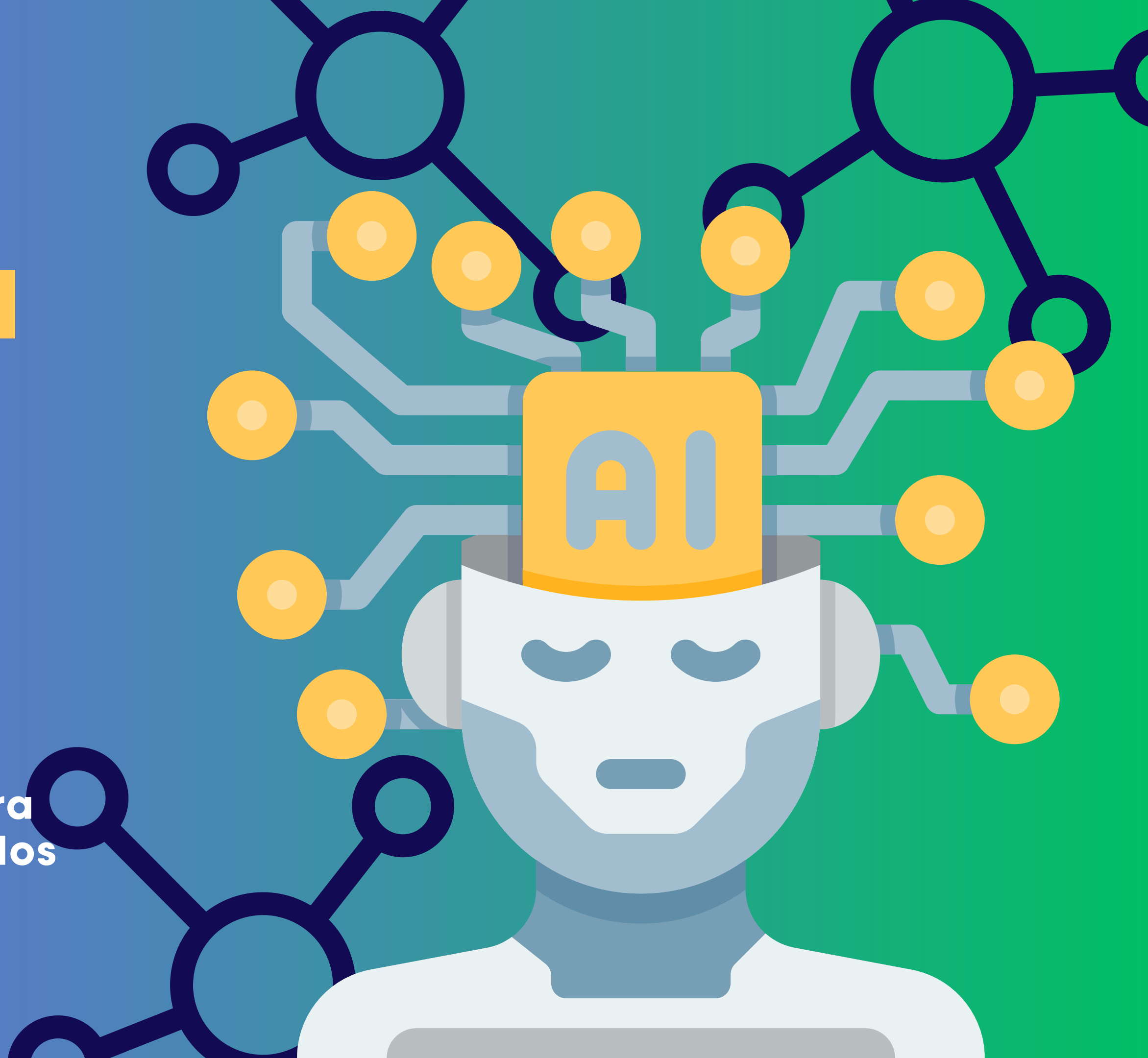


AutoParts A.I Classifier

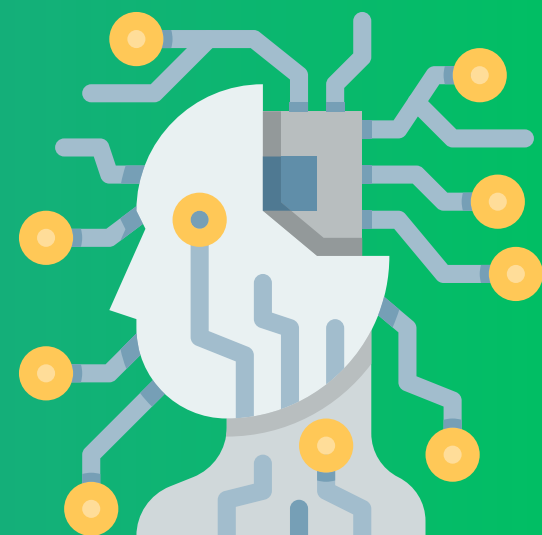
Oscar Julian Rey Jaimes
Juan Diego Sepulveda Herrera
Diego Andres Clavijo Granados



Contenido

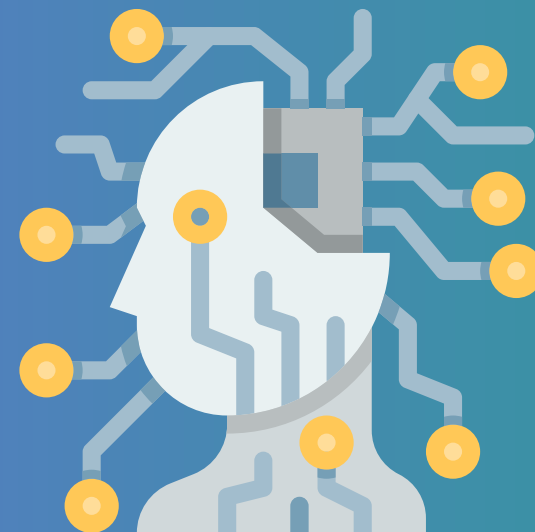


- Motivación y alcance
- Dataset utilizado
- Modelos implementados
- Pipeline
- Resultados
- Conclusiones



