07MIAR_Proyecto_Programacion

November 26, 2023

1 07MIAR - PROYECTO DE PROGRAMACIÓN

- Guido Alexander Heienbrok – gaheienbrok@gmail.com
- Oscar García González – oscargargon3@gmail.com
- Bruno González Fernández – brunogfrr@gmail.com

DATASET: https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset

OBJETIVOS: Evaluar y comparar dos estrategias para la clasificación de imágenes empleando el dataset asignado. Los alumnos deberá resolver el reto proponiendo una solución válida basada en aprendizaje profundo, más concretamente en redes neuronales convolucionales (CNNs). Será indispensable que la solución propuesta siga el pipeline visto en clase para resolver este tipo de tareas de inteligencia artificial:

1. **Carga** del conjunto de datos
2. **Inspección** del conjunto de datos
3. **Acondicionamiento** del conjunto de datos
4. Desarrollo de la **arquitectura*** de red neuronal y **entrenamiento** de la solución
5. **Monitorización** del proceso de entrenamiento para la toma de decisiones
6. **Evaluación** del modelo predictivo y planteamiento de la siguiente prueba experimental

2 ÍNDICE

3 0. Metodología

El proyecto se compone de 3 fases: preparación de los datos, desarrollo y entrenamiento de modelos utilizando redes neuronales from scratch, desarrollo y entrenamiento de modelos utilizando redes neuronales pre-entrenadas.

Se pone el foco en la comparativa de diferentes arquitecturas y técnicas de regularización y data augmentation: MLP, CNN, y varias redes pre-entrenadas (VGG16, Xception, InceptionV3, y ResNet50) siguiendo estrategias de transfer learning y fine tuning (parcial y completo).

Todos los apartados incluyen funciones que soportan las arquitecturas sobre las que se ejecutan los entrenamientos de los diferentes experimentos. Los modelos se guardan para facilitar su análisis y repartir las tareas de entrenamiento entre los miembros del equipo.

Los entrenamientos incluye un callback de parada temprana (*early stopping*) a partir de una epoch predefinida para controlar la convergencia y el sobreajuste de los modelos.

Se ha buscado diseñar un código al estilo funcional en todo momento. Esto es, que toda arquitectura esté embebida en una función que automatiza el proceso de aprendizaje y de variación de parámetros.

4 1. Configuración

En esta sección se fijan los hiperparámetros generales que regulan todo el notebook.

DATOS

- **BASE_FOLDER**: carpeta en la que se guardan los modelos, datos y resultados.
- **MODEL_PATH**: carpeta en la que se guardan los modelos.
- **dataset_name**: nombre del dataset de kaggle correspondiente a este proyecto.

ARQUITECTURAS

- **target_size**: tramaño de entrada de las imágenes, fijado en 75x75 pixels.
- **dense_size**: tamaño de capas densas de entrada.
- **conv2d_size**: tamaño de capas convolucionales de entrada.

ENTRENAMIENTO

- **batch_size**: número de instancias que se introducen en las redes para que en cada actualización durante el entrenamiento.
- **learning_rate**: tasa de aprendizaje.
- **epochs**: número de épocas máximo con el que se entrenen las redes.
- **early_stopping_patience**: número de épocas sin mejora después de las cuales se detendrá el entrenamiento cuando se utilice el callback de parada temprana.
- **early_stopping_monitor**: métrica a monitorizar

- **start_from_epoch**: número de épocas a esperar antes de comenzar a monitorear con el callback de parada temprana (early stopping).

[1]:

```
# CONFIGURACIÓN EXPERIMENTOS

from google.colab import drive
drive.mount('/content/drive')

BASE_FOLDER = "/content/drive/MyDrive/07MIAR_Proyecto_Programacion/"
MODEL_PATH = "Models"

dataset_name = "misrakahmed/vegetable-image-dataset"
```

```
Mounted at /content/drive
```

ARQUITECTURAS

- **target_size**: tramaño de entrada de las imágenes, fijado en 75x75 pixels.
- **dense_size**: tamaño de capas densas de entrada.
- **conv2d_size**: tamaño de capas convolucionales de entrada.

ENTRENAMIENTO - **do_training**: flag para entrenar modelo o leerlo de la carpeta Models, para que funcione el modelo tendrá que haber sido entrenado previamente (facilita el trabajo en equipo y la reproducibilidad de resultados, repartiendo la carga de entrenamiento entre todos los miembros del equpo). - **batch_size**: número de instancias que se introducen en las redes para que en cada actualización durante el entrenamiento. - **learning_rate**: tasa de aprendizaje. - **epochs**:

número de épocas máximo con el que se entrenen las redes. - **early_stopping_patience**: número de épocas sin mejora después de las cuales se detendrá el entrenamiento cuando se utilice el callback de parada temprana. - **early_stopping_monitor**: métrica a monitorizar
- **start_from_epoch**: número de épocas a esperar antes de comenzar a monitorear con el callback de parada temprana (early stopping). - **exp_set**: set que almacena el Id de todos los experimentos ejecutados en el notebbook.

[2]:

```
# Redes conv y densas
target_size = (75, 75) # entrada mínima InceptionV3
dense_size=128
conv2d_size= 16 #entrada

# Entrenamiento
do_training = True
batch_size=256 # reducir al probar el notebook
learning_rate=0.001
epochs=50  # reducir al probar el notebook
early_stopping_patience = 3
early_stopping_monitor = 'val_loss'
start_from_epoch = epochs // 2

# Inicialización lst de experimentos
exp_set = set()
```

5 2. Imports y funciones base

En esta sección se **importan librerías** necesarias y se definen: la semilla aleatoria, el callback de **parada temprana**, y **funciones base** para visualizacón de **curvas de aprendizaje** y **evaluación de modelos** y **persistencia en drive**.

[ ]:

```
# Importar todas las librerías necesarias

# Librerías base
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import pickle
import numpy as np

# Preprocesamiento de datos
from google.colab import files
import random
import os
import cv2
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Keras
```

```
from tensorflow.keras import layers, regularizers
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout,␣
 ↪BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.utils import set_random_seed
from tensorflow.keras.applications import VGG16, VGG19, ResNet50, Xception,␣
 ↪InceptionV3, InceptionResNetV2, MobileNetV2, DenseNet121, NASNetLarge,␣
 ↪ResNet101

# Definición callback para earling stopping en entrenamientos
early_stopping_cbck = EarlyStopping(monitor=early_stopping_monitor,␣
 ↪patience=early_stopping_patience, start_from_epoch=start_from_epoch)

# Semilla
set_random_seed(202311)
```

Definir funciones esenciales del proyecto:

visualize_learning_curve(H, lb): Genera gráficos para visualizar la curva de aprendizaje del modelo, mostrando tanto la pérdida (loss) como la precisión (accuracy) durante el entrenamiento y la validación a lo largo de las épocas.

evaluate_model(model, x, y): Evalúa el modelo de red neuronal proporcionado, imprimiendo un informe de clasificación que incluye métricas clave como precisión, recall y F1-score para cada clase.

load_images_and_labels(data_dir, target_size): Carga imágenes y sus etiquetas correspondientes desde un directorio especificado, ajustando el tamaño de todas las imágenes al tamaño objetivo para su posterior procesamiento y entrenamiento en la red neuronal.

save_trained_model(model, model_name, history, base_folder, model_path): persistencia del modelo y su historia de entrenamiento de un *model_name* dado.

load_history(model_name, base_folder, model_path): lectura de la historia de entrenamiento de un *model_name* dado.

load_keras_model(model_name, base_folder, model_path): lectura del modelo dado su *model_name*.

[ ]:

```
# Funciones base

# Visualizar la curva de aprendizaje
def visualize_learning_curve(H, lb =""):
  epochs = len(H.history["loss"]) #  El tamaño de la lista es el número de␣
 ↪épocas.
  plt.style.use("ggplot")
  plt.figure()
  plt.plot(np.arange(0, epochs), H.history["loss"], label="train_loss")
  plt.plot(np.arange(0, epochs), H.history["val_loss"], label="val_loss")
  plt.plot(np.arange(0, epochs), H.history["accuracy"], label="train_acc")
  plt.plot(np.arange(0, epochs), H.history["val_accuracy"], label="val_acc")
  plt.title("Training Loss and Accuracy " + lb)
  plt.xlabel("Epoch #")
  plt.ylabel("Loss/Accuracy")
  plt.legend()

# Evaluación del modelo
def evaluate_model(model, x, y):
  print("[INFO]: Evaluando red neuronal...")
  predictions = model.predict(x, batch_size=128)
  print(classification_report(y, predictions.argmax(axis=1)))

# Evaluación del modelo cuando hay Data Augmentation con flow_from_directory
def evaluate_model_aug(model, generator):
  print("[INFO]: Evaluando red neuronal...")
  # Predecir con el generador de pruebas
  predictions = model.predict_generator(generator, steps=len(generator))
  # Obtener las etiquetas verdaderas del generador
  y_true = generator.classes
  # Convertir las predicciones a etiquetas
  y_pred = np.argmax(predictions, axis=1)
  # Imprimir el informe de clasificación
  print(classification_report(y_true, y_pred))

# Función para cargar imágenes y etiquetas desde el directorio y ajustar tamaño
def load_images_and_labels(data_dir, target_size):
    images = []
    labels = []
    for label, vegetable_class in enumerate(classes):
        class_path = os.path.join(data_dir, vegetable_class)
        for filename in os.listdir(class_path):
            img = cv2.imread(os.path.join(class_path, filename))
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.resize(img, target_size)  # Reajustar tamaño para todas las imágenes
            images.append(img)
            labels.append(label)
    return images, labels

# clase historia de entremiento
class History_trained_model(object):
  def __init__(self, history, epoch, params):
    self.history = history
    self.epoch = epoch
    self.params = params
```

```python
# Función guardado de modelos y su historial de entrenamiento
def save_trained_model(model, model_name, history, base_folder = BASE_FOLDER, model_path = MODEL_PATH):
  path = base_folder + model_path + '/' + model_name

  model.save(path + '.keras')
  print("Saved model to disk")

  with open(path + '_history', 'wb') as file:
    model_history= History_trained_model(history.history, history.epoch, history.params)
    pickle.dump(model_history, file, pickle.HIGHEST_PROTOCOL)

# Función para carga de historia de entrenamiento
def load_history(model_name, base_folder = BASE_FOLDER, model_path = MODEL_PATH):
  path = base_folder + model_path + '/' + model_name

  with open(path +'_history', 'rb') as file:
    history=pickle.load(file)

  with open(path + '_history', 'wb') as file:
    model_history= History_trained_model(history.history, history.epoch, history.params)
    pickle.dump(model_history, file, pickle.HIGHEST_PROTOCOL)

  return model_history

# Función para carga de modelos guardados
def load_keras_model(model_name, base_folder = BASE_FOLDER, model_path = MODEL_PATH):
  path = base_folder + model_path + '/' + model_name

  return load_model(path + '.keras')
```

6 3. Descarga del dataset

[ ]:

```python
# Instalar la última version de Kaggle API en Google Colab
!pip install --upgrade --force-reinstall --no-deps kaggle
```

```
Collecting kaggle
  Using cached kaggle-1.5.16-py3-none-any.whl
Installing collected packages: kaggle
  Attempting uninstall: kaggle
    Found existing installation: kaggle 1.5.16
    Uninstalling kaggle-1.5.16:
      Successfully uninstalled kaggle-1.5.16
Successfully installed kaggle-1.5.16
```

```
[ ]: # Seleccionar el API Token personal previamente descargado (fichero kaggle.json)
     files.upload()
```

```
<IPython.core.display.HTML object>

Saving kaggle.json to kaggle (1).json
```

```
[ ]: {'kaggle (1).json':
     b'{"username":"brunogf","key":"e814d7d3040742a50e51f65264bbc979"}'}
```

```
[ ]: # Crear un directorio en el que copiar el fichero kaggle.json
     !mkdir ~/.kaggle
     !cp kaggle.json ~/.kaggle/
     !chmod 600 ~/.kaggle/kaggle.json
```

```
mkdir: cannot create directory ‘/root/.kaggle’: File exists
```

```
[ ]: import kaggle

     # Usar API Kaggle para descargar los datos.
     kaggle.api.dataset_download_files(dataset_name, path="/content/my_dataset",␣
      ↪unzip=True)
```

```
[ ]: # Comprobar si el data set se ha almacenado (temporalmente) correctamente en␣
      ↪Colab
     # !ls /content

     # OTRA ALTERNATIVA:
     my_dataset_path = '/content/my_dataset'

     if os.path.exists(my_dataset_path):
         folder = os.listdir(my_dataset_path)
         for item in folder:
             print(item)
     else:
         print(f"La carpeta 'my_dataset'({my_dataset_path}) no existe.")
```

```
Vegetable Images
```

7 4. Exploración y preprocesado

7.1 4.1 Obtener datos train, validation y test

1. Definir la ruta al conjunto de datos y listar las etiquetas de clases a partir de los datos de entrenamiento.

2. Inicializar listas para almacenar los conjuntos de datos de entrenamiento, validación y prueba.
3. Cargar las imágenes y sus etiquetas correspondientes para cada conjunto de datos.
4. Convertir las listas de imágenes y etiquetas a arreglos de NumPy.
5. Determinar el número de clases únicas en el conjunto de entrenamiento.

