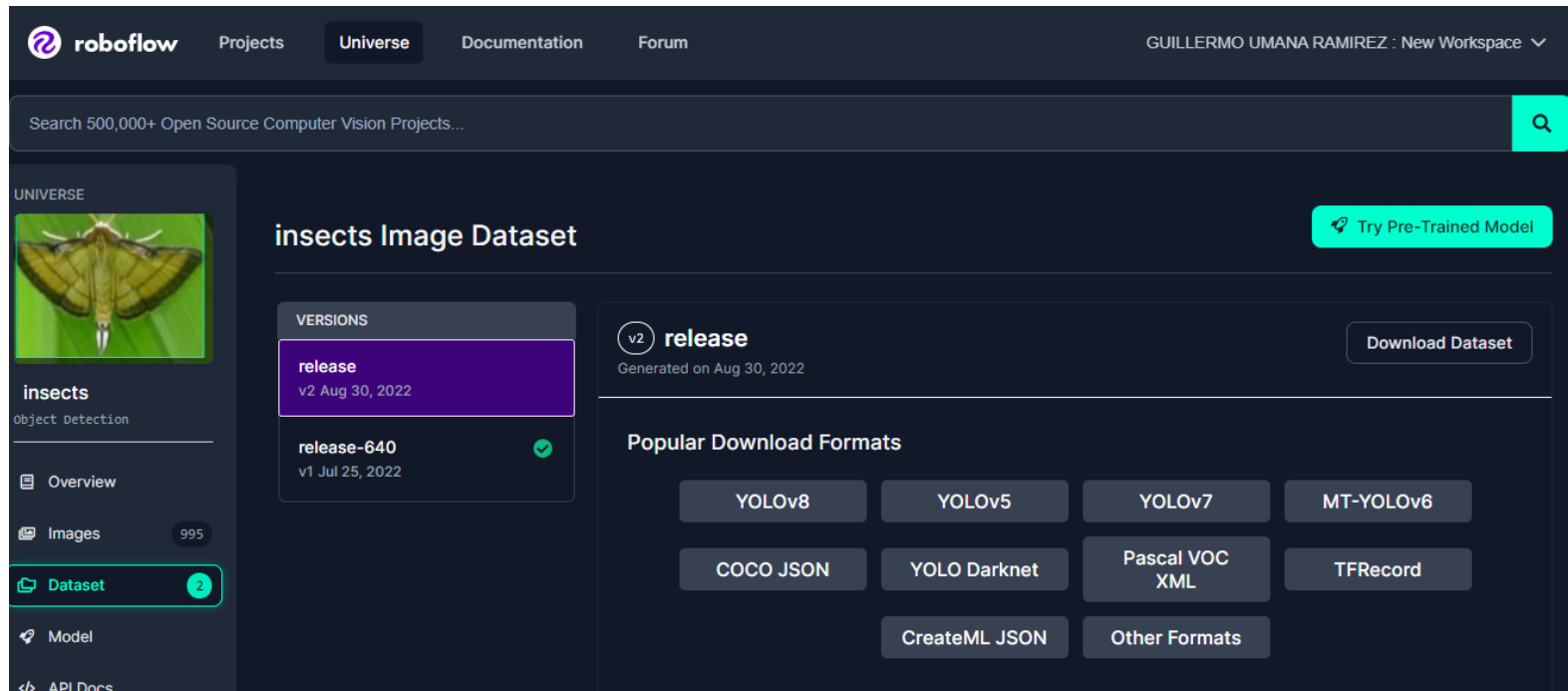


Desarrollamos una aplicación web amigable con el apoyo de inteligencia artificial (IA) para identificar con precisión y proporcionar información sobre diversos insectos a partir de fotografías tomadas o cargadas por los usuarios.

La aplicación tiene como objetivo que los usuarios conozcan información acerca de diversos insectos y como trasfondo busca promover la conciencia sobre la biodiversidad, asistir en la investigación científica y comprometer a los usuarios en proyectos de ciencia ciudadana.