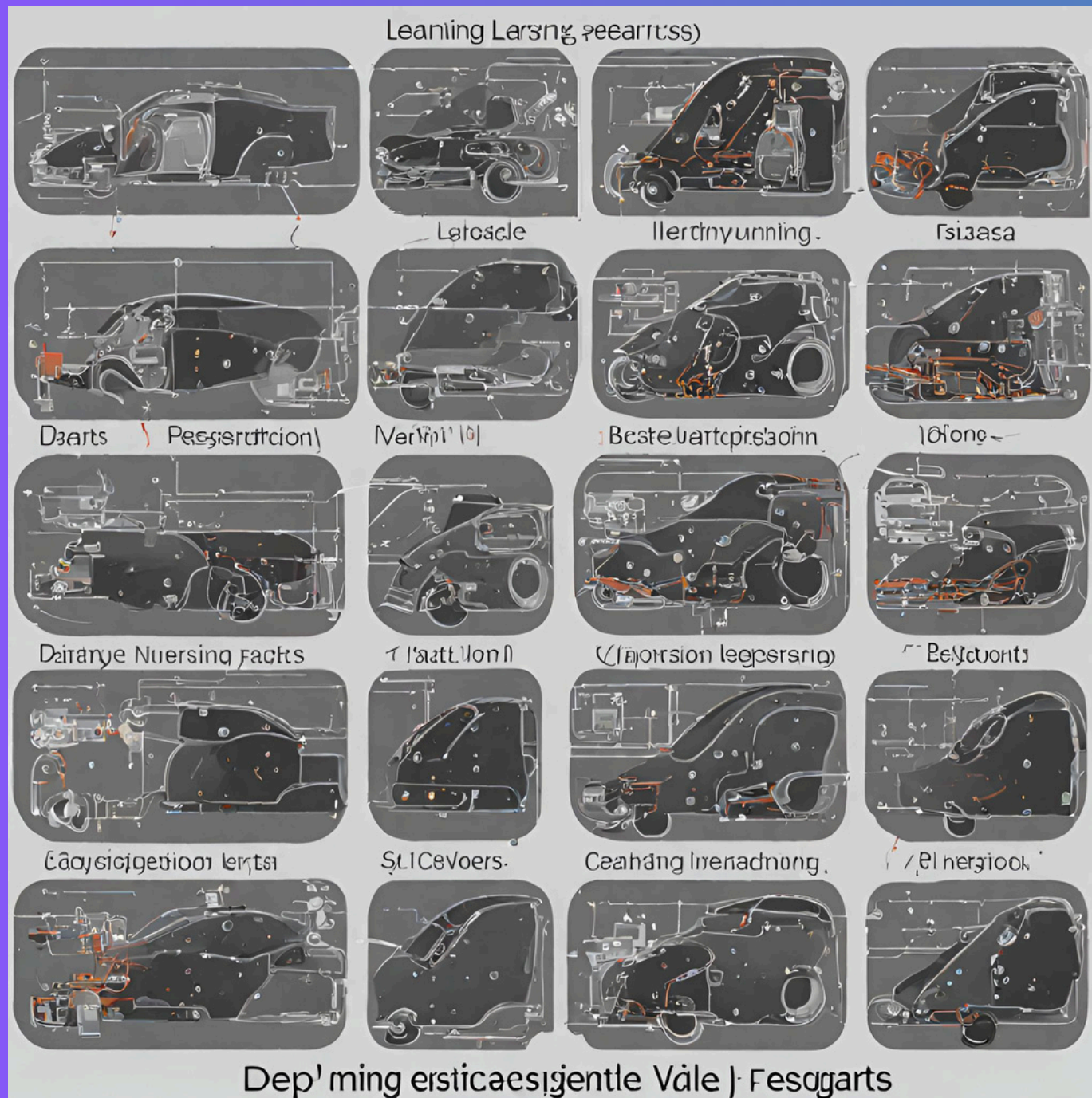
Motivacion

En un almacén de repuestos para vehículos, la clasificación manual es propensa a errores debido a la amplia variedad de modelos y marcas, tediosa y repetitiva. Implementar un sistema de inteligencia artificial puede simplificar este proceso, ofreciendo clasificación a los repuestos.













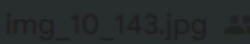
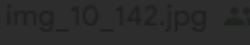
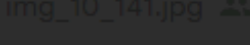
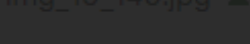
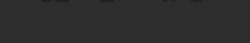



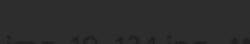
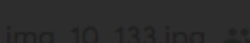
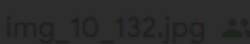
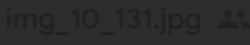

Alcance


El presente proyecto tiene como objetivo clasificar diferentes tipos de repuestos para vehículos utilizando técnicas de deep learning como clasificación.
















Dataset

Nombre	↓
	class_10
	class_9
	class_8
	class_7
	class_6
	class_5
	class_4
	class_3
	class_2
	class_1