[ ]:

```
# Definir la ruta al data set
dataset_path = '/content/my_dataset/Vegetable Images'

# Definir lista de las etiquetas (tipos de verduras)
classes = os.listdir(os.path.join(dataset_path, 'train'))

# Inicializar listas de train, validación y test
x_train, y_train = [], []
x_val, y_val = [], []
x_test, y_test = [], []

# Almacenar datos train, test y validación
x_train, y_train = load_images_and_labels(os.path.join(dataset_path, 'train'),␣
 ↪target_size)
x_val, y_val = load_images_and_labels(os.path.join(dataset_path, 'validation'),␣
 ↪target_size)
x_test, y_test = load_images_and_labels(os.path.join(dataset_path, 'test'),␣
 ↪target_size)

# Convertir a NumPy arrays
x_train, x_val, x_test = np.array(x_train), np.array(x_val), np.array(x_test)
y_train, y_val, y_test = np.array(y_train), np.array(y_val), np.array(y_test)

num_clases=len(np.unique(y_train))
```

7.2 4.2 Visualizar formato

[ ]:

```
# Mostrar formato de los datos
print("Formato x_train:", x_train.shape)
print("Formato y_train:", y_train.shape, '\n')
print("Formato x_val:", x_val.shape)
print("Formato y_val:", y_val.shape, '\n')
print("Formato x_test:", x_test.shape)
print("Formato y_test:", y_test.shape, '\n')

# Mostrar etiquetas
values_y_test = np.unique(y_test)
values_y_val = np.unique(y_val)
values_y_train = np.unique(y_train)

print("Valores únicos en y_test:", values_y_test)
print("Valores únicos en y_val:", values_y_val)
print("Valores únicos en y_train:", values_y_train)
```

```
Formato x_train: (15000, 75, 75, 3)
Formato y_train: (15000,)

Formato x_val: (3000, 75, 75, 3)
Formato y_val: (3000,)

Formato x_test: (3000, 75, 75, 3)
Formato y_test: (3000,)

Valores únicos en y_test: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
Valores únicos en y_val: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
Valores únicos en y_train: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

7.3 4.3 Normalizar

Normalizar los conjuntos de datos de entrenamiento, prueba y validación dividiendo cada píxel por 255, lo cual escala los valores de píxeles a un rango de 0 a 1.

[ ]:

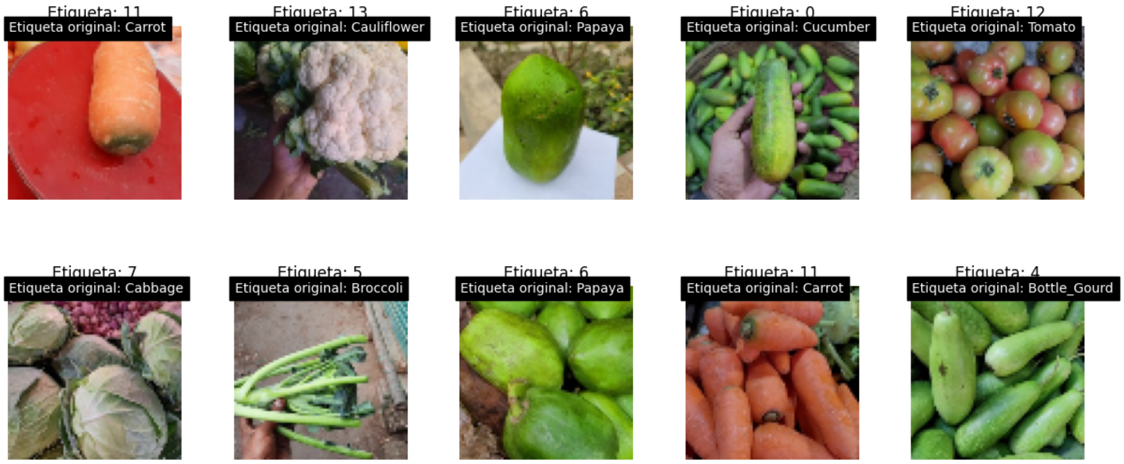
```
# Normalizar datos
x_train_norm = x_train / 255.0
x_test_norm = x_test / 255.0
x_val_norm = x_val / 255.0
```

7.4 4.4 Visualizar muestras

Seleccionar aleatoriamente y visualizar un conjunto de ejemplos del conjunto de entrenamiento, mostrando la imagen y sus etiquetas correspondientes.

[ ]:

```
# Número de ejemplos
num_samples = 10

# Generar lista de ejemplos aleatorios
random_indices = random.sample(range(len(x_train)), num_samples)

# Generar plot
fig, axes = plt.subplots(2, 5, figsize=(15, 6))
fig.subplots_adjust(hspace=0.5)
for i, ax in enumerate(axes.flat):
    index = random_indices[i]

    # Mostrar
    ax.imshow(x_train[index])
    ax.set_title(f"Etiqueta: {y_train[index]}")
    ax.axis('off')
    original_label = classes[y_train[index]]
    ax.text(0, 2, f"Etiqueta original: {original_label}", color='white',␣
↪backgroundcolor='black')

plt.show()
```


Etiqueta: 11
Etiqueta original: Carrot
Etiqueta: 13
Etiqueta original: Cauliflower
Etiqueta: 6
Etiqueta original: Papaya
Etiqueta: 0
Etiqueta original: Cucumber
Etiqueta: 12
Etiqueta original: Tomato
Etiqueta: 7
Etiqueta original: Cabbage
Etiqueta: 5
Etiqueta original: Broccoli
Etiqueta: 6
Etiqueta original: Papaya
Etiqueta: 11
Etiqueta original: Carrot
Etiqueta: 4
Etiqueta original: Bottle_Gourd

7.5 4.5 Pasar etiquetas a one-hot encoding (OHE)

Aplicar binarización a las etiquetas de los conjuntos de entrenamiento, validación y prueba, convirtiéndolas en representaciones de one-hot encoding, y mostrar la dimensión del conjunto de etiquetas de entrenamiento binarizadas.

```
[ ]: # Acondicionamiento/Binarización dataset

lb = LabelBinarizer()
y_train_ohe = lb.fit_transform(y_train)
y_val_ohe = lb.transform(y_val)
y_test_ohe = lb.transform(y_test)

print(y_train_ohe.shape)
```

```
(15000, 15)
```

7.6 4.6 Data Augmentation - flow_from_directory

Comentario: Aunque el preprocesamiento de los datos ya se haya realizado previamente para poder aplicarlo a aquellos modelos MLP o CNN sin aumentación de datos, se volverá a aplicar ahora el preprocesamiento utilizando ahora flow_from_directory(). Esto tiene como finalidad mostrar técnicas más allá de las vistas en los notebooks de clase y avanzadas que las vistas en clase y explorar metodologías alternativas a flow():

[ ]:

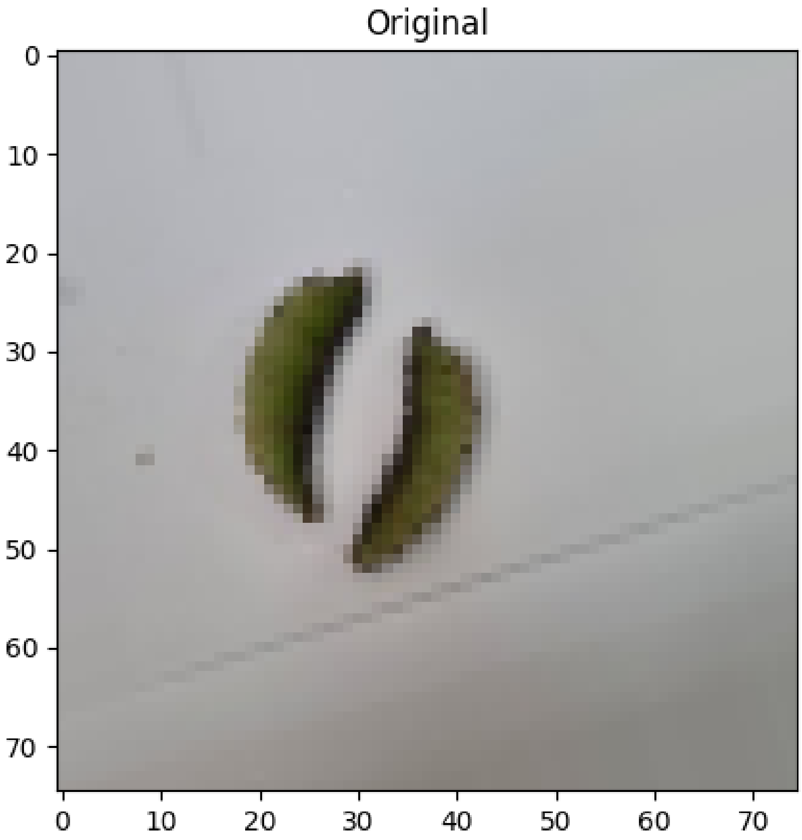
```python
# Definir el tamaño de lote y el tamaño de destino de la imagen como variables
batch_size_aug = 32

# Configuración de las técnicas de augmentation para train
## Train
train_datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    rescale=1./255
)

## Validation
val_datagen = ImageDataGenerator(rescale=1./255)

## Test
test_datagen = ImageDataGenerator(rescale=1./255)


# Definir los generadores de train, validation y test
train_generator = train_datagen.flow_from_directory(
    '/content/my_dataset/Vegetable Images/train',
    target_size=target_size,
    batch_size=batch_size_aug,
    class_mode='categorical',
    shuffle = True,
    seed=42
)

val_generator = val_datagen.flow_from_directory(
    '/content/my_dataset/Vegetable Images/validation',
    target_size=target_size,
    batch_size=batch_size_aug,
    class_mode='categorical',
    shuffle = True,
    seed=42
)

test_generator = test_datagen.flow_from_directory(
    '/content/my_dataset/Vegetable Images/test',
    target_size=target_size,
    batch_size=batch_size_aug,
    class_mode='categorical',
    shuffle = False,
    seed=42
)
```

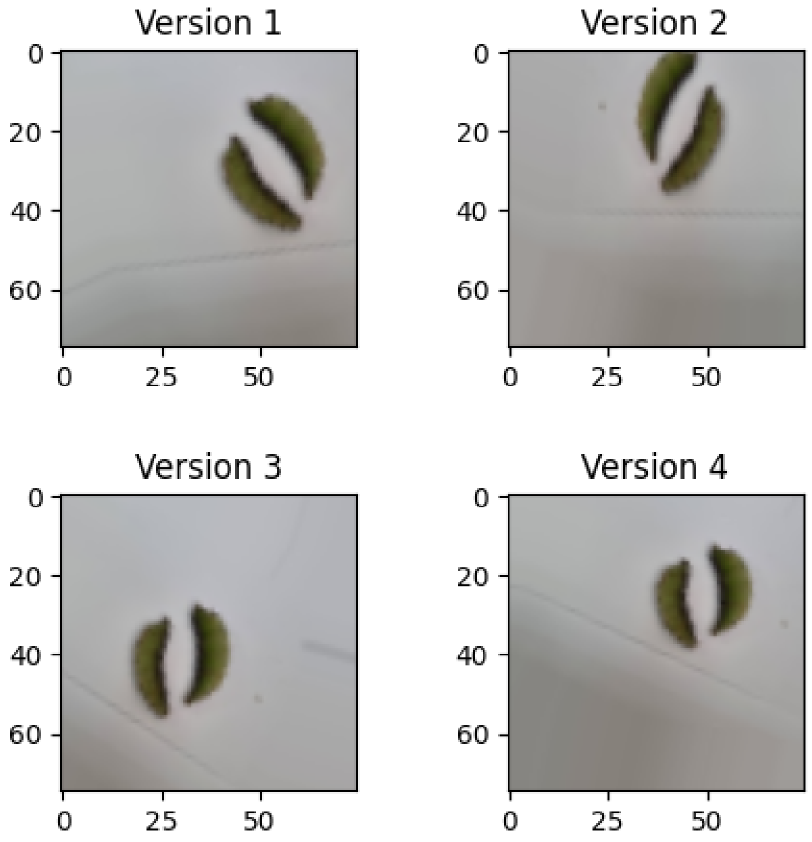
```
Found 15000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
```

Visualizar muestras generadas por data augmentation

[ ]:

```
# Seleccionar batch | almacenar imagen y label
images, labels = train_generator.next()

# Seleccionar imagen aleatoria del batch
random_index = np.random.randint(0, batch_size_aug)
original_image = images[random_index]

# Visualizar imagen original
plt.figure(figsize=(5, 5))
plt.imshow(original_image)
plt.title("Original")

# Visualizar 4 augmentations de la imagen original
fig, axes = plt.subplots(2, 2, figsize=(5, 5))
plt.subplots_adjust(wspace=0.5, hspace=0.5)

i = 0
for _ in range(4):
    augmented_image = train_datagen.random_transform(original_image)
    axes[i // 2, i % 2].imshow(augmented_image)
    axes[i // 2, i % 2].set_title(f'Version {i + 1}')
    i += 1

plt.show()
```


Original
0
10
20
30
40
50
60
70
0
10
20
30
40
50
60
70



#5. Estrategia 1: Entrenar desde cero o from scratch

La primera estrategia a comparar será una red neuronal profunda que el alumno debe diseñar, entrenar y optimizar. Se debe justificar empíricamente las decisiones que llevaron a la selección de la arquitectura e hiperparámetros final. Se espera que el alumno utilice todas las técnicas de regularización mostradas en clase de forma justificada para la mejora del rendimiento de la red neuronal (weight regularization, dropout, batch normalization, data augmentation, etc.).

##5.1 MLP

7.6.1 5.1.1 MLP Base

Construcción modelo MLP sin ningún tipo de normalización.

[ ]:
```
# Función arquitectura MLP base
def get_mlp_model (dense_size, target_size, num_clases):

  model_mlp = Sequential()
  model_mlp.add(Flatten())
  model_mlp.add(Dense(dense_size, input_shape=target_size, activation = 'relu'))
  model_mlp.add(Dense(dense_size//2, activation = 'relu'))
  model_mlp.add(Dense(num_clases, activation = 'softmax'))
  return model_mlp
```

[ ]:

```
# Entrenamiento MLP base

mlp_exp = "MLP_BASE_" + str(batch_size) + "_" + str(epochs)
exp_set.add(mlp_exp)

if do_training == True:

  model_mlp = get_mlp_model(dense_size, target_size, num_clases)

  model_mlp.compile(optimizer=Adam(learning_rate=learning_rate),
                    loss="categorical_crossentropy",
                    metrics=["accuracy"])

  print("[INFO]: Entrenando la red neuronal....")