# Dataset

Desarrollamos el modelo con el conjunto de datos "[Insects Image Dataset](#)" de Roboflow



The screenshot displays the Roboflow interface for the 'Insects Image Dataset'. The top navigation bar includes the Roboflow logo, 'Projects', 'Universe', 'Documentation', and 'Forum'. A search bar is present with the text 'Search 500,000+ Open Source Computer Vision Projects...'. The user's name 'GUILLERMO UMANA RAMIREZ' and 'New Workspace' are shown in the top right.

The main content area is titled 'insects Image Dataset'. It features a 'Try Pre-Trained Model' button. Below this, the 'VERSIONS' section lists two versions: 'release' (v2 Aug 30, 2022) and 'release-640' (v1 Jul 25, 2022). A 'Download Dataset' button is located next to the 'release' version.

The 'Popular Download Formats' section offers various export options: YOLOv8, YOLOv5, YOLOv7, MT-YOLOv6, COCO JSON, YOLO Darknet, Pascal VOC XML, TFRecord, CreateML JSON, and Other Formats.

The left sidebar provides navigation options: 'Overview', 'Images' (995), 'Dataset' (2), 'Model', and 'API Docs'.

# Dataset

995 Total Images

[View All Images →](#)



## CLASSES

army worm legume  
blister beetle red  
spider rice gall midge  
rice leaf roller rice  
leafhopper rice water  
weevil wheat  
phloeothrips white  
backed plant hopper  
yellow rice borer

## Dataset Split

TRAIN SET

70%

696 Images

VALID SET

20%

199 Images

TEST SET

10%

100 Images





# Modelo 1 MobileNetV2

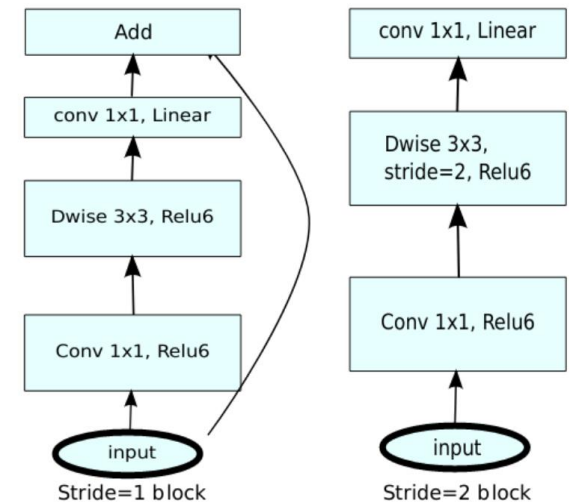
MobileNetV2 es una arquitectura de CNN eficiente en términos computacionales diseñada para dispositivos móviles y embebidos. Fue introducido por Google y es la sucesión de MobileNetV1. Se centra en la eficiencia de la velocidad y el tamaño al tiempo que mantiene una precisión razonable.

Bloques residuales de cuello de botella: Cada uno de estos bloques tiene varias capas:

- Capa de expansión (Conv 1x1, ReLU6): Usa convoluciones 1x1 para expandir el número de canales, agregando más características a los datos.
- Convolución de profundidad (Dwise 3x3, ReLU6): Aplica filtros de manera separada a cada canal de entrada (convolución de profundidad) y luego activa las características utilizando ReLU6.
- Capa de proyección (Conv 1x1, Linear): Convolución 1x1 que reduce el número de canales, proyectando las características obtenidas.
- Bloques con stride=1 no cambian el tamaño de la imagen.
- Bloques con stride=2 reducen el tamaño de la imagen a la mitad para enfocarse en las características más relevantes.

•Bloque final:

- Otra capa convolucional con un tamaño de filtro de 1x1 seguida de una activación ReLU6.
- Capa de promediado global (Global Average Pooling) que reduce cada mapa de características a un solo valor promediando todos los valores.
- Capa densa (Fully Connected) con una activación softmax que se utiliza para la clasificación final.





