Alert Tomato

Presentado por:
Dairon Alexis Vallejo Vallejo
Erika Yamile Lache Blanco

¿Por qué? ¿Para qué?

- Conocer los tipos de enfermedades en hojas de tomate que pueden estar en un cultivo colombiano.
- Tomar medidas preventivas y evitar la propagación de esta en todo el cultivo.

- A largo plazo, evitar perdidas en los cultivos de tomate y obtener una planta saludable en el proceso de cosecha.

Objetivo

Clasificar el tipo de enfermedad que puede sufrir la planta de tomate usando modelos de Deep learning a partir de características captadas por medio de las hojas de los tomates.

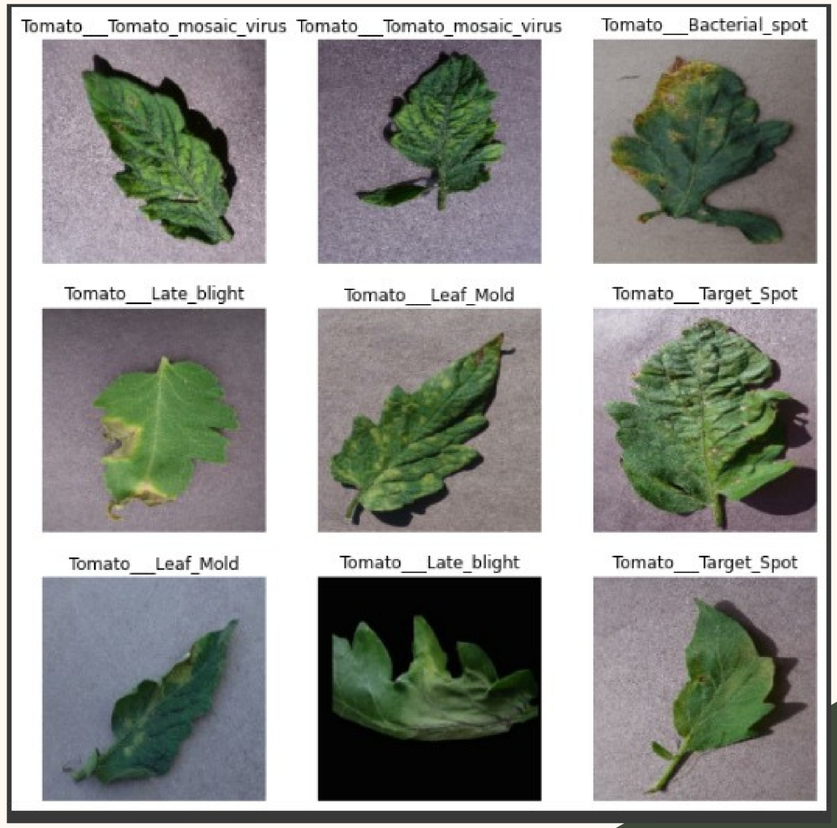
Introducción

11000 imágenes etiquetadas para la clasificación de enfermedades de la hoja del tomate con 10 clases.