  H = model_mlp.fit(x_train_norm, y_train_ohe,
                    batch_size=batch_size,
                    epochs=epochs,
                    steps_per_epoch=x_train_norm.shape[0] // batch_size,
                    validation_data=(x_val_norm, y_val_ohe),
                    callbacks=[early_stopping_cbck])

  save_trained_model(model = model_mlp, history = H, model_name = mlp_exp)

else:
    print("[INFO]: Cargando la red neuronal entrenada....")
    model_mlp = load_keras_model(mlp_exp)
    model_mlp.summary()
    H = load_history(mlp_exp)
```

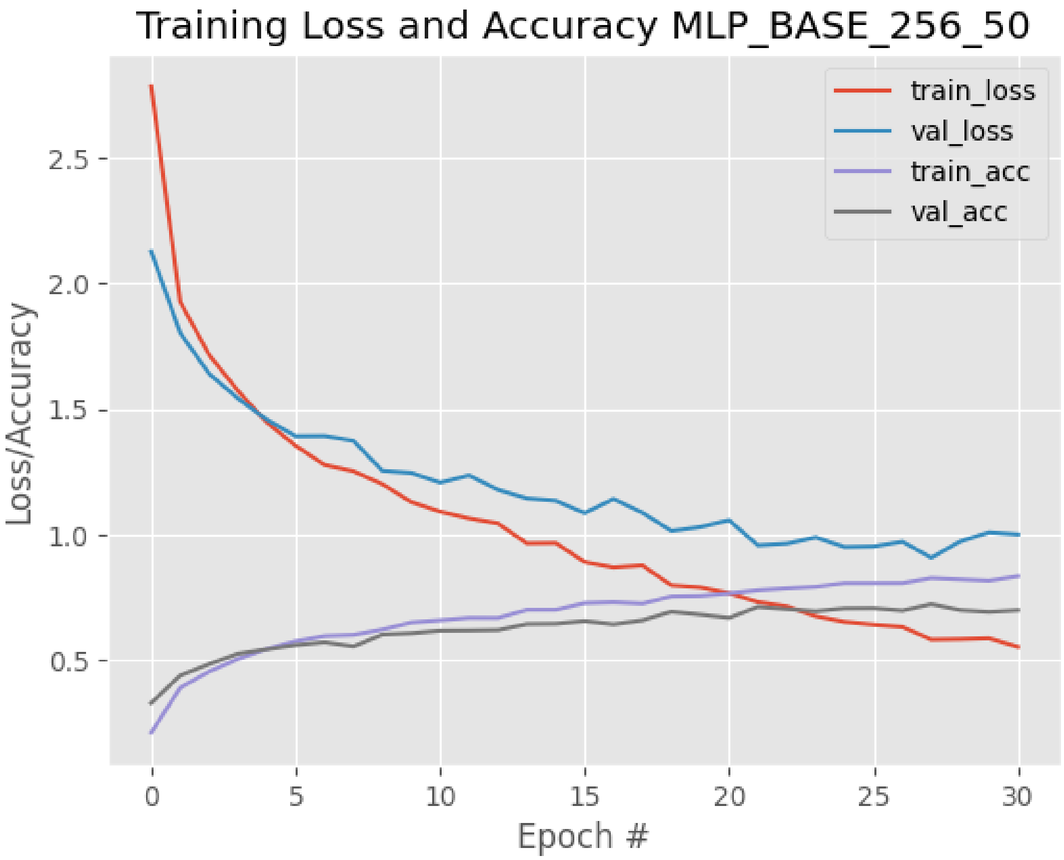
```
[INFO]: Entrenando la red neuronal…
Epoch 1/50
58/58 [==============================] - 3s 28ms/step - loss: 2.7840 - accuracy:
0.2115 - val_loss: 2.1267 - val_accuracy: 0.3297
Epoch 2/50
58/58 [==============================] - 1s 16ms/step - loss: 1.9268 - accuracy:
0.3897 - val_loss: 1.8028 - val_accuracy: 0.4387
Epoch 3/50
58/58 [==============================] - 1s 21ms/step - loss: 1.7158 - accuracy:
0.4544 - val_loss: 1.6404 - val_accuracy: 0.4843
Epoch 4/50
58/58 [==============================] - 1s 21ms/step - loss: 1.5722 - accuracy:
0.5045 - val_loss: 1.5419 - val_accuracy: 0.5250
Epoch 5/50
```

```
58/58 [==============================] - 1s 19ms/step - loss: 1.4464 - accuracy:
0.5443 - val_loss: 1.4571 - val_accuracy: 0.5430
Epoch 6/50
58/58 [==============================] - 1s 22ms/step - loss: 1.3533 - accuracy:
0.5750 - val_loss: 1.3917 - val_accuracy: 0.5597
Epoch 7/50
58/58 [==============================] - 1s 21ms/step - loss: 1.2771 - accuracy:
0.5961 - val_loss: 1.3924 - val_accuracy: 0.5710
Epoch 8/50
58/58 [==============================] - 1s 18ms/step - loss: 1.2518 - accuracy:
0.6002 - val_loss: 1.3732 - val_accuracy: 0.5543
Epoch 9/50
58/58 [==============================] - 1s 16ms/step - loss: 1.2005 - accuracy:
0.6220 - val_loss: 1.2535 - val_accuracy: 0.6017
Epoch 10/50
58/58 [==============================] - 1s 16ms/step - loss: 1.1295 - accuracy:
0.6497 - val_loss: 1.2450 - val_accuracy: 0.6067
Epoch 11/50
58/58 [==============================] - 1s 16ms/step - loss: 1.0912 - accuracy:
0.6584 - val_loss: 1.2070 - val_accuracy: 0.6173
Epoch 12/50
58/58 [==============================] - 1s 17ms/step - loss: 1.0642 - accuracy:
0.6678 - val_loss: 1.2356 - val_accuracy: 0.6180
Epoch 13/50
58/58 [==============================] - 1s 16ms/step - loss: 1.0440 - accuracy:
0.6676 - val_loss: 1.1782 - val_accuracy: 0.6197
Epoch 14/50
58/58 [==============================] - 1s 16ms/step - loss: 0.9650 - accuracy:
0.7001 - val_loss: 1.1439 - val_accuracy: 0.6437
Epoch 15/50
58/58 [==============================] - 1s 17ms/step - loss: 0.9661 - accuracy:
0.7001 - val_loss: 1.1347 - val_accuracy: 0.6447
Epoch 16/50
58/58 [==============================] - 1s 16ms/step - loss: 0.8909 - accuracy:
0.7273 - val_loss: 1.0855 - val_accuracy: 0.6557
Epoch 17/50
58/58 [==============================] - 1s 16ms/step - loss: 0.8689 - accuracy:
0.7308 - val_loss: 1.1416 - val_accuracy: 0.6423
Epoch 18/50
58/58 [==============================] - 1s 17ms/step - loss: 0.8771 - accuracy:
0.7252 - val_loss: 1.0870 - val_accuracy: 0.6583
Epoch 19/50
58/58 [==============================] - 1s 19ms/step - loss: 0.7977 - accuracy:
0.7537 - val_loss: 1.0144 - val_accuracy: 0.6930
Epoch 20/50
58/58 [==============================] - 1s 18ms/step - loss: 0.7895 - accuracy:
0.7556 - val_loss: 1.0298 - val_accuracy: 0.6813
Epoch 21/50
```

```
58/58 [==============================] - 1s 20ms/step - loss: 0.7663 - accuracy:
0.7654 - val_loss: 1.0566 - val_accuracy: 0.6680
Epoch 22/50
58/58 [==============================] - 1s 19ms/step - loss: 0.7315 - accuracy:
0.7781 - val_loss: 0.9570 - val_accuracy: 0.7123
Epoch 23/50
58/58 [==============================] - 1s 21ms/step - loss: 0.7142 - accuracy:
0.7854 - val_loss: 0.9639 - val_accuracy: 0.7030
Epoch 24/50
58/58 [==============================] - 1s 19ms/step - loss: 0.6745 - accuracy:
0.7917 - val_loss: 0.9889 - val_accuracy: 0.6947
Epoch 25/50
58/58 [==============================] - 1s 16ms/step - loss: 0.6522 - accuracy:
0.8061 - val_loss: 0.9501 - val_accuracy: 0.7060
Epoch 26/50
58/58 [==============================] - 1s 16ms/step - loss: 0.6413 - accuracy:
0.8061 - val_loss: 0.9523 - val_accuracy: 0.7067
Epoch 27/50
58/58 [==============================] - 1s 16ms/step - loss: 0.6323 - accuracy:
0.8064 - val_loss: 0.9715 - val_accuracy: 0.6970
Epoch 28/50
58/58 [==============================] - 1s 16ms/step - loss: 0.5818 - accuracy:
0.8266 - val_loss: 0.9080 - val_accuracy: 0.7230
Epoch 29/50
58/58 [==============================] - 1s 17ms/step - loss: 0.5836 - accuracy:
0.8215 - val_loss: 0.9725 - val_accuracy: 0.6990
Epoch 30/50
58/58 [==============================] - 1s 17ms/step - loss: 0.5869 - accuracy:
0.8161 - val_loss: 1.0085 - val_accuracy: 0.6920
Epoch 31/50
58/58 [==============================] - 1s 17ms/step - loss: 0.5525 - accuracy:
0.8343 - val_loss: 1.0003 - val_accuracy: 0.6987
Saved model to disk
```

[ ]:

```
visualize_learning_curve(H, lb = mlp_exp)
```



```
[ ]: # Evaluación MLP base

     evaluate_model(model_mlp, x_test_norm, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 0s 6ms/step
              precision    recall  f1-score   support

           0       0.73      0.67      0.70       200
           1       0.68      0.56      0.62       200
           2       0.81      0.75      0.78       200
           3       0.77      0.85      0.81       200
           4       0.65      0.89      0.75       200
           5       0.79      0.61      0.69       200
           6       0.82      0.84      0.83       200
           7       0.44      0.72      0.55       200
           8       0.71      0.66      0.68       200
           9       0.70      0.37      0.49       200
          10       0.80      0.80      0.80       200
          11       0.87      0.93      0.90       200
          12       0.83      0.41      0.54       200
          13       0.68      0.58      0.63       200
          14       0.51      0.77      0.61       200

    accuracy                           0.69      3000
   macro avg       0.72      0.69      0.69      3000
weighted avg       0.72      0.69      0.69      3000
```

7.6.2 5.1.2 MLP Regularizado

a) Dropout

[ ]:

```
# Función arquitectura MLP regularizado

def get_mlp_model_drop(dense_size, target_size, num_clases, dropout_rate=0.25):

    model_mlp_drop = Sequential()
    model_mlp_drop.add(Flatten())
    model_mlp_drop.add(Dense(dense_size, input_shape=target_size, ␣
 ↪activation='relu'))
    model_mlp_drop.add(Dropout(dropout_rate))  # 1er Dropout
    model_mlp_drop.add(Dense(dense_size // 2, activation='relu'))
    model_mlp_drop.add(Dropout(dropout_rate))  # 2do  Dropout
    model_mlp_drop.add(Dense(num_clases, activation='softmax'))

    return model_mlp_drop
```

[ ]:

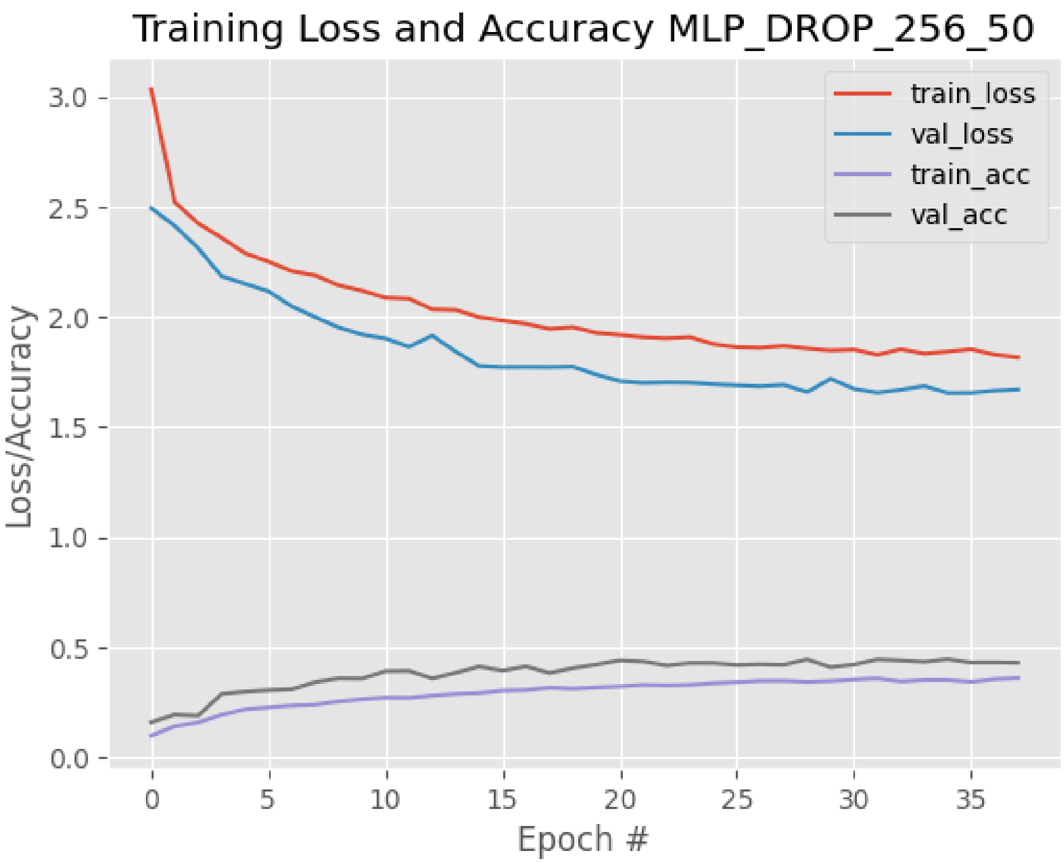
```
# Entrenamiento MLP con dropout

mlp_drop_exp = "MLP_DROP_" + str(batch_size) + "_" + str(epochs)
exp_set.add(mlp_drop_exp)

if do_training == True:

  mlp_model_drop = get_mlp_model_drop(dense_size, target_size, num_clases)

  mlp_model_drop.compile(optimizer=Adam(learning_rate=learning_rate),
                         loss="categorical_crossentropy",
                         metrics=["accuracy"])

  print("[INFO]: Entrenando la red neuronal con dropout....")