# Modelo 2 VGG16

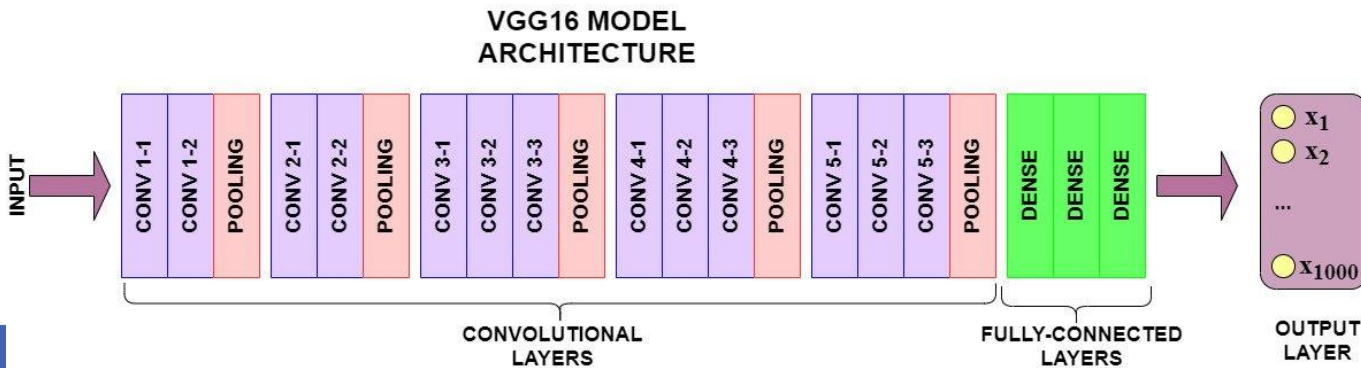
VGG16 es una arquitectura CNN que fue uno de los primeros enfoques que demostró que las redes más profundas pueden conducir a mejores resultados. Fue presentado por el Visual Graphics Group de la Universidad de Oxford, de ahí su nombre VGG.

La red VGG16 está dividida en bloques. Cada bloque tiene 2-3 capas convolucionales seguidas de una capa de max pooling. Después de 5 bloques convolucionales, la red aplana los mapas de características y los pasa a través de tres capas densas (una de ellas es la capa de salida) con ReLU como función de activación para las dos primeras capas densas y softmax para la capa de salida.

Bloques convolucionales: Cada bloque tiene un conjunto de capas convolucionales, cada una seguida de una activación ReLU.

Capas de max pooling: Después de cada bloque, se aplica max pooling para reducir la resolución espacial.

Capas densas: Al final de la red, tres capas densas concluyen la arquitectura, con la última capa densa que sirve como capa de salida con la activación softmax para la clasificación multiclase.



# Modelo 2 VGG16

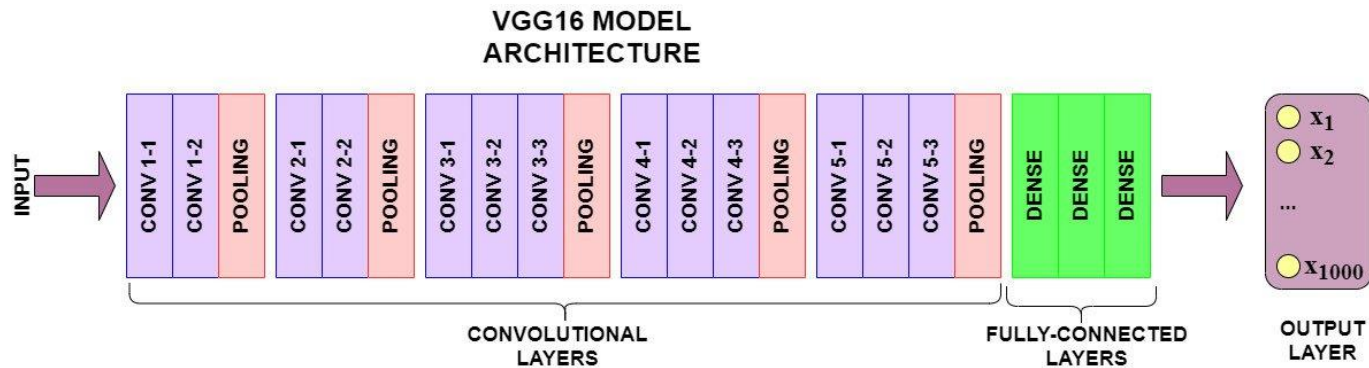
VGG16 es una arquitectura CNN que fue uno de los primeros enfoques que demostró que las redes más profundas pueden conducir a mejores resultados. Fue presentado por el Visual Graphics Group de la Universidad de Oxford, de ahí su nombre VGG.