Partición de Datos

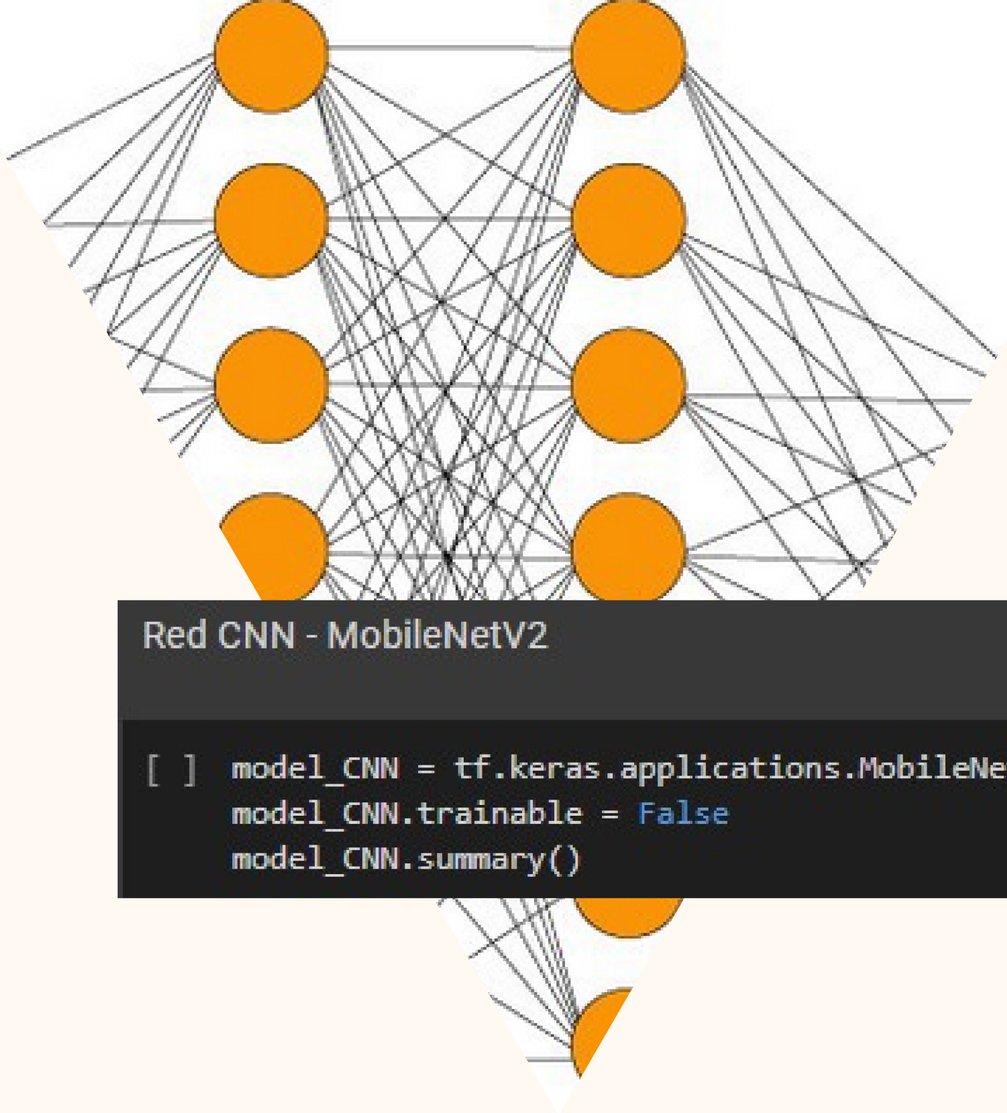
```
dir2= "/content/drive/MyDrive/PROYECTO/tomato/train"
entrenados2 = tf.keras.preprocessing.image_dataset_from_directory(directory=dir2,

                                                  image_size=(224,224),
                                                  batch_size=64,
                                                  subset="training",
                                                  shuffle=True,
                                                  seed=123,
                                                  validation_split=0.2
                                                  )
```

```
Found 10000 files belonging to 10 classes.
Using 8000 files for training.
```



20% Validation

80% Train



MobileNetV2

Red CNN - MobileNetV2

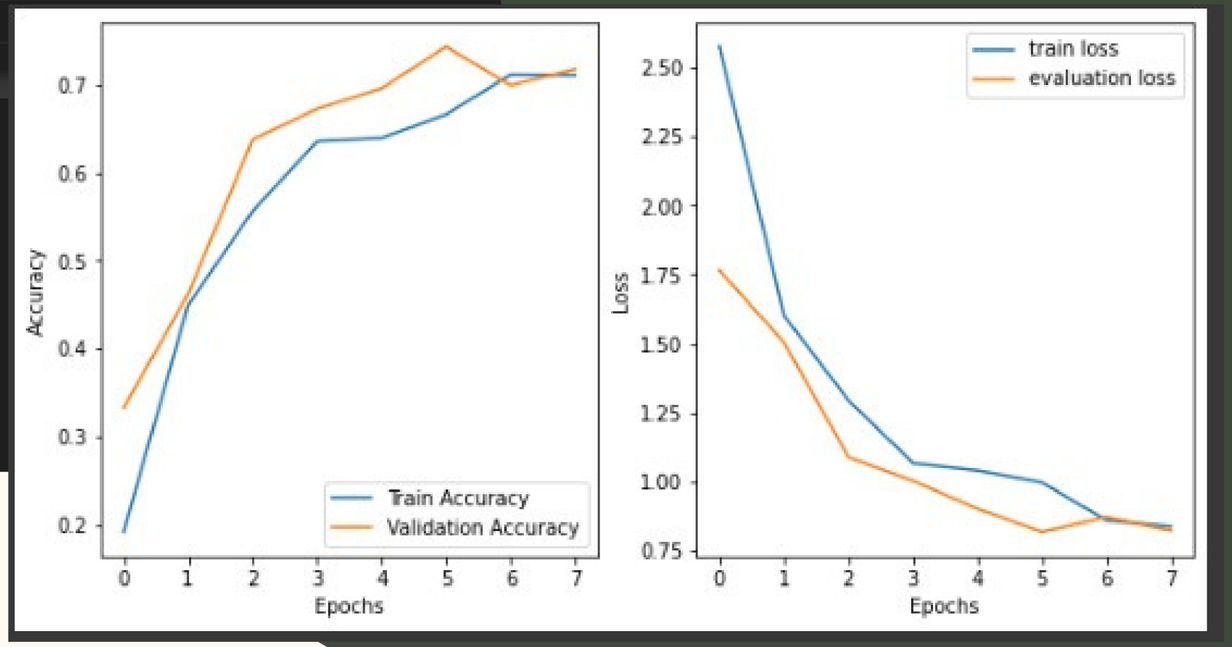
```
[ ]  model_CNN = tf.keras.applications.MobileNetV2(weights='imagenet', include_top=False, input_shape=(224,224,3))
     model_CNN.trainable = False
     model_CNN.summary()
```