  H_drop = mlp_model_drop.fit(x_train_norm, y_train_ohe, batch_size=batch_size,
                              epochs=epochs,
                              steps_per_epoch=x_train_norm.shape[0] // batch_size,
                              validation_data=(x_val_norm, y_val_ohe),
                              callbacks=[early_stopping_cbck])
  save_trained_model(model = mlp_model_drop, history = H_drop, model_name = mlp_drop_exp)

else:
    print("[INFO]: Cargando la red neuronal entrenada con dropout....")
    mlp_model_drop = load_keras_model(mlp_drop_exp)
    mlp_model_drop.summary()
    H_drop = load_history(mlp_drop_exp)
```

```
[INFO]: Entrenando la red neuronal con dropout…
Epoch 1/50
58/58 [==============================] - 10s 38ms/step - loss: 3.0316 -
accuracy: 0.0988 - val_loss: 2.4926 - val_accuracy: 0.1597
Epoch 2/50
58/58 [==============================] - 1s 16ms/step - loss: 2.5205 - accuracy:
0.1411 - val_loss: 2.4131 - val_accuracy: 0.1933
Epoch 3/50
58/58 [==============================] - 1s 16ms/step - loss: 2.4240 - accuracy:
0.1588 - val_loss: 2.3112 - val_accuracy: 0.1887
Epoch 4/50
58/58 [==============================] - 1s 16ms/step - loss: 2.3586 - accuracy:
0.1926 - val_loss: 2.1839 - val_accuracy: 0.2877
Epoch 5/50
58/58 [==============================] - 1s 16ms/step - loss: 2.2883 - accuracy:
0.2176 - val_loss: 2.1499 - val_accuracy: 0.2977
Epoch 6/50
58/58 [==============================] - 1s 16ms/step - loss: 2.2503 - accuracy:
0.2263 - val_loss: 2.1156 - val_accuracy: 0.3043
Epoch 7/50
58/58 [==============================] - 1s 16ms/step - loss: 2.2067 - accuracy:
0.2349 - val_loss: 2.0467 - val_accuracy: 0.3087
Epoch 8/50
58/58 [==============================] - 1s 16ms/step - loss: 2.1874 - accuracy:
0.2392 - val_loss: 1.9982 - val_accuracy: 0.3413
Epoch 9/50
58/58 [==============================] - 1s 16ms/step - loss: 2.1428 - accuracy:
0.2540 - val_loss: 1.9513 - val_accuracy: 0.3583
Epoch 10/50
58/58 [==============================] - 1s 16ms/step - loss: 2.1183 - accuracy:
0.2634 - val_loss: 1.9190 - val_accuracy: 0.3573
Epoch 11/50
58/58 [==============================] - 1s 16ms/step - loss: 2.0865 - accuracy:
0.2708 - val_loss: 1.9013 - val_accuracy: 0.3910
Epoch 12/50
58/58 [==============================] - 1s 20ms/step - loss: 2.0817 - accuracy:
0.2698 - val_loss: 1.8638 - val_accuracy: 0.3923
```

```
Epoch 13/50
58/58 [==============================] - 1s 22ms/step - loss: 2.0336 - accuracy:
0.2800 - val_loss: 1.9146 - val_accuracy: 0.3580
Epoch 14/50
58/58 [==============================] - 1s 22ms/step - loss: 2.0307 - accuracy:
0.2876 - val_loss: 1.8416 - val_accuracy: 0.3840
Epoch 15/50
58/58 [==============================] - 1s 23ms/step - loss: 1.9981 - accuracy:
0.2916 - val_loss: 1.7763 - val_accuracy: 0.4120
Epoch 16/50
58/58 [==============================] - 1s 22ms/step - loss: 1.9829 - accuracy:
0.3025 - val_loss: 1.7710 - val_accuracy: 0.3937
Epoch 17/50
58/58 [==============================] - 1s 19ms/step - loss: 1.9678 - accuracy:
0.3051 - val_loss: 1.7716 - val_accuracy: 0.4127
Epoch 18/50
58/58 [==============================] - 1s 16ms/step - loss: 1.9450 - accuracy:
0.3157 - val_loss: 1.7709 - val_accuracy: 0.3830
Epoch 19/50
58/58 [==============================] - 1s 17ms/step - loss: 1.9525 - accuracy:
0.3114 - val_loss: 1.7731 - val_accuracy: 0.4050
Epoch 20/50
58/58 [==============================] - 1s 17ms/step - loss: 1.9268 - accuracy:
0.3169 - val_loss: 1.7373 - val_accuracy: 0.4213
Epoch 21/50
58/58 [==============================] - 1s 16ms/step - loss: 1.9178 - accuracy:
0.3213 - val_loss: 1.7082 - val_accuracy: 0.4407
Epoch 22/50
58/58 [==============================] - 1s 16ms/step - loss: 1.9072 - accuracy:
0.3284 - val_loss: 1.6999 - val_accuracy: 0.4353
Epoch 23/50
58/58 [==============================] - 1s 16ms/step - loss: 1.9024 - accuracy:
0.3260 - val_loss: 1.7027 - val_accuracy: 0.4170
Epoch 24/50
58/58 [==============================] - 1s 16ms/step - loss: 1.9072 - accuracy:
0.3287 - val_loss: 1.7011 - val_accuracy: 0.4280
Epoch 25/50
58/58 [==============================] - 1s 16ms/step - loss: 1.8759 - accuracy:
0.3363 - val_loss: 1.6939 - val_accuracy: 0.4277
Epoch 26/50
58/58 [==============================] - 1s 17ms/step - loss: 1.8619 - accuracy:
0.3408 - val_loss: 1.6891 - val_accuracy: 0.4193
Epoch 27/50
58/58 [==============================] - 1s 16ms/step - loss: 1.8593 - accuracy:
0.3464 - val_loss: 1.6846 - val_accuracy: 0.4227
Epoch 28/50
58/58 [==============================] - 1s 22ms/step - loss: 1.8679 - accuracy:
0.3471 - val_loss: 1.6912 - val_accuracy: 0.4200
```

```
Epoch 29/50
58/58 [==============================] - 1s 22ms/step - loss: 1.8563 - accuracy:
0.3420 - val_loss: 1.6580 - val_accuracy: 0.4450
Epoch 30/50
58/58 [==============================] - 1s 22ms/step - loss: 1.8472 - accuracy:
0.3458 - val_loss: 1.7185 - val_accuracy: 0.4090
Epoch 31/50
58/58 [==============================] - 1s 19ms/step - loss: 1.8511 - accuracy:
0.3530 - val_loss: 1.6728 - val_accuracy: 0.4210
Epoch 32/50
58/58 [==============================] - 1s 21ms/step - loss: 1.8273 - accuracy:
0.3583 - val_loss: 1.6549 - val_accuracy: 0.4447
Epoch 33/50
58/58 [==============================] - 1s 18ms/step - loss: 1.8524 - accuracy:
0.3440 - val_loss: 1.6685 - val_accuracy: 0.4400
Epoch 34/50
58/58 [==============================] - 1s 16ms/step - loss: 1.8328 - accuracy:
0.3509 - val_loss: 1.6856 - val_accuracy: 0.4340
Epoch 35/50
58/58 [==============================] - 1s 19ms/step - loss: 1.8417 - accuracy:
0.3511 - val_loss: 1.6539 - val_accuracy: 0.4457
Epoch 36/50
58/58 [==============================] - 1s 16ms/step - loss: 1.8527 - accuracy:
0.3424 - val_loss: 1.6543 - val_accuracy: 0.4303
Epoch 37/50
58/58 [==============================] - 1s 16ms/step - loss: 1.8278 - accuracy:
0.3545 - val_loss: 1.6642 - val_accuracy: 0.4310
Epoch 38/50
58/58 [==============================] - 1s 16ms/step - loss: 1.8158 - accuracy:
0.3594 - val_loss: 1.6698 - val_accuracy: 0.4290
Saved model to disk
```

```
[ ]: visualize_learning_curve(H_drop, lb = mlp_drop_exp)
```



```
[ ]: # Evaluación MLP dropout

     evaluate_model(mlp_model_drop, x_test_norm, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 0s 5ms/step
              precision    recall  f1-score   support

           0       0.25      0.17      0.20       200
           1       0.41      0.56      0.47       200
           2       0.53      0.65      0.58       200
           3       0.63      0.69      0.66       200
           4       0.79      0.47      0.59       200
           5       0.37      0.86      0.51       200
           6       0.38      0.41      0.40       200
           7       0.38      0.15      0.22       200
           8       0.22      0.59      0.33       200
           9       0.21      0.14      0.17       200
          10       0.78      0.40      0.53       200
          11       0.86      0.70      0.77       200
          12       0.29      0.37      0.33       200
          13       0.56      0.16      0.25       200
          14       0.20      0.01      0.03       200

    accuracy                           0.42      3000
   macro avg       0.46      0.42      0.40      3000
weighted avg       0.46      0.42      0.40      3000
```

b) Regularización L1 o L2

[ ]:

```
# Función arquitectura MLP regularizado

def get_mlp_reg_model(dense_size, target_size, num_clases, drop_out_prop=0.05,␣
 ↪reg_type='l1', reg_value=0.001):

    model_mlp_reg = Sequential()
    model_mlp_reg.add(Flatten())

    if reg_type == 'l2':
        reg = regularizers.l2(reg_value)
    elif reg_type == 'l1':
        reg = regularizers.l1(reg_value)
    else:
        raise ValueError("Input de regularización no válido")

    model_mlp_reg.add(Dense(dense_size, input_shape=target_size,␣
 ↪kernel_regularizer=reg, activation='relu'))
    model_mlp_reg.add(Dense(dense_size // 2, kernel_regularizer=reg,␣
 ↪activation='relu'))
    model_mlp_reg.add(Dropout(drop_out_prop))
    model_mlp_reg.add(Dense(num_clases, activation='softmax'))

    return model_mlp_reg
```

[ ]:

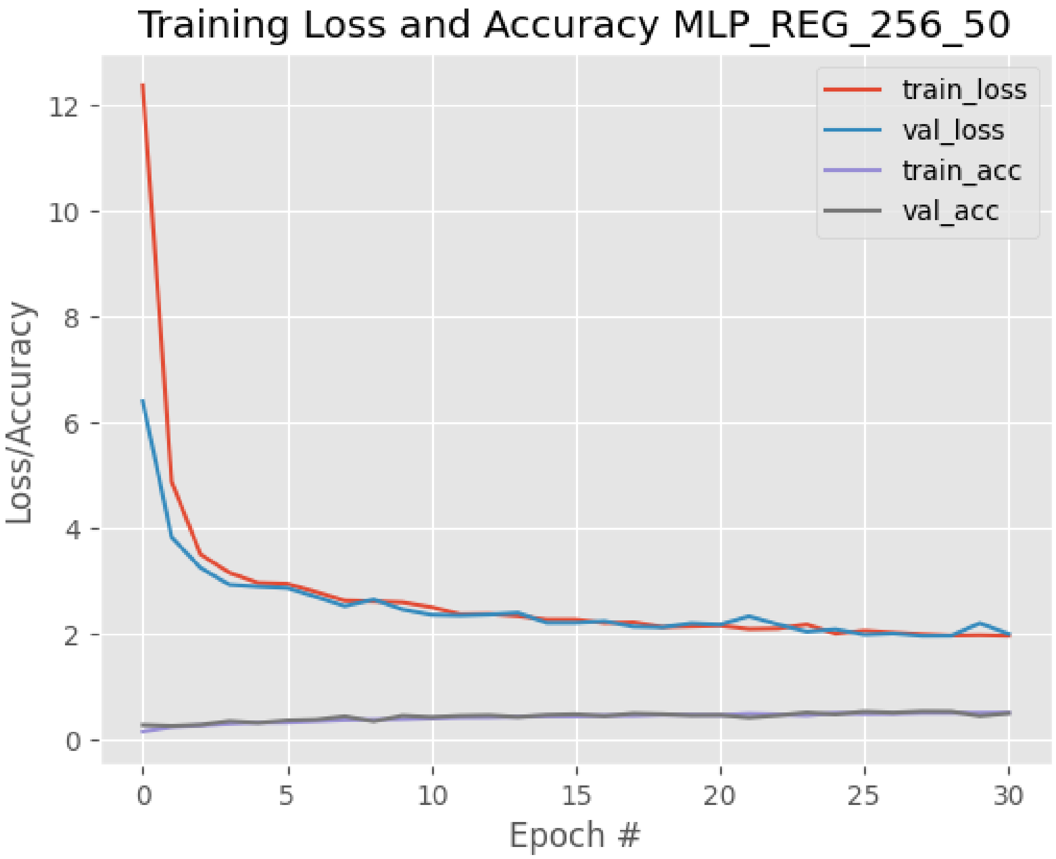
```
# Entrenamiento MLP regularizado

mlp_reg_exp = "MLP_REG_" + str(batch_size) + "_" + str(epochs)
exp_set.add(mlp_reg_exp)

if do_training == True:

  mlp_model_reg = get_mlp_reg_model(dense_size, target_size, num_clases)

  mlp_model_reg.compile(optimizer=Adam(learning_rate=learning_rate),
                         loss="categorical_crossentropy",
                         metrics=["accuracy"])
  print("[INFO]: Entrenando la red neuronal regularizada....")