La red VGG16 está dividida en bloques. Cada bloque tiene 2-3 capas convolucionales seguidas de una capa de max pooling. Después de 5 bloques convolucionales, la red aplanar los mapas de características y los pasa a través de tres capas densas (una de ellas es la capa de salida) con ReLU como función de activación para las dos primeras capas densas y softmax para la capa de salida.

**Bloques convolucionales:** Cada bloque tiene un conjunto de capas convolucionales, cada una seguida de una activación ReLU.

**Capas de max pooling:** Después de cada bloque, se aplica max pooling para reducir la resolución espacial.

**Capas densas:** Al final de la red, tres capas densas concluyen la arquitectura, con la última capa densa que sirve como capa de salida con la activación softmax para la clasificación multiclase.



# Modelo3 Arquitectura propia

- **Capas conv2d:** Detectan características visuales variadas desde bordes simples hasta texturas más complejas.
- **Batch normalization:** Estabiliza el aprendizaje al normalizar las entradas de las capas.
- **MaxPooling2D:** Reduce la resolución espacial, enfatizando las características más prominentes.
- **Dropout:** Previene el sobreajuste al "desactivar" aleatoriamente algunas conexiones neuronales.
- **Capas conv2d adicionales:** Capturan patrones más complejos con un número creciente de filtros.
- **Batch normalization:** Continúa la normalización para cada nuevo conjunto de mapas de características.
- **MaxPooling2D:** Aplica otra reducción de tamaño para simplificar la información de la imagen.
- **Dropout:** Incrementa la robustez del modelo con otra ronda de desconexiones aleatorias.
- **Flatten:** Convierte los mapas de características tridimensionales en un vector plano.
- **Dense (Capas densas):** Realizan la clasificación final basándose en las características aprendidas por la red.

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
batch_normalization (Batch Normalization)	(None, 148, 148, 32)	128
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
dropout (Dropout)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
batch_normalization_1 (Batch Normalization)	(None, 72, 72, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
dropout_1 (Dropout)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
batch_normalization_2 (Batch Normalization)	(None, 34, 34, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
dropout_2 (Dropout)	(None, 17, 17, 128)	0
flatten (Flatten)	(None, 36992)	0
dense (Dense)	(None, 512)	18940416
batch_normalization_3 (Batch Normalization)	(None, 512)	2048
dropout_3 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 9)	4617

```
=====  
Total params: 19041225 (72.64 MB)  
Trainable params: 19039753 (72.63 MB)  
Non-trainable params: 1472 (5.75 KB)
```

# Métricas para evaluar los modelos



## **Test Loss (Pérdida de prueba):**

Es una medida de qué tan bien el modelo está haciendo predicciones en comparación con las etiquetas verdaderas en el conjunto de datos de prueba. Una pérdida más baja indica un mejor rendimiento, con una pérdida de cero que significaría predicciones perfectas.

## **Test Accuracy (Exactitud de prueba):**

Mide el porcentaje de predicciones correctas del modelo sobre el total de predicciones realizadas. Una precisión del 100% significaría que el modelo predijo correctamente cada instancia en el conjunto de datos de prueba.

## **Precision (Precisión):**

Indica qué tan precisas son las predicciones positivas del modelo. Es decir, de todas las instancias que el modelo clasificó como positivas, cuántas son realmente positivas.

## **Recall (Exhaustividad):**

Mide la capacidad del modelo para encontrar todas las instancias positivas reales en los datos. Es decir, de todas las instancias que son realmente positivas, cuántas identificó correctamente el modelo.

## **F1-Score:**

Combina la precisión y el recall en una sola métrica mediante la media armónica de ambos. El F1-Score da una idea del balance entre la precisión y el recall, y es particularmente útil con clases desbalanceadas. Un F1-Score perfecto es 1, mientras que un modelo completamente inefectivo tendría un F1-Score de 0.