Accuracy

```
opt = tf.keras.optimizers.Adam(lr=0.001)
model_CNN_2.compile(optimizer=opt,loss='categorical_crossentropy',metrics=['accuracy'])
```

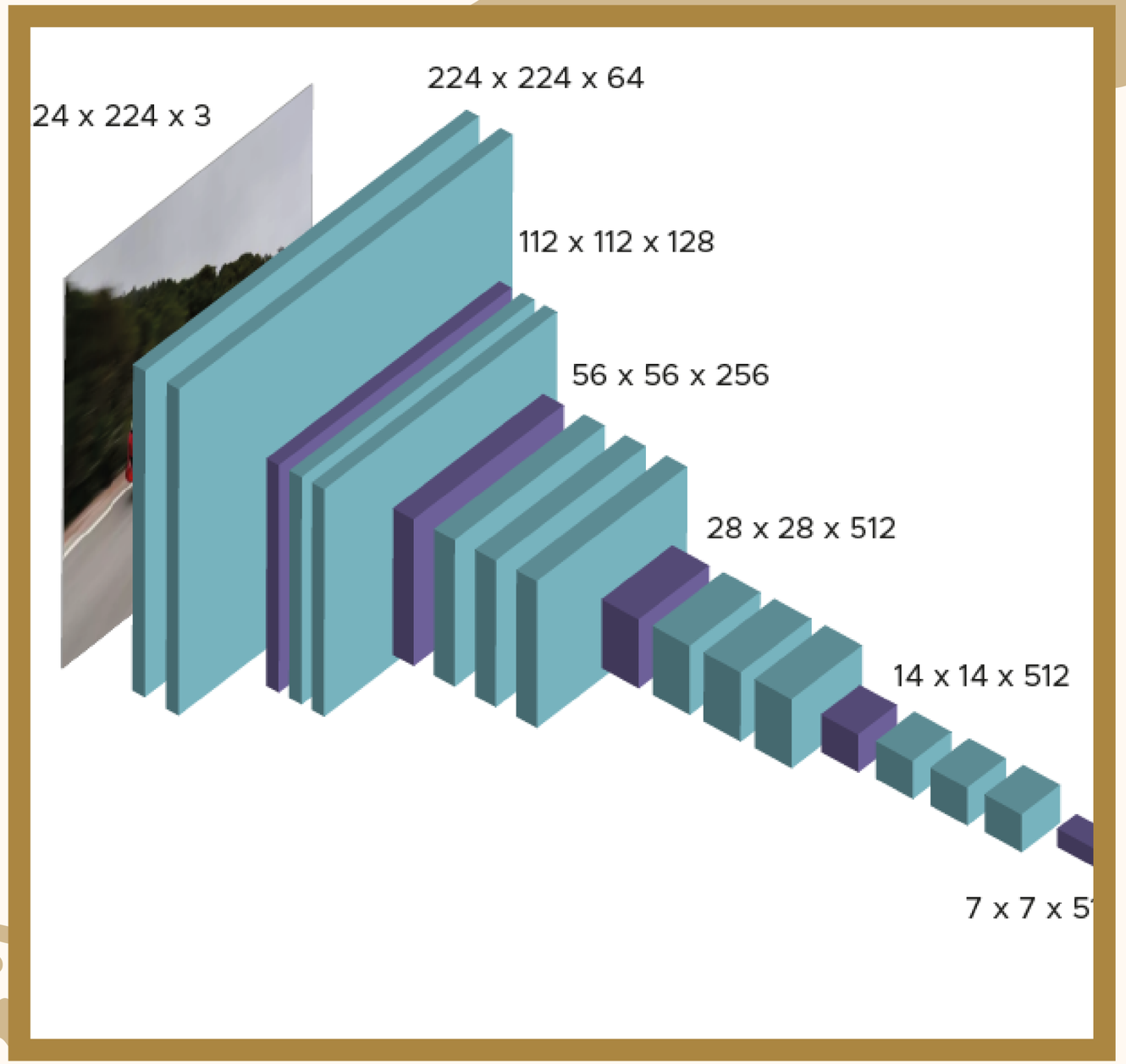
```
early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy',
                                              patience=2,
                                              mode='auto',
                                              restore_best_weights=True)
```

```
history =  model_CNN_2.fit_generator(generator=entrenados,
                                     steps_per_epoch=pasos_entrenamiento,
                                     validation_data=validados,
                                     validation_steps=pasos_validacion,
                                     epochs=10,
                                     callbacks=[early_stop, reduce_lr]
                                     )
```