  H_reg = mlp_model_reg.fit(x_train_norm, y_train_ohe, batch_size=batch_size,
                            epochs=epochs,
                            steps_per_epoch=x_train_norm.shape[0] // batch_size,
                            validation_data=(x_val_norm, y_val_ohe),
                            callbacks=[early_stopping_cbck])

  save_trained_model(model = mlp_model_reg, history = H_reg, model_name =␣
 ↪mlp_reg_exp)

else:
    print("[INFO]: Cargando la red neuronal regularizada....")
    mlp_model_reg = load_keras_model(mlp_reg_exp)
    mlp_model_reg.summary()
    H_reg = load_history(mlp_reg_exp)
```

```
[INFO]: Entrenando la red neuronal regularizada…
Epoch 1/50
58/58 [==============================] - 4s 38ms/step - loss: 12.3719 -
accuracy: 0.1636 - val_loss: 6.4086 - val_accuracy: 0.2860
Epoch 2/50
58/58 [==============================] - 1s 21ms/step - loss: 4.8901 - accuracy:
0.2539 - val_loss: 3.8355 - val_accuracy: 0.2677
Epoch 3/50
58/58 [==============================] - 1s 20ms/step - loss: 3.5111 - accuracy:
0.2790 - val_loss: 3.2588 - val_accuracy: 0.2897
Epoch 4/50
58/58 [==============================] - 1s 20ms/step - loss: 3.1670 - accuracy:
0.3129 - val_loss: 2.9379 - val_accuracy: 0.3623
Epoch 5/50
58/58 [==============================] - 1s 16ms/step - loss: 2.9711 - accuracy:
0.3340 - val_loss: 2.9019 - val_accuracy: 0.3290
Epoch 6/50
58/58 [==============================] - 1s 16ms/step - loss: 2.9548 - accuracy:
0.3370 - val_loss: 2.8766 - val_accuracy: 0.3760
Epoch 7/50
58/58 [==============================] - 1s 17ms/step - loss: 2.8022 - accuracy:
0.3590 - val_loss: 2.7110 - val_accuracy: 0.3897
Epoch 8/50
58/58 [==============================] - 1s 16ms/step - loss: 2.6390 - accuracy:
0.3826 - val_loss: 2.5352 - val_accuracy: 0.4523
Epoch 9/50
58/58 [==============================] - 1s 17ms/step - loss: 2.6252 - accuracy:
0.3916 - val_loss: 2.6601 - val_accuracy: 0.3583
Epoch 10/50
58/58 [==============================] - 1s 17ms/step - loss: 2.6057 - accuracy:
```

```
0.3999 - val_loss: 2.4700 - val_accuracy: 0.4643
Epoch 11/50
58/58 [==============================] - 1s 16ms/step - loss: 2.5140 - accuracy:
0.4169 - val_loss: 2.3704 - val_accuracy: 0.4377
Epoch 12/50
58/58 [==============================] - 1s 16ms/step - loss: 2.3797 - accuracy:
0.4327 - val_loss: 2.3584 - val_accuracy: 0.4623
Epoch 13/50
58/58 [==============================] - 1s 17ms/step - loss: 2.3860 - accuracy:
0.4355 - val_loss: 2.3749 - val_accuracy: 0.4713
Epoch 14/50
58/58 [==============================] - 1s 17ms/step - loss: 2.3475 - accuracy:
0.4474 - val_loss: 2.4065 - val_accuracy: 0.4407
Epoch 15/50
58/58 [==============================] - 1s 20ms/step - loss: 2.2808 - accuracy:
0.4568 - val_loss: 2.2187 - val_accuracy: 0.4773
Epoch 16/50
58/58 [==============================] - 1s 24ms/step - loss: 2.2810 - accuracy:
0.4542 - val_loss: 2.2194 - val_accuracy: 0.4910
Epoch 17/50
58/58 [==============================] - 1s 21ms/step - loss: 2.2160 - accuracy:
0.4684 - val_loss: 2.2494 - val_accuracy: 0.4587
Epoch 18/50
58/58 [==============================] - 1s 21ms/step - loss: 2.2300 - accuracy:
0.4655 - val_loss: 2.1485 - val_accuracy: 0.5037
Epoch 19/50
58/58 [==============================] - 1s 23ms/step - loss: 2.1396 - accuracy:
0.4821 - val_loss: 2.1355 - val_accuracy: 0.4917
Epoch 20/50
58/58 [==============================] - 1s 19ms/step - loss: 2.1550 - accuracy:
0.4819 - val_loss: 2.2101 - val_accuracy: 0.4707
Epoch 21/50
58/58 [==============================] - 1s 17ms/step - loss: 2.1689 - accuracy:
0.4741 - val_loss: 2.1868 - val_accuracy: 0.4763
Epoch 22/50
58/58 [==============================] - 1s 20ms/step - loss: 2.1020 - accuracy:
0.4992 - val_loss: 2.3428 - val_accuracy: 0.4297
Epoch 23/50
58/58 [==============================] - 2s 28ms/step - loss: 2.1115 - accuracy:
0.4864 - val_loss: 2.1900 - val_accuracy: 0.4683
Epoch 24/50
58/58 [==============================] - 1s 25ms/step - loss: 2.1888 - accuracy:
0.4704 - val_loss: 2.0521 - val_accuracy: 0.5220
Epoch 25/50
58/58 [==============================] - 1s 26ms/step - loss: 2.0112 - accuracy:
0.5133 - val_loss: 2.0957 - val_accuracy: 0.4900
Epoch 26/50
58/58 [==============================] - 1s 26ms/step - loss: 2.0685 - accuracy:
```

```
0.4993 - val_loss: 1.9945 - val_accuracy: 0.5390
Epoch 27/50
58/58 [==============================] - 1s 24ms/step - loss: 2.0330 - accuracy:
0.5021 - val_loss: 2.0151 - val_accuracy: 0.5187
Epoch 28/50
58/58 [==============================] - 2s 30ms/step - loss: 2.0019 - accuracy:
0.5159 - val_loss: 1.9731 - val_accuracy: 0.5433
Epoch 29/50
58/58 [==============================] - 2s 29ms/step - loss: 1.9788 - accuracy:
0.5159 - val_loss: 1.9743 - val_accuracy: 0.5403
Epoch 30/50
58/58 [==============================] - 2s 30ms/step - loss: 1.9846 - accuracy:
0.5159 - val_loss: 2.2117 - val_accuracy: 0.4603
Epoch 31/50
58/58 [==============================] - 2s 34ms/step - loss: 1.9742 - accuracy:
0.5208 - val_loss: 2.0086 - val_accuracy: 0.5073
Saved model to disk
```

```
[ ]: visualize_learning_curve(H_reg, lb = mlp_reg_exp)
```



[ ]:

```
## Evaluación MLP regularizado
evaluate_model(mlp_model_reg, x_test_norm, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 0s 6ms/step
              precision    recall  f1-score   support

           0       0.30      0.46      0.37       200
           1       0.59      0.56      0.58       200
           2       0.77      0.52      0.62       200
           3       0.59      0.55      0.57       200
           4       0.36      0.70      0.48       200
           5       0.74      0.47      0.57       200
           6       0.58      0.58      0.58       200
           7       0.37      0.24      0.29       200
           8       0.53      0.36      0.43       200
           9       0.36      0.18      0.24       200
          10       0.71      0.77      0.74       200
          11       0.77      0.86      0.82       200
          12       0.43      0.37      0.40       200
          13       0.44      0.48      0.46       200
          14       0.39      0.52      0.44       200

    accuracy                           0.51      3000
   macro avg       0.53      0.51      0.51      3000
weighted avg       0.53      0.51      0.51      3000
```

Comentario resultados regularización L1 y L2: Con regularización *L2* el modelo no mejora (L2_regularization_accuracy = 0.69 ; *non_regularization_score* = 0.69). Con regularización *L1* el modelo empeora (*L1_regularization_accuracy* = 0.51 ; *non_regularization_score* = 0.69).

Se han probado varios valores de regularización como 0.0001, 0.001, 0.009, 0.01 Se han encontrado que 0.001 devolvía los mejores resultados.

c) Data Augmentation Entrenar modelo de perceptrón multicapa (MLP) regularizado utilizando técnicas de aumentación de datos (data augmentation).

[ ]:

```
# Entrenamiento MLP con data augmentation

mlp_aug_exp = "MLP_AUG_" + str(batch_size) + "_" + str(epochs)
exp_set.add(mlp_aug_exp)

if do_training == True:

  model_mlp_aug = get_mlp_model(dense_size, target_size, num_clases)

  model_mlp_aug.compile(optimizer=Adam(learning_rate=learning_rate),
                          loss="categorical_crossentropy",
                          metrics=["accuracy"])

  print("[INFO]: Entrenando la red neuronal con data augmentation...")

  H_aug = model_mlp_aug.fit(train_generator,
                            steps_per_epoch=train_generator.samples //␣
 ↪batch_size_aug,
                            validation_data=val_generator,
                            validation_steps=val_generator.samples //␣
 ↪batch_size_aug,
                            epochs=epochs,
                            callbacks=[early_stopping_cbck])

  save_trained_model(model = model_mlp_aug, history = H_aug, model_name =␣
 ↪mlp_aug_exp)

else:
  print("[INFO]: Cargando la red neuronal aumentada regularizada entrenada....")
  model_mlp_aug = load_keras_model(mlp_aug_exp)
  model_mlp_aug.summary()
  H_aug = load_history(mlp_aug_exp)
```

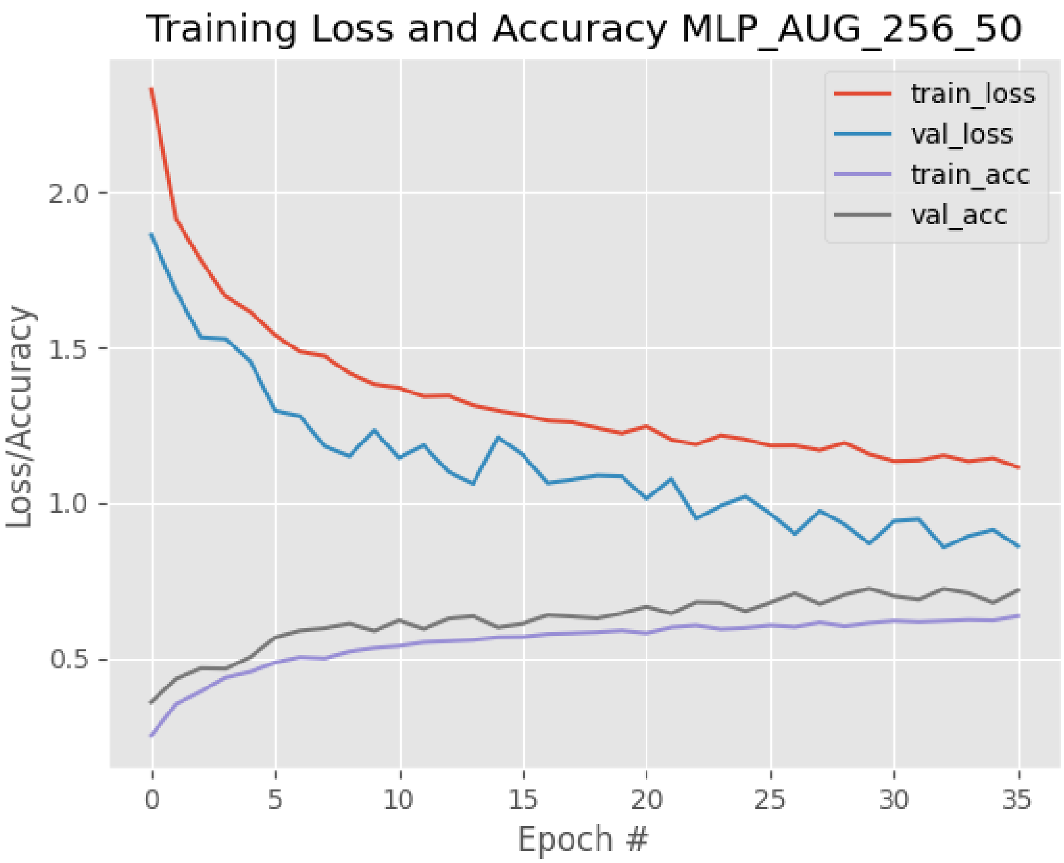
```
[INFO]: Entrenando la red neuronal con data augmentation…
Epoch 1/50
468/468 [==============================] - 55s 115ms/step - loss: 2.3297 -
accuracy: 0.2503 - val_loss: 1.8618 - val_accuracy: 0.3582
Epoch 2/50
468/468 [==============================] - 53s 114ms/step - loss: 1.9133 -
accuracy: 0.3520 - val_loss: 1.6799 - val_accuracy: 0.4331
Epoch 3/50
468/468 [==============================] - 54s 114ms/step - loss: 1.7804 -
accuracy: 0.3924 - val_loss: 1.5322 - val_accuracy: 0.4671
Epoch 4/50
468/468 [==============================] - 51s 108ms/step - loss: 1.6633 -
accuracy: 0.4374 - val_loss: 1.5268 - val_accuracy: 0.4657
Epoch 5/50
468/468 [==============================] - 52s 111ms/step - loss: 1.6143 -
accuracy: 0.4560 - val_loss: 1.4553 - val_accuracy: 0.5027
Epoch 6/50
468/468 [==============================] - 52s 111ms/step - loss: 1.5391 -
accuracy: 0.4855 - val_loss: 1.2962 - val_accuracy: 0.5662
Epoch 7/50
468/468 [==============================] - 51s 109ms/step - loss: 1.4861 -
accuracy: 0.5022 - val_loss: 1.2787 - val_accuracy: 0.5880
Epoch 8/50
468/468 [==============================] - 59s 126ms/step - loss: 1.4720 -
```

```
accuracy: 0.4987 - val_loss: 1.1816 - val_accuracy: 0.5961
Epoch 9/50
468/468 [==============================] - 72s 154ms/step - loss: 1.4163 -
accuracy: 0.5207 - val_loss: 1.1496 - val_accuracy: 0.6106
Epoch 10/50
468/468 [==============================] - 54s 116ms/step - loss: 1.3808 -
accuracy: 0.5324 - val_loss: 1.2329 - val_accuracy: 0.5874
Epoch 11/50
468/468 [==============================] - 71s 153ms/step - loss: 1.3696 -
accuracy: 0.5387 - val_loss: 1.1440 - val_accuracy: 0.6210
Epoch 12/50
468/468 [==============================] - 74s 158ms/step - loss: 1.3416 -
accuracy: 0.5508 - val_loss: 1.1845 - val_accuracy: 0.5938
Epoch 13/50
468/468 [==============================] - 75s 160ms/step - loss: 1.3437 -
accuracy: 0.5542 - val_loss: 1.0990 - val_accuracy: 0.6267
Epoch 14/50
468/468 [==============================] - 70s 149ms/step - loss: 1.3132 -
accuracy: 0.5581 - val_loss: 1.0606 - val_accuracy: 0.6344
Epoch 15/50
468/468 [==============================] - 61s 131ms/step - loss: 1.2964 -
accuracy: 0.5667 - val_loss: 1.2112 - val_accuracy: 0.5985
Epoch 16/50
468/468 [==============================] - 71s 152ms/step - loss: 1.2820 -
accuracy: 0.5677 - val_loss: 1.1541 - val_accuracy: 0.6106
Epoch 17/50
468/468 [==============================] - 80s 171ms/step - loss: 1.2637 -
accuracy: 0.5771 - val_loss: 1.0629 - val_accuracy: 0.6381
Epoch 18/50
468/468 [==============================] - 58s 123ms/step - loss: 1.2587 -
accuracy: 0.5796 - val_loss: 1.0740 - val_accuracy: 0.6334
Epoch 19/50
468/468 [==============================] - 64s 138ms/step - loss: 1.2405 -
accuracy: 0.5831 - val_loss: 1.0874 - val_accuracy: 0.6274
Epoch 20/50
468/468 [==============================] - 57s 122ms/step - loss: 1.2232 -
accuracy: 0.5885 - val_loss: 1.0846 - val_accuracy: 0.6442
Epoch 21/50
468/468 [==============================] - 61s 130ms/step - loss: 1.2458 -
accuracy: 0.5799 - val_loss: 1.0117 - val_accuracy: 0.6657
Epoch 22/50
468/468 [==============================] - 55s 118ms/step - loss: 1.2029 -
accuracy: 0.5991 - val_loss: 1.0772 - val_accuracy: 0.6428
Epoch 23/50
468/468 [==============================] - 65s 140ms/step - loss: 1.1869 -
accuracy: 0.6054 - val_loss: 0.9476 - val_accuracy: 0.6798
Epoch 24/50
468/468 [==============================] - 50s 107ms/step - loss: 1.2167 -
```