## **Confusion Matrix (Matriz de confusión):**

Permite visualizar el rendimiento de un modelo de clasificación. En una tabla, muestra la frecuencia con la que las predicciones del modelo coinciden o difieren de las etiquetas reales. En otras palabras, nos indica cuántas veces el modelo ha clasificado correctamente o incorrectamente cada clase.

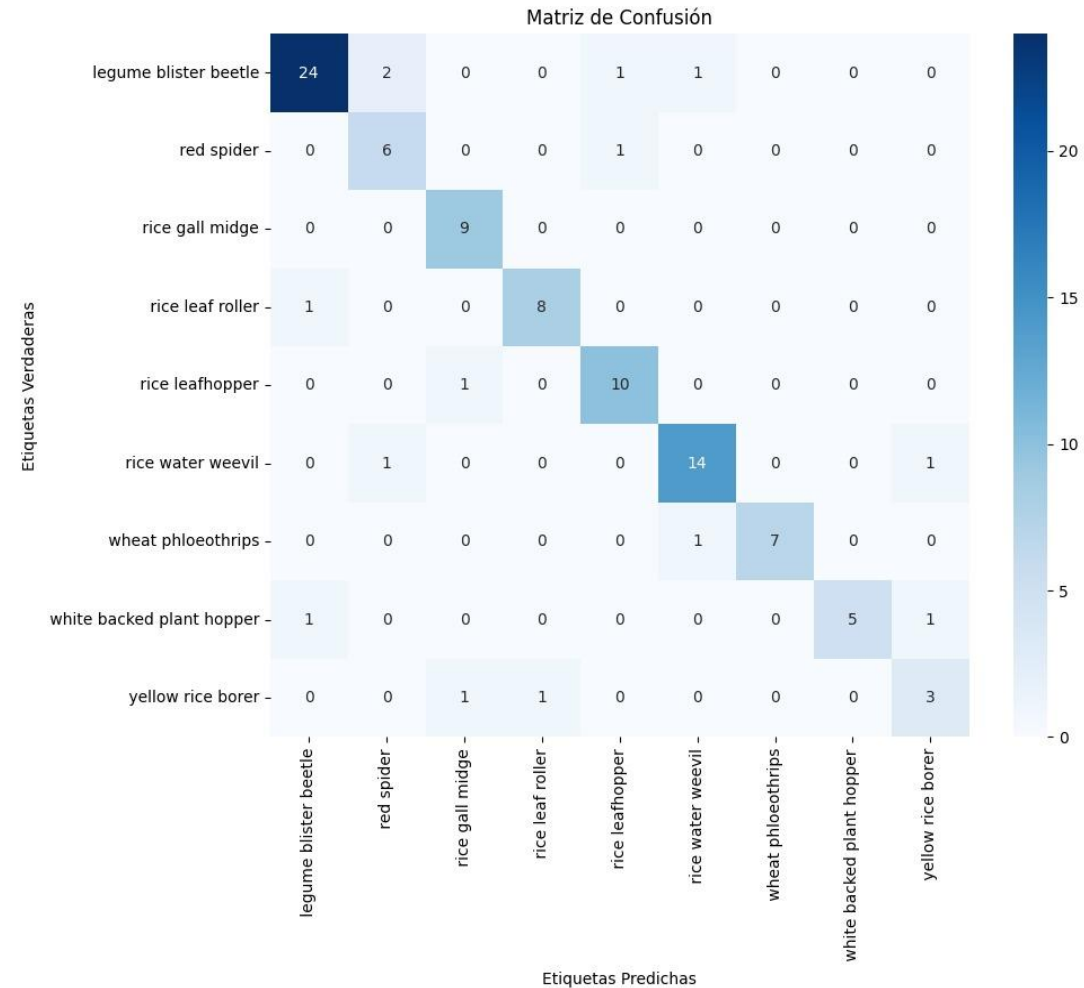


# Modelos utilizados - Métricas

## MobileNetV2

```
3/3 [=====] - 0s 42ms/step - loss: 0.4004 - accuracy: 0.8958  
Test Loss: 0.4004108033180237  
Test Accuracy: 0.8958333134651184
```

```
4/4 [=====] - 2s 54ms/step  
precision recall f1-score support  
  
legume blister beetle 0.96 0.86 0.91 28  
red spider 1.00 1.00 1.00 7  
rice gall midge 1.00 0.78 0.88 9  
rice leaf roller 0.90 1.00 0.95 9  
rice leafhopper 0.92 1.00 0.96 11  
rice water weevil 1.00 0.88 0.93 16  
wheat phloeothrips 0.53 1.00 0.70 8  
white backed plant hopper 1.00 0.71 0.83 7  
yellow rice borer 0.80 0.80 0.80 5  
  
accuracy 0.89 100  
macro avg 0.90 0.89 0.88 100  
weighted avg 0.92 0.89 0.90 100
```