 yo  yo  yo  yo  yo  yo  yo  yo  yo  yo  yo  yo  yo

Classes

1



2



3



4



5



6



7



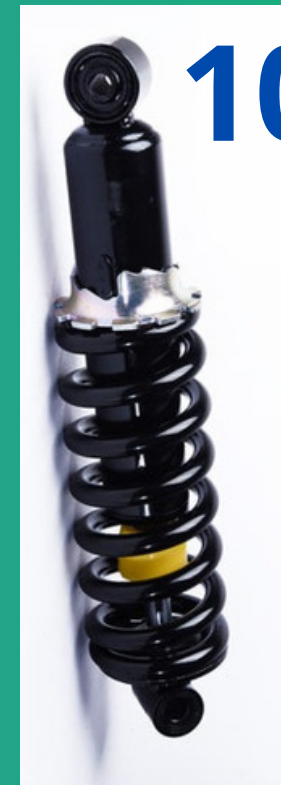
8



9



10



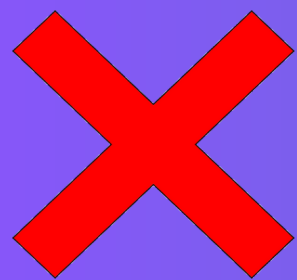
Modelos usados:



Vgg 16

```
# Evaluar el modelo
vgg16_eval = model_vgg16.evaluate(validation_generator)
print(f'Model VGG16 - Loss: {vgg16_eval[0]}, Accuracy: {vgg16_eval[1]}')

10/10 [=====] - 19s 2s/step - loss: 0.3898 - accuracy: 0.8812
Model VGG16 - Loss: 0.3898434042930603, Accuracy: 0.8811880946159363
```

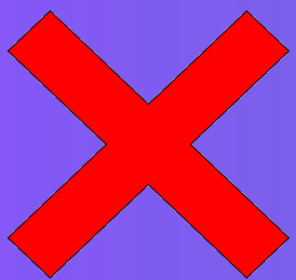


Vgg 19

```
[ ] vgg19_eval = model_vgg19.evaluate(validation_generator)
    print(f'Model VGG19 - Loss: {vgg19_eval[0]}, Accuracy: {vgg19_eval[1]}')

⇒ 10/10 [=====] - 26s 3s/step - loss: 0.6552 - accuracy: 0.7690
Model VGG19 - Loss: 0.6551784873008728, Accuracy: 0.7689769268035889
```