```
accuracy: 0.5930 - val_loss: 0.9899 - val_accuracy: 0.6778
Epoch 25/50
468/468 [==============================] - 49s 105ms/step - loss: 1.2037 -
accuracy: 0.5972 - val_loss: 1.0201 - val_accuracy: 0.6505
Epoch 26/50
468/468 [==============================] - 50s 107ms/step - loss: 1.1830 -
accuracy: 0.6054 - val_loss: 0.9645 - val_accuracy: 0.6784
Epoch 27/50
468/468 [==============================] - 51s 108ms/step - loss: 1.1836 -
accuracy: 0.6002 - val_loss: 0.8992 - val_accuracy: 0.7077
Epoch 28/50
468/468 [==============================] - 49s 104ms/step - loss: 1.1685 -
accuracy: 0.6145 - val_loss: 0.9739 - val_accuracy: 0.6737
Epoch 29/50
468/468 [==============================] - 50s 107ms/step - loss: 1.1926 -
accuracy: 0.6022 - val_loss: 0.9297 - val_accuracy: 0.7033
Epoch 30/50
468/468 [==============================] - 52s 111ms/step - loss: 1.1561 -
accuracy: 0.6125 - val_loss: 0.8684 - val_accuracy: 0.7235
Epoch 31/50
468/468 [==============================] - 56s 119ms/step - loss: 1.1336 -
accuracy: 0.6195 - val_loss: 0.9404 - val_accuracy: 0.6989
Epoch 32/50
468/468 [==============================] - 69s 148ms/step - loss: 1.1358 -
accuracy: 0.6158 - val_loss: 0.9461 - val_accuracy: 0.6878
Epoch 33/50
468/468 [==============================] - 71s 153ms/step - loss: 1.1520 -
accuracy: 0.6193 - val_loss: 0.8557 - val_accuracy: 0.7228
Epoch 34/50
468/468 [==============================] - 69s 147ms/step - loss: 1.1331 -
accuracy: 0.6226 - val_loss: 0.8923 - val_accuracy: 0.7083
Epoch 35/50
468/468 [==============================] - 70s 150ms/step - loss: 1.1430 -
accuracy: 0.6209 - val_loss: 0.9128 - val_accuracy: 0.6781
Epoch 36/50
468/468 [==============================] - 78s 167ms/step - loss: 1.1137 -
accuracy: 0.6350 - val_loss: 0.8600 - val_accuracy: 0.7174
Saved model to disk
```

[ ]:

```
visualize_learning_curve(H_aug, lb = mlp_aug_exp)
```



```
[ ]: # Evaluación MLP con data augmentation
     evaluate_model_aug(model_mlp_aug, test_generator)
```

```
[INFO]: Evaluando red neuronal…

<ipython-input-4-e54c28b45500>:27: UserWarning: `Model.predict_generator` is
deprecated and will be removed in a future version. Please use `Model.predict`,
which supports generators.
  predictions = model.predict_generator(generator, steps=len(generator))
              precision    recall  f1-score   support

           0       0.73      0.36      0.49       200
           1       0.54      0.77      0.64       200
           2       0.55      0.92      0.69       200
           3       0.79      0.69      0.73       200
           4       0.54      0.94      0.68       200
           5       0.72      0.58      0.64       200
           6       0.86      0.83      0.84       200
           7       0.89      0.99      0.94       200
           8       0.82      0.47      0.60       200
           9       0.77      0.46      0.58       200
          10       0.88      0.62      0.73       200
          11       0.87      0.82      0.85       200
          12       0.74      0.74      0.74       200
          13       0.74      0.89      0.80       200
          14       0.75      0.69      0.72       200

    accuracy                           0.72      3000
   macro avg       0.75      0.72      0.71      3000
weighted avg       0.75      0.72      0.71      3000
```

Comentario: Al aplicar data augmentation al MLP, se observa una convergencia más irregular debido a la mayor variabilidad de las imágenes introducida por esta técnica. A nivel global, se aprecia un incremento en la precisión (accuracy) tanto en los conjuntos de entrenamiento (train) como de validación. Asimismo, se constata una disminución en el error de pérdidas tanto en las fases de entrenamiento como de validación. Estos cambios indican una mejora potencial en la capacidad del modelo para generalizar, beneficiándose de un conjunto de datos más diverso y representativo.

##**5.2 CNNs**

###**5.2.1. Función arquitectura CNN**

Comentario: Obsérvese que la regularización `batch normalization` es uno de los parámetros de del modelo. La intención era hacer el modelo lo más funcional posible. Se han explorado también las regularizaciones `dropout`, `L1` y `L2`. Sin embargo, ninguna de ellas aplicaba mejoras significativas a la perfomance. Cómo su implementación ya se ha mostrado en en el apartado **5.1)**, este apartado se ceñirá únicamente a la exploración del incremento de performance al añadir bloques convolucionales y `data augmentation` con `flow_from_directory`.

```
[ ]: # Función arquitectura de red CNN: soporta de 1 a 3 bloques convolucionales 2D␣
      ↪simples con la opción de batch normalization

     def get_cnn_model(blocks, x_train, num_clases, dense_size, batch_norm = False):

       input = layers.Input(shape=(x_train.shape[1], x_train.shape[2], x_train.
      ↪shape[3]))

       # Bloque 1
       x1 = layers.Conv2D(conv2d_size, (3,3), padding="same",␣
      ↪activation="relu")(input)
       if batch_norm: x1 = layers.BatchNormalization()(x1)
       x1 = layers.MaxPooling2D(pool_size=(2,2))(x1)

       # Bloque 2
       x2 = layers.Conv2D(2*conv2d_size, (3,3), padding="same",␣
      ↪activation="relu")(x1)
       if batch_norm: x2= layers.BatchNormalization()(x2)
       x2 = layers.MaxPooling2D(pool_size=(2,2))(x2)
```

```
  # Bloque 3
  x3 = layers.Conv2D(4*conv2d_size, (3,3), padding="same",␣
 ↪activation="relu")(x2)
  if batch_norm: x3 = layers.BatchNormalization()(x3)
  x3 = layers.MaxPooling2D(pool_size=(2,2))(x3)

  # 2.TOP MODEL
  if blocks == 1: xfc = layers.Flatten()(x1)
  elif blocks == 2: xfc = layers.Flatten()(x2)
  elif blocks == 3:  xfc = layers.Flatten()(x3)
  else: return 'Arquitectura no valida'

  xfc = layers.Dense(dense_size, activation="relu")(xfc)

  predictions = layers.Dense(num_clases, activation="softmax")(xfc)

  return  Model(inputs=input, outputs=predictions)
```

###**5.2.2. 1CNN**

[ ]:

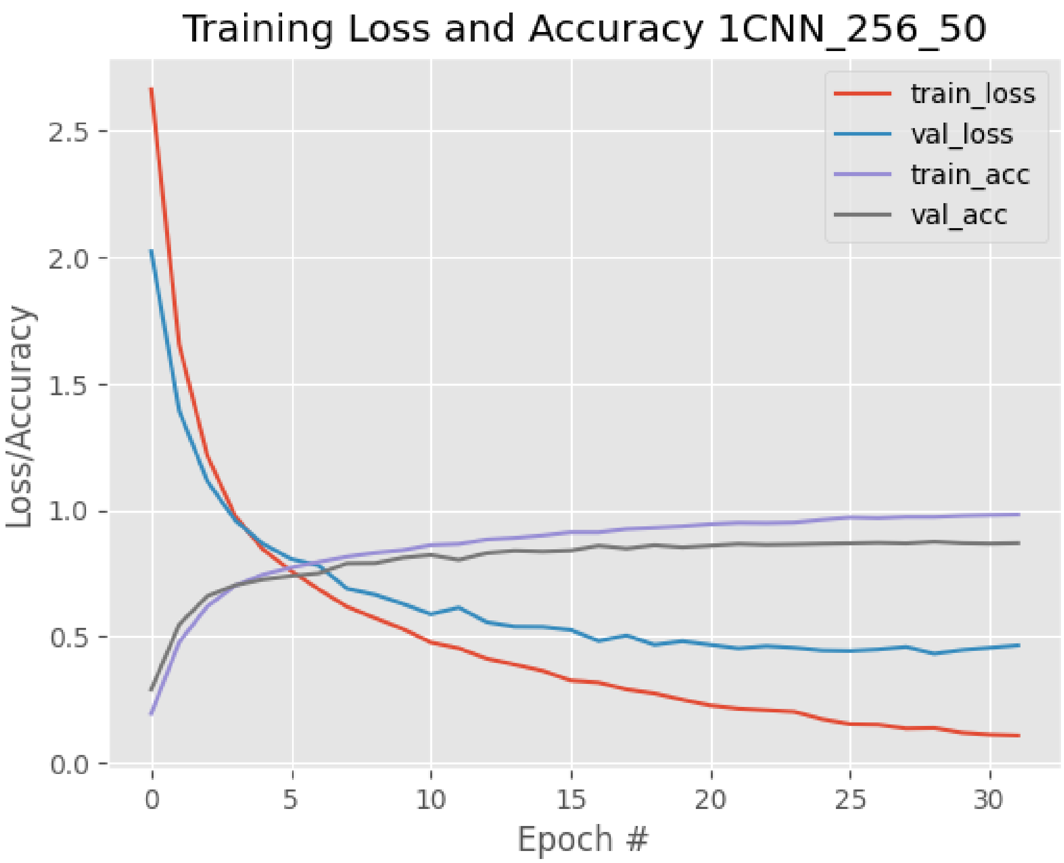
```
# Entrenamiento 1 bloque CNN

n_cnn_blocks = 1
cnn_exp = str(n_cnn_blocks) + "CNN_" + str(batch_size) + "_" + str(epochs)
exp_set.add(cnn_exp)

if do_training == True:

  model_1cnn = get_cnn_model(n_cnn_blocks , x_train_norm,  num_clases,␣
 ↪dense_size)

  model_1cnn.summary()

  model_1cnn.compile(optimizer=Adam(learning_rate=learning_rate),
                     loss="categorical_crossentropy",
                     metrics=["accuracy"])

  print("[INFO]: Entrenando la red convolucional " + cnn_exp + "....")

  H = model_1cnn.fit(x_train_norm, y_train_ohe,
                     batch_size=batch_size,
                     epochs=epochs,
                     steps_per_epoch=x_train_norm.shape[0] // batch_size,
                     validation_data=(x_val_norm, y_val_ohe),
                     callbacks=[early_stopping_cbck])
  save_trained_model(model = model_1cnn, history = H, model_name = cnn_exp)

else:
  print("[INFO]: Cargando la red convolucional " + cnn_exp + "....")
  model_1cnn = load_keras_model(cnn_exp)
  model_1cnn.summary()
  H = load_history(cnn_exp)
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 75, 75, 3)]       0

 conv2d (Conv2D)             (None, 75, 75, 16)        448

 max_pooling2d (MaxPooling2  (None, 37, 37, 16)        0
 D)

 flatten (Flatten)           (None, 21904)             0

 dense (Dense)               (None, 128)               2803840

 dense_1 (Dense)             (None, 15)                1935

=================================================================
Total params: 2806223 (10.70 MB)
Trainable params: 2806223 (10.70 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
[INFO]: Entrenando la red neuronal 1CNN_256_50…
Epoch 1/50
58/58 [==============================] - 14s 56ms/step - loss: 2.6621 -
accuracy: 0.1954 - val_loss: 2.0222 - val_accuracy: 0.2907
Epoch 2/50
58/58 [==============================] - 1s 23ms/step - loss: 1.6526 - accuracy:
0.4780 - val_loss: 1.3920 - val_accuracy: 0.5480
Epoch 3/50
58/58 [==============================] - 1s 25ms/step - loss: 1.2161 - accuracy:
0.6191 - val_loss: 1.1149 - val_accuracy: 0.6597
Epoch 4/50
58/58 [==============================] - 1s 26ms/step - loss: 0.9763 - accuracy:
0.7013 - val_loss: 0.9591 - val_accuracy: 0.7027
Epoch 5/50
58/58 [==============================] - 2s 26ms/step - loss: 0.8438 - accuracy:
0.7450 - val_loss: 0.8659 - val_accuracy: 0.7257
Epoch 6/50
```