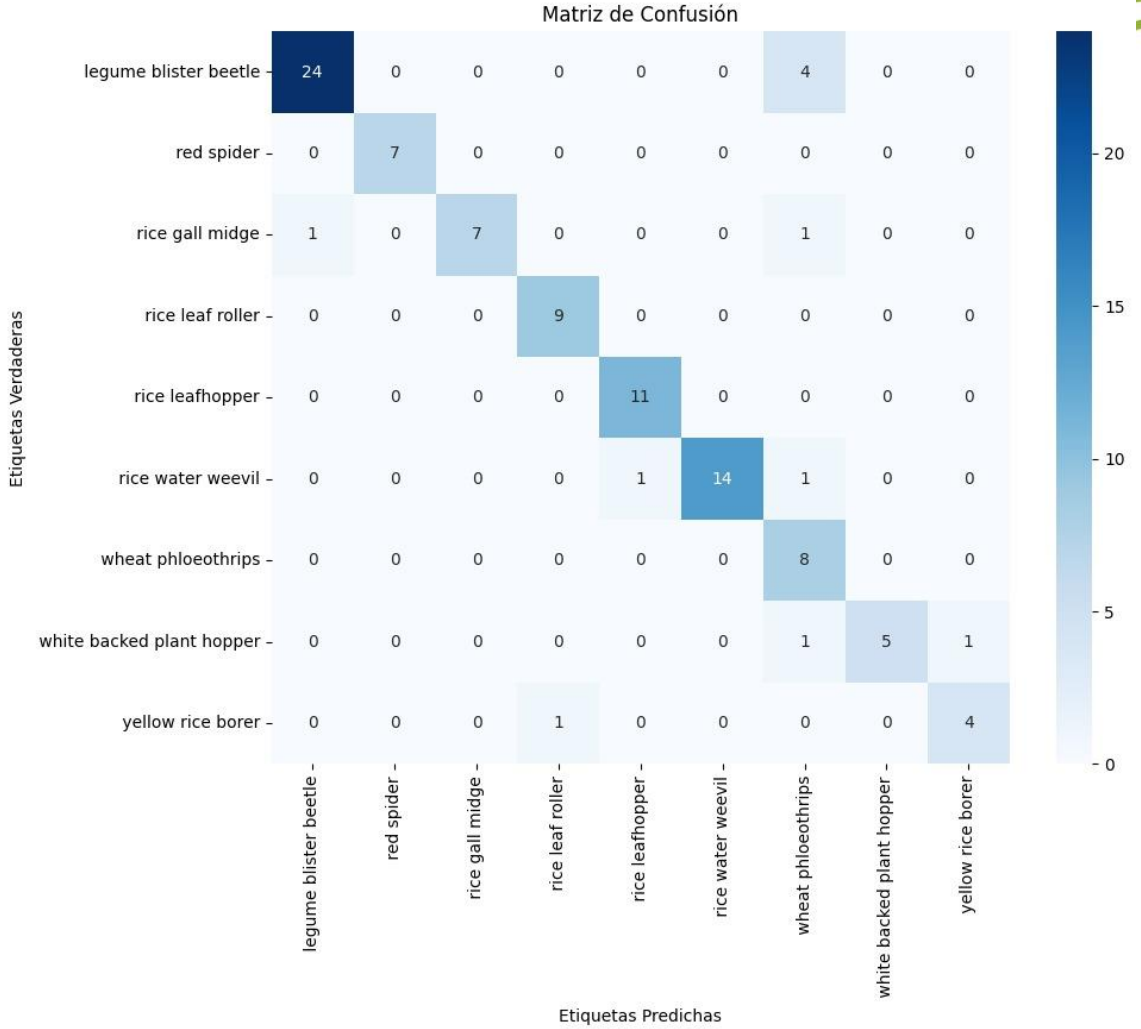


# Modelos utilizados - Métricas

## VGG16

3/3 [=====] - 0s 80ms/step - loss: 0.4408 - accuracy: 0.8542  
Test Loss: 0.4408233165740967  
Test Accuracy: 0.8541666865348816

4/4 [=====] - 3s 715ms/step				
	precision	recall	f1-score	support
legume blister beetle	0.92	0.86	0.89	28
red spider	0.67	0.86	0.75	7
rice gall midge	0.82	1.00	0.90	9
rice leaf roller	0.89	0.89	0.89	9
rice leafhopper	0.83	0.91	0.87	11
rice water weevil	0.88	0.88	0.88	16
wheat phloeothrips	1.00	0.88	0.93	8
white backed plant hopper	1.00	0.71	0.83	7
yellow rice borer	0.60	0.60	0.60	5
accuracy			0.86	100
macro avg	0.85	0.84	0.84	100
weighted avg	0.87	0.86	0.86	100