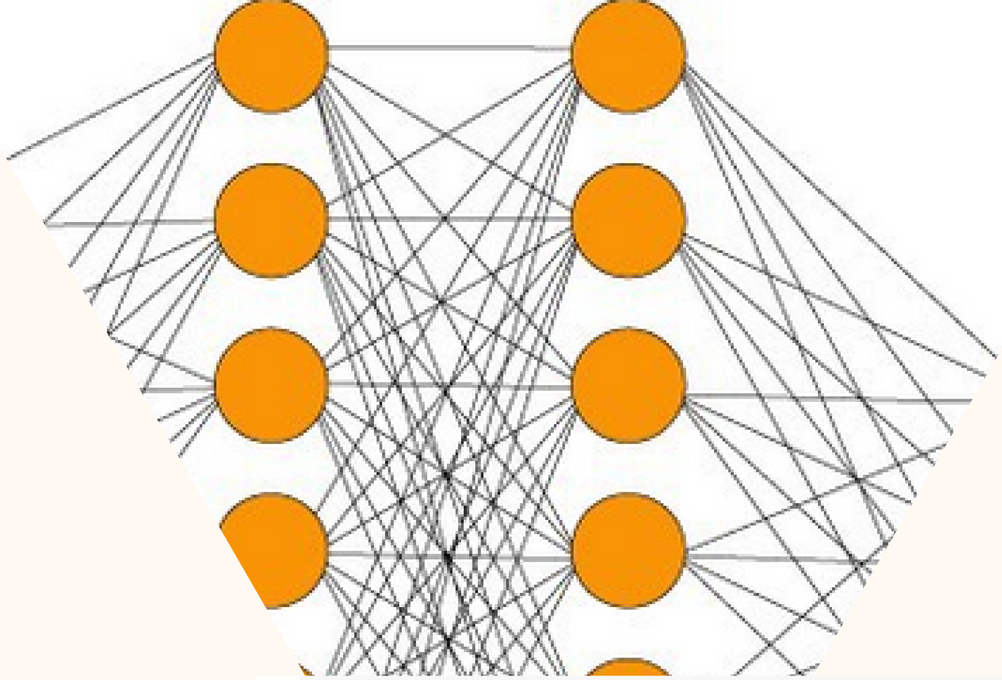


Red Neuronal

El objetivo de estas capas es filtrar la imagen manteniendo sólo la información discriminante, como las formas geométricas atípicas.



VGG16



VGG16

```
[81] image_size = [256, 256]
     vgg = VGG16(input_shape = image_size + [3], weights = 'imagenet', include_top = False)
```

Implementación

```
[11] x = Flatten()(vgg.output)
     prediction = Dense(len(folders), activation = 'softmax')(x)
     model = Model(inputs = vgg.input, outputs = prediction)
```

```
[19] from tensorflow.keras.preprocessing.image import ImageDataGenerator
     train_data_gen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2, horizontal_flip = True)
     test_data_gen = ImageDataGenerator(rescale = 1./255)
     train_set = train_data_gen.flow_from_directory('./tomato/train/', target_size = (256,256), batch_size = 32, class_mode = 'categorical')
```

```
     Found 924 images belonging to 10 classes.
```

```
[20] test_set = test_data_gen.flow_from_directory('./tomato/val/', target_size = (256,256), batch_size = 32, class_mode = 'categorical')
```

```
     Found 690 images belonging to 10 classes.
```

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

```
     #correr con 30 epoch para llegar a un acc del 96% y una perdida de 0.8
     #correr con 50 epoch para llegar a un acc del 97% y una perdida de 0.8
     history = model.fit_generator(
       train_set,
       validation_data=test_set,
       epochs=2
     )
```