```
58/58 [==============================] - 1s 22ms/step - loss: 0.7596 - accuracy:
0.7727 - val_loss: 0.8068 - val_accuracy: 0.7390
Epoch 7/50
58/58 [==============================] - 1s 23ms/step - loss: 0.6858 - accuracy:
0.7951 - val_loss: 0.7802 - val_accuracy: 0.7500
Epoch 8/50
58/58 [==============================] - 2s 30ms/step - loss: 0.6181 - accuracy:
0.8158 - val_loss: 0.6895 - val_accuracy: 0.7890
Epoch 9/50
58/58 [==============================] - 2s 39ms/step - loss: 0.5730 - accuracy:
0.8308 - val_loss: 0.6656 - val_accuracy: 0.7900
Epoch 10/50
58/58 [==============================] - 2s 35ms/step - loss: 0.5304 - accuracy:
0.8413 - val_loss: 0.6296 - val_accuracy: 0.8117
Epoch 11/50
58/58 [==============================] - 2s 32ms/step - loss: 0.4768 - accuracy:
0.8615 - val_loss: 0.5883 - val_accuracy: 0.8223
Epoch 12/50
58/58 [==============================] - 1s 26ms/step - loss: 0.4536 - accuracy:
0.8658 - val_loss: 0.6145 - val_accuracy: 0.8033
Epoch 13/50
58/58 [==============================] - 1s 26ms/step - loss: 0.4126 - accuracy:
0.8835 - val_loss: 0.5569 - val_accuracy: 0.8300
Epoch 14/50
58/58 [==============================] - 1s 23ms/step - loss: 0.3893 - accuracy:
0.8898 - val_loss: 0.5392 - val_accuracy: 0.8390
Epoch 15/50
58/58 [==============================] - 1s 22ms/step - loss: 0.3644 - accuracy:
0.8996 - val_loss: 0.5384 - val_accuracy: 0.8357
Epoch 16/50
58/58 [==============================] - 1s 23ms/step - loss: 0.3265 - accuracy:
0.9122 - val_loss: 0.5265 - val_accuracy: 0.8397
Epoch 17/50
58/58 [==============================] - 1s 23ms/step - loss: 0.3179 - accuracy:
0.9120 - val_loss: 0.4826 - val_accuracy: 0.8590
Epoch 18/50
58/58 [==============================] - 1s 23ms/step - loss: 0.2910 - accuracy:
0.9249 - val_loss: 0.5041 - val_accuracy: 0.8477
Epoch 19/50
58/58 [==============================] - 2s 37ms/step - loss: 0.2751 - accuracy:
0.9301 - val_loss: 0.4676 - val_accuracy: 0.8610
Epoch 20/50
58/58 [==============================] - 2s 32ms/step - loss: 0.2504 - accuracy:
0.9356 - val_loss: 0.4817 - val_accuracy: 0.8523
Epoch 21/50
58/58 [==============================] - 2s 30ms/step - loss: 0.2278 - accuracy:
0.9443 - val_loss: 0.4675 - val_accuracy: 0.8590
Epoch 22/50
```

```
58/58 [==============================] - 2s 30ms/step - loss: 0.2151 - accuracy:
0.9494 - val_loss: 0.4531 - val_accuracy: 0.8660
Epoch 23/50
58/58 [==============================] - 2s 26ms/step - loss: 0.2088 - accuracy:
0.9484 - val_loss: 0.4623 - val_accuracy: 0.8627
Epoch 24/50
58/58 [==============================] - 2s 26ms/step - loss: 0.2028 - accuracy:
0.9505 - val_loss: 0.4555 - val_accuracy: 0.8643
Epoch 25/50
58/58 [==============================] - 1s 24ms/step - loss: 0.1734 - accuracy:
0.9615 - val_loss: 0.4449 - val_accuracy: 0.8667
Epoch 26/50
58/58 [==============================] - 2s 26ms/step - loss: 0.1540 - accuracy:
0.9706 - val_loss: 0.4431 - val_accuracy: 0.8683
Epoch 27/50
58/58 [==============================] - 2s 26ms/step - loss: 0.1521 - accuracy:
0.9681 - val_loss: 0.4489 - val_accuracy: 0.8707
Epoch 28/50
58/58 [==============================] - 1s 26ms/step - loss: 0.1369 - accuracy:
0.9729 - val_loss: 0.4590 - val_accuracy: 0.8680
Epoch 29/50
58/58 [==============================] - 1s 26ms/step - loss: 0.1388 - accuracy:
0.9729 - val_loss: 0.4320 - val_accuracy: 0.8750
Epoch 30/50
58/58 [==============================] - 2s 29ms/step - loss: 0.1201 - accuracy:
0.9775 - val_loss: 0.4467 - val_accuracy: 0.8690
Epoch 31/50
58/58 [==============================] - 2s 28ms/step - loss: 0.1122 - accuracy:
0.9799 - val_loss: 0.4556 - val_accuracy: 0.8670
Epoch 32/50
58/58 [==============================] - 2s 28ms/step - loss: 0.1087 - accuracy:
0.9818 - val_loss: 0.4651 - val_accuracy: 0.8690
Saved model to disk
```

```
[ ]: visualize_learning_curve(H, lb = cnn_exp)
```



```
[ ]: # Evaluación 1CNN

     evaluate_model(model_1cnn, x_test_norm, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 0s 10ms/step
              precision    recall  f1-score   support

           0       0.85      0.77      0.81       200
           1       0.90      0.80      0.85       200
           2       0.96      0.93      0.95       200
           3       0.87      0.96      0.92       200
           4       0.90      0.90      0.90       200
           5       0.87      0.83      0.85       200
           6       0.89      0.94      0.91       200
           7       0.85      0.73      0.79       200
           8       0.90      0.88      0.89       200
           9       0.69      0.92      0.79       200
          10       0.95      0.93      0.94       200
          11       0.99      0.98      0.99       200
          12       0.92      0.72      0.81       200
          13       0.71      0.77      0.73       200
          14       0.80      0.88      0.84       200

    accuracy                           0.86      3000
   macro avg       0.87      0.86      0.86      3000
weighted avg       0.87      0.86      0.86      3000
```

###5.2.3. 2CNN

```
[ ]: # Entrenamiento 2 bloque CNN

n_cnn_blocks = 2
cnn_exp = str(n_cnn_blocks) + "CNN_" + str(batch_size) + "_" + str(epochs)
exp_set.add(cnn_exp)


if do_training == True:

  model_2cnn = get_cnn_model(n_cnn_blocks , x_train_norm,  num_clases,␣
 ↪dense_size)

  model_2cnn.summary()

  model_2cnn.compile(optimizer=Adam(learning_rate=learning_rate),
                     loss="categorical_crossentropy",
                     metrics=["accuracy"])

  print("[INFO]: Entrenando la red convolucional " + cnn_exp + "....")

  H = model_2cnn.fit(x_train_norm, y_train_ohe,
                     batch_size=batch_size,
                     epochs=epochs,
                     steps_per_epoch=x_train_norm.shape[0] // batch_size,
                     validation_data=(x_val_norm, y_val_ohe),
                     callbacks=[early_stopping_cbck])

  save_trained_model(model = model_2cnn, history = H, model_name = cnn_exp)

else:
  print("[INFO]: Cargando la red convolucional " + cnn_exp + "....")
  model_2cnn = load_keras_model(cnn_exp)
  model_2cnn.summary()
  H = load_history(cnn_exp)
```

```
Model: "model_1"
_________________________________________________________________
```

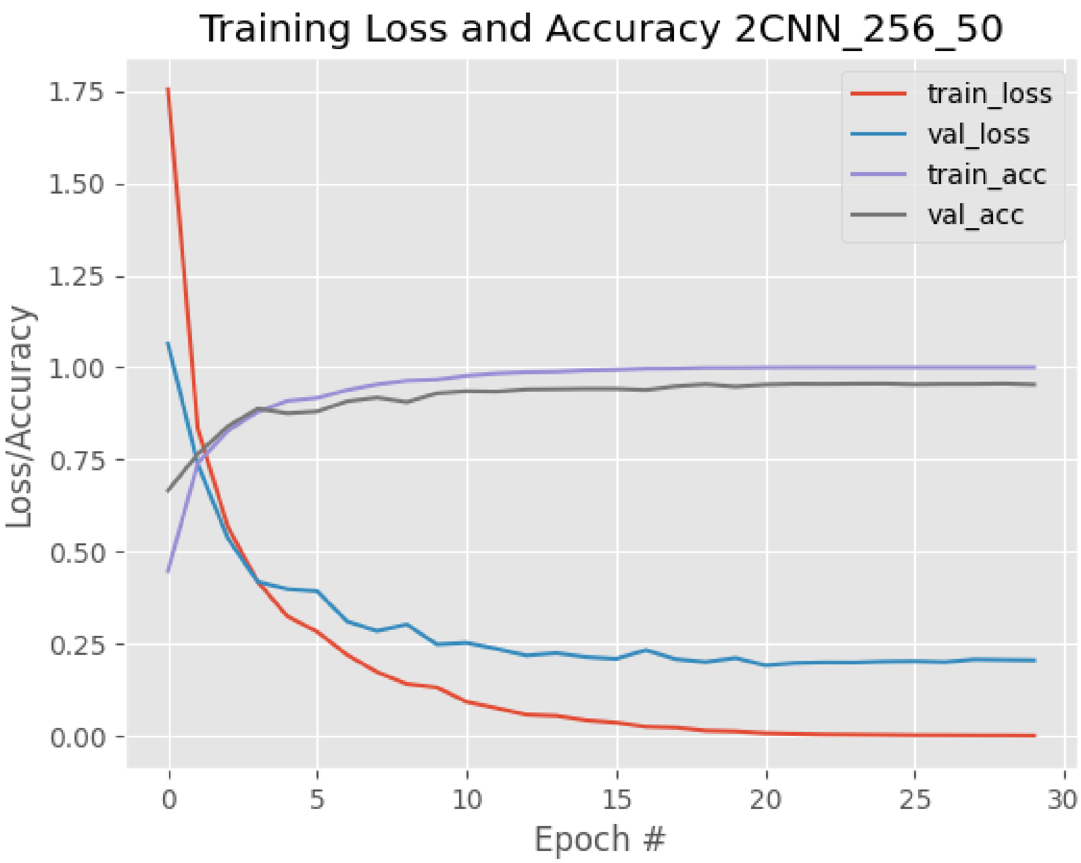
```
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 75, 75, 3)]       0

 conv2d_3 (Conv2D)           (None, 75, 75, 16)        448

 max_pooling2d_3 (MaxPoolin  (None, 37, 37, 16)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 37, 37, 32)        4640

 max_pooling2d_4 (MaxPoolin  (None, 18, 18, 32)        0
 g2D)

 flatten_1 (Flatten)         (None, 10368)             0

 dense_2 (Dense)             (None, 128)               1327232

 dense_3 (Dense)             (None, 15)                1935

=================================================================
Total params: 1334255 (5.09 MB)
Trainable params: 1334255 (5.09 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
[INFO]: Entrenando la red neuronal 2CNN_256_50…
Epoch 1/50
58/58 [==============================] - 5s 42ms/step - loss: 1.7534 - accuracy:
0.4482 - val_loss: 1.0650 - val_accuracy: 0.6663
Epoch 2/50
58/58 [==============================] - 2s 27ms/step - loss: 0.8341 - accuracy:
0.7409 - val_loss: 0.7372 - val_accuracy: 0.7657
Epoch 3/50
58/58 [==============================] - 2s 27ms/step - loss: 0.5690 - accuracy:
0.8279 - val_loss: 0.5375 - val_accuracy: 0.8403
Epoch 4/50
58/58 [==============================] - 2s 27ms/step - loss: 0.4196 - accuracy:
0.8793 - val_loss: 0.4190 - val_accuracy: 0.8883
Epoch 5/50
58/58 [==============================] - 2s 30ms/step - loss: 0.3254 - accuracy:
0.9095 - val_loss: 0.3993 - val_accuracy: 0.8760
Epoch 6/50
58/58 [==============================] - 2s 34ms/step - loss: 0.2837 - accuracy:
0.9173 - val_loss: 0.3935 - val_accuracy: 0.8810
Epoch 7/50
58/58 [==============================] - 2s 34ms/step - loss: 0.2209 - accuracy:
0.9387 - val_loss: 0.3117 - val_accuracy: 0.9083
Epoch 8/50
```

```
58/58 [==============================] - 2s 31ms/step - loss: 0.1744 - accuracy:
0.9539 - val_loss: 0.2859 - val_accuracy: 0.9183
Epoch 9/50
58/58 [==============================] - 2s 29ms/step - loss: 0.1420 - accuracy:
0.9641 - val_loss: 0.3034 - val_accuracy: 0.9063
Epoch 10/50
58/58 [==============================] - 2s 27ms/step - loss: 0.1328 - accuracy:
0.9670 - val_loss: 0.2495 - val_accuracy: 0.9300
Epoch 11/50
58/58 [==============================] - 2s 29ms/step - loss: 0.0937 - accuracy:
0.9775 - val_loss: 0.2536 - val_accuracy: 0.9357
Epoch 12/50
58/58 [==============================] - 2s 29ms/step - loss: 0.0766 - accuracy:
0.9837 - val_loss: 0.2374 - val_accuracy: 0.9343
Epoch 13/50
58/58 [==============================] - 2s 29ms/step - loss: 0.0589 - accuracy:
0.9871 - val_loss: 0.2193 - val_accuracy: 0.9407
Epoch 14/50
58/58 [==============================] - 2s 30ms/step - loss: 0.0558 - accuracy:
0.9882 - val_loss: 0.2259 - val_accuracy: 0.9413
Epoch 15/50
58/58 [==============================] - 2s 36ms/step - loss: 0.0439 - accuracy:
0.9919 - val_loss: 0.2151 - val_accuracy: 0.9423
Epoch 16/50
58/58 [==============================] - 2s 35ms/step - loss: 0.0375 - accuracy:
0.9935 - val_loss: 0.2102 - val_accuracy: 0.9423
Epoch 17/50
58/58 [==============================] - 2s 35ms/step - loss: 0.0265 - accuracy:
0.9965 - val_loss: 0.2338 - val_accuracy: 0.9390
Epoch 18/50
58/58 [==============================] - 2s 27ms/step - loss: 0.0241 - accuracy:
0.9968 - val_loss: 0.2094 - val_accuracy: 0.9490
Epoch 19/50
58/58 [==============================] - 2s 27ms/step - loss: 0.0157 - accuracy:
0.9988 - val_loss: 0.2017 - val_accuracy: 0.9543
Epoch 20/50
58/58 [==============================] - 2s 30ms/step - loss: 0.0136 - accuracy:
0.9989 - val_loss: 0.2125 - val_accuracy: 0.9480
Epoch 21/50
58/58 [==============================] - 2s 27ms/step - loss: 0.0088 - accuracy:
0.9996 - val_loss: 0.1927 - val_accuracy: 0.9533
Epoch 22/50
58/58 [==============================] - 2s 30ms/step - loss: 0.0071 - accuracy:
0.9999 - val_loss: 0.1989 - val_accuracy: 0.9553
Epoch 23/50
58/58 [==============================] - 2s 30ms/step - loss: 0.0056 - accuracy:
1.0000 - val_loss: 0.2005 - val_accuracy: 0.9550
Epoch 24/50
```