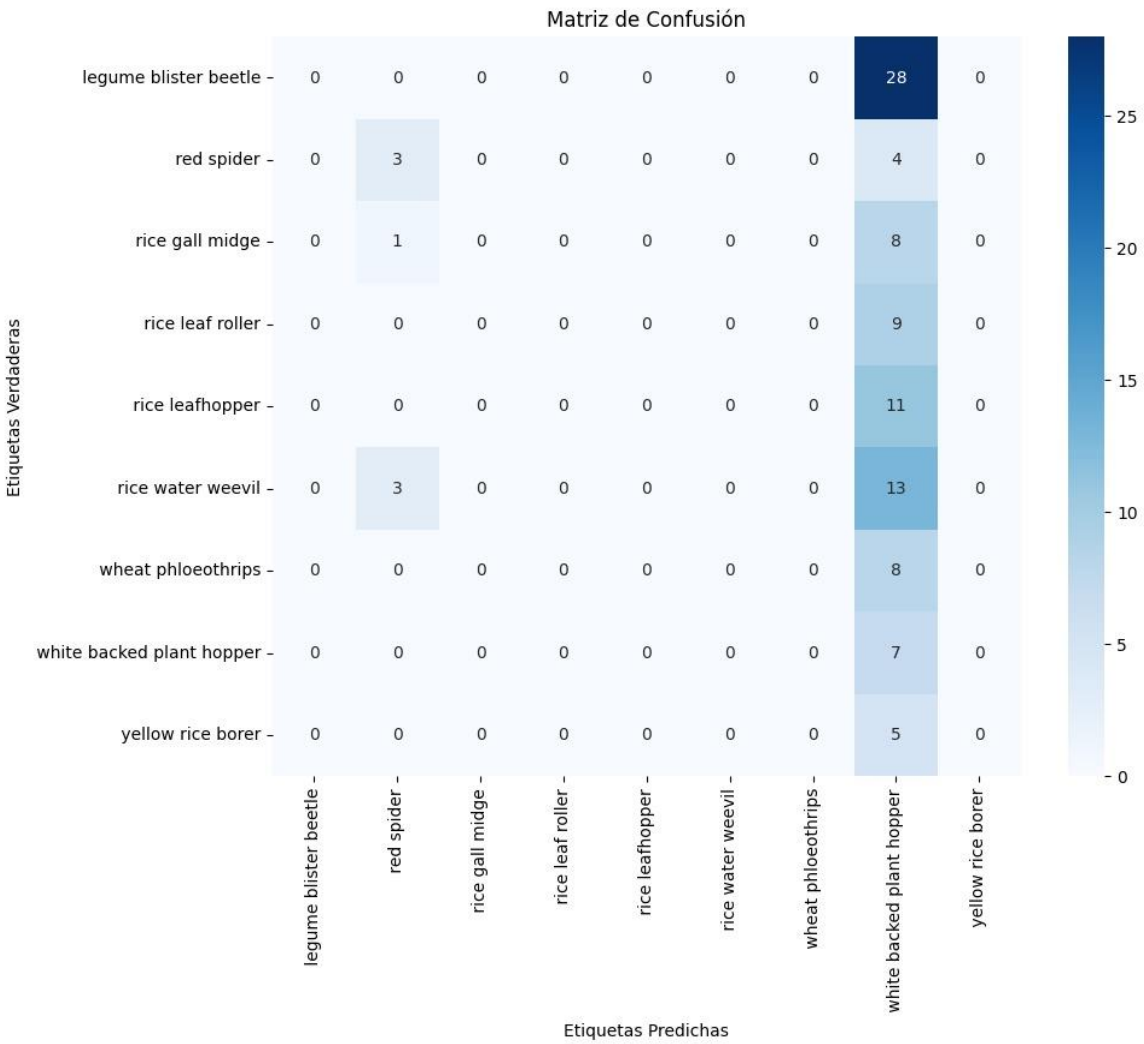


# Modelos utilizados - Métricas

## Arquitectura Propia

```
3/3 [=====] - 0s 50ms/step - loss: 5.3120 - accuracy: 0.1042
Test Loss: 5.311956882476807
Test Accuracy: 0.1041666641831398
```

	precision	recall	f1-score	support
legume blister beetle	0.00	0.00	0.00	28
red spider	0.43	0.43	0.43	7
rice gall midge	0.00	0.00	0.00	9
rice leaf roller	0.00	0.00	0.00	9
rice leafhopper	0.00	0.00	0.00	11
rice water weevil	0.00	0.00	0.00	16
wheat phloeothrips	0.00	0.00	0.00	8
white backed plant hopper	0.08	1.00	0.14	7
yellow rice borer	0.00	0.00	0.00	5
accuracy			0.10	100
macro avg	0.06	0.16	0.06	100
weighted avg	0.04	0.10	0.04	100



# ¿Por qué MobileNetv2?

**Test Loss más bajo:** MobileNetV2 tiene una pérdida de prueba más baja en comparación con VGG16 y la arquitectura personalizada, lo que indica que el modelo está haciendo mejores predicciones con menos errores en el conjunto de datos de prueba.

**Mayor Test Accuracy:** La exactitud de MobileNetV2 es superior a las de VGG16 y la arquitectura personalizada, lo que significa que está clasificando correctamente un mayor porcentaje de imágenes de prueba.

**Consistencia en Precision y Recall:** MobileNetV2 muestra un equilibrio sólido entre precisión y recall, lo que indica que no solo es bueno identificando correctamente las clases positivas (alta precisión), sino que también es confiable al detectar la mayoría de los positivos reales disponibles (alto recall).

**Alto F1-Score:** El F1-score de MobileNetV2 es relativamente alto a través de las clases, lo que implica un buen equilibrio entre precisión y recall. Esto es importante en la clasificación de múltiples clases donde se desea un rendimiento equilibrado entre todas las categorías.

**Matriz de Confusión:** La matriz de confusión para MobileNetV2 muestra una distribución equilibrada de predicciones correctas, con menos confusiones entre las clases en comparación con las otras dos arquitecturas.





# Tecnologías utilizadas

Para esta primer version se usaron las siguientes tecnologias (Despliegue local)

- **Lenguajes de programación:** python y Javascript
- **Lenguaje de etiquetas:** HTML
- **Archivo de estilos web:** CSS
- **Docker:** Dos contenedores, uno para la aplicación web y otro para la base de datos
- **Base de datos:** relacional MySql con el gestor de bases php My admin
- **Jupyter notebook:** para el entrenamiento de los modelos
- **Git y Github:** para el manejo de versiones



# Funcionamiento de la app



Guillermo Umaña Ramirez

<https://github.com/Alejoval26/Proyecto-Final-Clasificador-Insectos---CursoIAFullStack>



**Gracias**