7

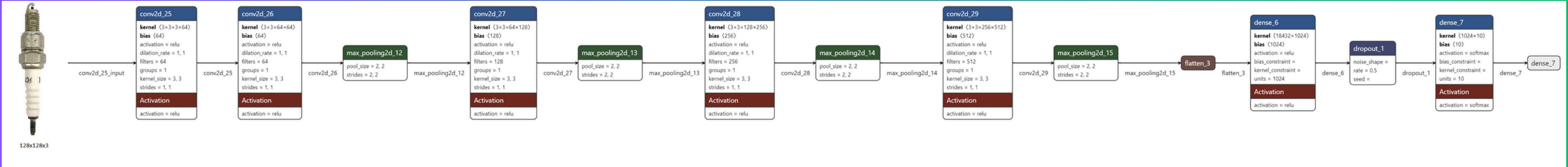


CNN

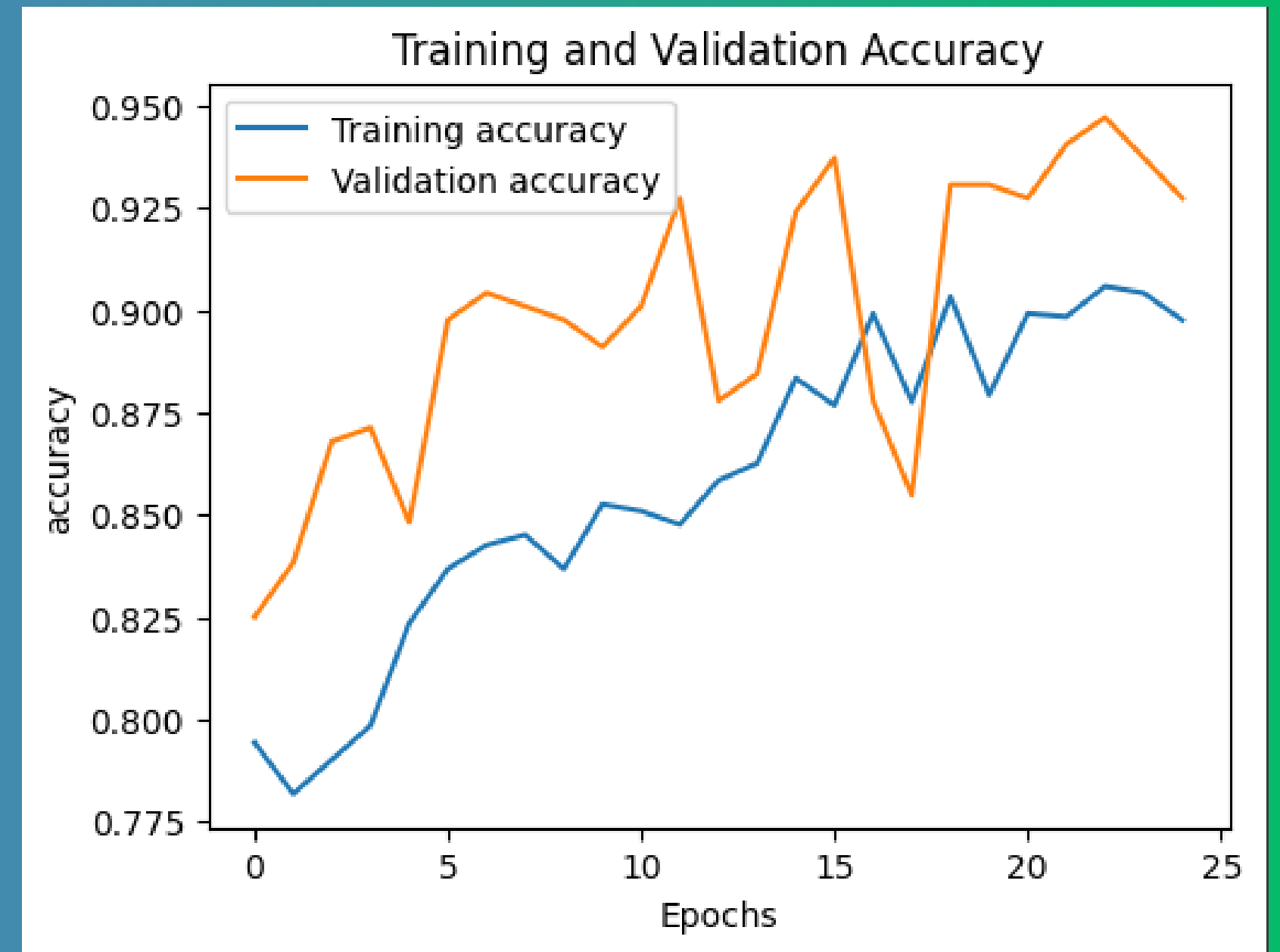
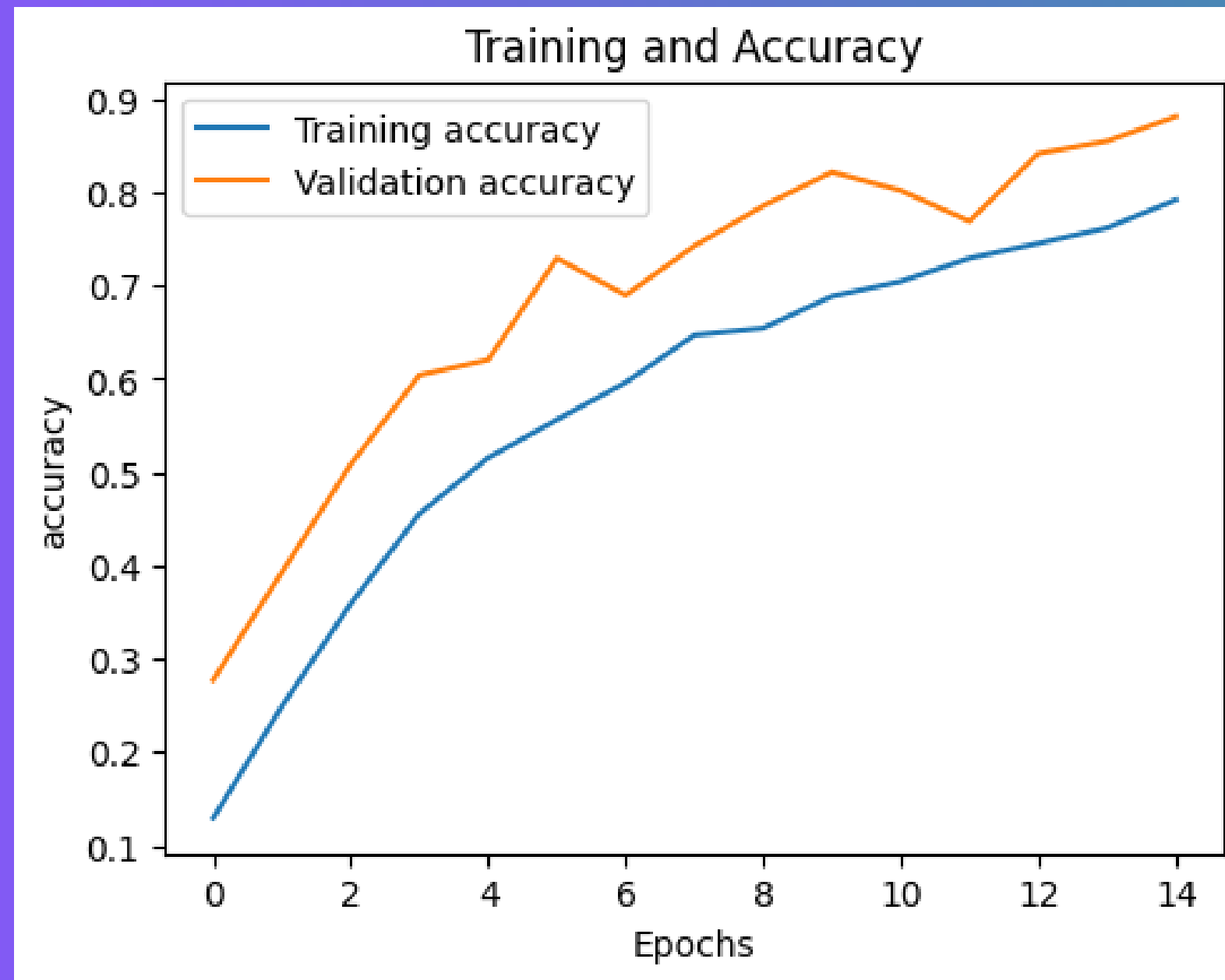
```
# Evaluar el modelo
cnn_eval = model_cnn.evaluate(validation_generator)
print(f'Model CNN - Loss: {cnn_eval[0]}, Accuracy: {cnn_eval[1]}')

10/10 [=====] - 4s 425ms/step - loss: 0.6007 - accuracy: 0.7888
Model CNN - Loss: 0.6007445454597473, Accuracy: 0.7887789011001587
```


Pipeline Vgg16 model implementedo



Gráficas



Resultados



Accuracy: 0.9273927392739274
Precision: 0.9316892421706351
Recall: 0.9273927392739274

Conclusiones

- **VGG16:** Mejor equilibrio entre profundidad y generalización, regularización efectiva, y buen aprovechamiento de augmentación de datos.
- **VGG19:** Mayor profundidad no resultó en mejor rendimiento, más complejo y susceptible al sobreajuste.
- **CNN Personalizada:** Simplicidad en la arquitectura, entrenamiento rápido pero menor capacidad de generalización y extracción de características complejas.