```
58/58 [==============================] - 3s 50ms/step - loss: 0.0050 - accuracy:
0.9999 - val_loss: 0.2003 - val_accuracy: 0.9557
Epoch 25/50
58/58 [==============================] - 4s 66ms/step - loss: 0.0043 - accuracy:
1.0000 - val_loss: 0.2029 - val_accuracy: 0.9560
Epoch 26/50
58/58 [==============================] - 3s 54ms/step - loss: 0.0035 - accuracy:
1.0000 - val_loss: 0.2037 - val_accuracy: 0.9540
Epoch 27/50
58/58 [==============================] - 3s 46ms/step - loss: 0.0033 - accuracy:
1.0000 - val_loss: 0.2016 - val_accuracy: 0.9550
Epoch 28/50
58/58 [==============================] - 3s 45ms/step - loss: 0.0029 - accuracy:
1.0000 - val_loss: 0.2086 - val_accuracy: 0.9550
Epoch 29/50
58/58 [==============================] - 2s 43ms/step - loss: 0.0028 - accuracy:
1.0000 - val_loss: 0.2074 - val_accuracy: 0.9560
Epoch 30/50
58/58 [==============================] - 4s 71ms/step - loss: 0.0023 - accuracy:
1.0000 - val_loss: 0.2065 - val_accuracy: 0.9537
Saved model to disk
```

[ ]:

```
visualize_learning_curve(H, lb = cnn_exp)
```



```
[ ]: # Evaluación 2CNN

     evaluate_model(model_2cnn, x_test_norm, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 0s 10ms/step
              precision    recall  f1-score   support

           0       0.97      0.94      0.96       200
           1       0.94      0.94      0.94       200
           2       0.99      0.99      0.99       200
           3       0.98      0.99      0.99       200
           4       0.97      0.97      0.97       200
           5       0.92      0.94      0.93       200
           6       0.96      0.96      0.96       200
           7       0.94      0.95      0.95       200
           8       0.95      0.95      0.95       200
           9       0.90      0.94      0.92       200
          10       0.98      0.99      0.98       200
          11       1.00      1.00      1.00       200
          12       0.96      0.88      0.92       200
          13       0.91      0.92      0.91       200
          14       0.95      0.94      0.95       200

    accuracy                           0.95      3000
   macro avg       0.95      0.95      0.95      3000
weighted avg       0.95      0.95      0.95      3000
```

###**5.2.4. 3CNN**

a) Original

```
[ ]: # Entrenamiento 3 bloques CNN

     n_cnn_blocks = 3
     cnn_exp = str(n_cnn_blocks) + "CNN_" + str(batch_size) + "_" + str(epochs)
     exp_set.add(cnn_exp)

     if do_training == True:

       model_3cnn = get_cnn_model(n_cnn_blocks , x_train_norm,  num_clases,␣
      ↪dense_size)

       model_3cnn.summary()

       model_3cnn.compile(optimizer=Adam(learning_rate=learning_rate),
                          loss="categorical_crossentropy",
                          metrics=["accuracy"])

       print("[INFO]: Entrenando la red neuronal " + cnn_exp + "....")

       H = model_3cnn.fit(x_train_norm, y_train_ohe,
                          batch_size=batch_size,
                          epochs=epochs,
                          steps_per_epoch=x_train_norm.shape[0] // batch_size,
                          validation_data=(x_val_norm, y_val_ohe),
                          callbacks=[early_stopping_cbck])

       save_trained_model(model = model_3cnn, history = H, model_name = cnn_exp)

     else:
       print("[INFO]: Cargando la red convolucional " + cnn_exp + "....")
       model_3cnn = load_keras_model(cnn_exp)
       model_3cnn.summary()
       H = load_history(cnn_exp)
```

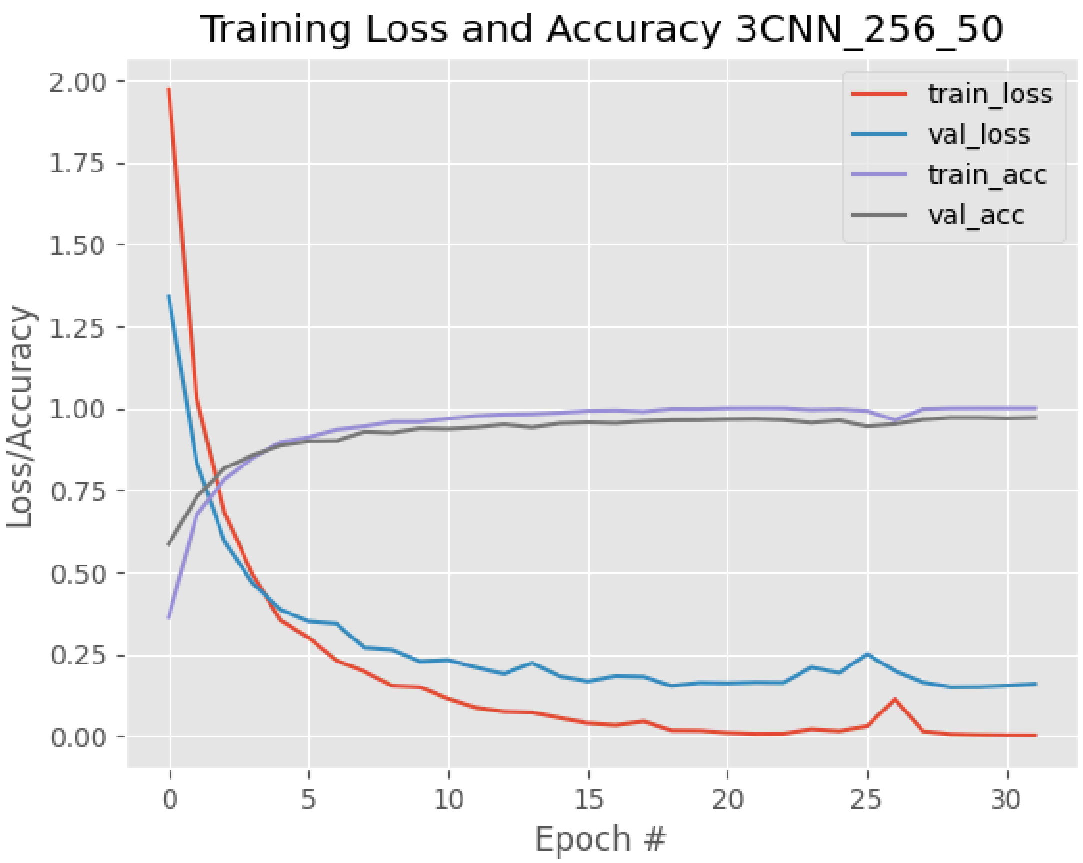
```
Model: "model"
_________________________________________________________________
```

```
 Layer (type)                Output Shape              Param #   
=================================================================
 input_1 (InputLayer)        [(None, 75, 75, 3)]       0         
                                                                 
 conv2d (Conv2D)             (None, 75, 75, 16)        448       
                                                                 
 max_pooling2d (MaxPooling2  (None, 37, 37, 16)        0         
 D)                                                              
                                                                 
 conv2d_1 (Conv2D)           (None, 37, 37, 32)        4640      
                                                                 
 max_pooling2d_1 (MaxPoolin  (None, 18, 18, 32)        0         
 g2D)                                                            
                                                                 
 conv2d_2 (Conv2D)           (None, 18, 18, 64)        18496     
                                                                 
 max_pooling2d_2 (MaxPoolin  (None, 9, 9, 64)          0         
 g2D)                                                            
                                                                 
 flatten (Flatten)           (None, 5184)              0         
                                                                 
 dense (Dense)               (None, 128)               663680    
                                                                 
 dense_1 (Dense)             (None, 15)                1935      
                                                                 
=================================================================
Total params: 689199 (2.63 MB)
Trainable params: 689199 (2.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
[INFO]: Entrenando la red neuronal 3CNN_256_50…
Epoch 1/50
58/58 [==============================] - 15s 60ms/step - loss: 1.9718 -
accuracy: 0.3616 - val_loss: 1.3408 - val_accuracy: 0.5850
Epoch 2/50
58/58 [==============================] - 2s 38ms/step - loss: 1.0307 - accuracy:
0.6740 - val_loss: 0.8346 - val_accuracy: 0.7283
Epoch 3/50
58/58 [==============================] - 2s 38ms/step - loss: 0.6809 - accuracy:
0.7831 - val_loss: 0.5934 - val_accuracy: 0.8167
Epoch 4/50
58/58 [==============================] - 2s 32ms/step - loss: 0.4923 - accuracy:
0.8475 - val_loss: 0.4660 - val_accuracy: 0.8557
Epoch 5/50
58/58 [==============================] - 2s 33ms/step - loss: 0.3521 - accuracy:
0.8961 - val_loss: 0.3850 - val_accuracy: 0.8857
Epoch 6/50
58/58 [==============================] - 2s 39ms/step - loss: 0.3000 - accuracy:
```

```
0.9111 - val_loss: 0.3491 - val_accuracy: 0.8990
Epoch 7/50
58/58 [==============================] - 2s 41ms/step - loss: 0.2301 - accuracy:
0.9340 - val_loss: 0.3422 - val_accuracy: 0.9003
Epoch 8/50
58/58 [==============================] - 2s 34ms/step - loss: 0.1962 - accuracy:
0.9445 - val_loss: 0.2687 - val_accuracy: 0.9280
Epoch 9/50
58/58 [==============================] - 2s 36ms/step - loss: 0.1525 - accuracy:
0.9582 - val_loss: 0.2620 - val_accuracy: 0.9247
Epoch 10/50
58/58 [==============================] - 2s 37ms/step - loss: 0.1482 - accuracy:
0.9582 - val_loss: 0.2271 - val_accuracy: 0.9383
Epoch 11/50
58/58 [==============================] - 2s 38ms/step - loss: 0.1128 - accuracy:
0.9679 - val_loss: 0.2313 - val_accuracy: 0.9367
Epoch 12/50
58/58 [==============================] - 2s 36ms/step - loss: 0.0861 - accuracy:
0.9761 - val_loss: 0.2085 - val_accuracy: 0.9413
Epoch 13/50
58/58 [==============================] - 2s 34ms/step - loss: 0.0738 - accuracy:
0.9805 - val_loss: 0.1892 - val_accuracy: 0.9503
Epoch 14/50
58/58 [==============================] - 2s 31ms/step - loss: 0.0713 - accuracy:
0.9819 - val_loss: 0.2223 - val_accuracy: 0.9413
Epoch 15/50
58/58 [==============================] - 2s 31ms/step - loss: 0.0546 - accuracy:
0.9858 - val_loss: 0.1817 - val_accuracy: 0.9537
Epoch 16/50
58/58 [==============================] - 2s 31ms/step - loss: 0.0389 - accuracy:
0.9913 - val_loss: 0.1663 - val_accuracy: 0.9567
Epoch 17/50
58/58 [==============================] - 2s 35ms/step - loss: 0.0329 - accuracy:
0.9927 - val_loss: 0.1824 - val_accuracy: 0.9543
Epoch 18/50
58/58 [==============================] - 2s 38ms/step - loss: 0.0442 - accuracy:
0.9893 - val_loss: 0.1803 - val_accuracy: 0.9600
Epoch 19/50
58/58 [==============================] - 2s 38ms/step - loss: 0.0167 - accuracy:
0.9976 - val_loss: 0.1520 - val_accuracy: 0.9633
Epoch 20/50
58/58 [==============================] - 2s 34ms/step - loss: 0.0159 - accuracy:
0.9973 - val_loss: 0.1618 - val_accuracy: 0.9637
Epoch 21/50
58/58 [==============================] - 2s 31ms/step - loss: 0.0093 - accuracy:
0.9994 - val_loss: 0.1604 - val_accuracy: 0.9660
Epoch 22/50
58/58 [==============================] - 2s 31ms/step - loss: 0.0061 - accuracy:
```

```
0.9997 - val_loss: 0.1629 - val_accuracy: 0.9670
Epoch 23/50
58/58 [==============================] - 2s 32ms/step - loss: 0.0066 - accuracy:
0.9995 - val_loss: 0.1621 - val_accuracy: 0.9640
Epoch 24/50
58/58 [==============================] - 2s 34ms/step - loss: 0.0203 - accuracy:
0.9946 - val_loss: 0.2088 - val_accuracy: 0.9553
Epoch 25/50
58/58 [==============================] - 2s 31ms/step - loss: 0.0146 - accuracy:
0.9966 - val_loss: 0.1923 - val_accuracy: 0.9627
Epoch 26/50
58/58 [==============================] - 2s 37ms/step - loss: 0.0300 - accuracy:
0.9910 - val_loss: 0.2491 - val_accuracy: 0.9443
Epoch 27/50
58/58 [==============================] - 2s 35ms/step - loss: 0.1119 - accuracy:
0.9632 - val_loss: 0.1978 - val_accuracy: 0.9517
Epoch 28/50
58/58 [==============================] - 2s 37ms/step - loss: 0.0139 - accuracy:
0.9972 - val_loss: 0.1632 - val_accuracy: 0.9650
Epoch 29/50
58/58 [==============================] - 2s 32ms/step - loss: 0.0049 - accuracy:
0.9996 - val_loss: 0.1482 - val_accuracy: 0.9710
Epoch 30/50
58/58 [==============================] - 2s 34ms/step - loss: 0.0028 - accuracy:
0.9999 - val_loss: 0.1489 - val_accuracy: 0.9710
Epoch 31/50
58/58 [==============================] - 2s 31ms/step - loss: 0.0017 - accuracy:
1.0000 - val_loss: 0.1528 - val_accuracy: 0.9690
Epoch 32/50
58/58 [==============================] - 2s 34ms/step - loss: 0.0013 - accuracy:
1.0000 - val_loss: 0.1581 - val_accuracy: 0.9710
Saved model to disk
```

```
[ ]: visualize_learning_curve(H, lb = cnn_exp)
```



```
[ ]: # Evaluación 3CNN

     evaluate_model(model_3cnn, x_test_norm, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 1s 11ms/step
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       200
           1       0.94      0.93      0.93       200
           2       1.00      1.00      1.00       200
           3       0.98      0.98      0.98       200
           4       0.98      0.97      0.98       200
           5       0.92      0.95      0.93       200
           6       0.97      0.98      0.97       200
           7       0.95      0.95      0.95       200
           8       0.99      0.97      0.98       200
           9       0.93      0.96      0.95       200
          10       0.98      0.98      0.98       200
          11       1.00      0.99      1.00       200
          12       0.95      0.92      0.93       200
          13       0.94      0.94      0.94       200
          14       0.98      0.96      0.97       200

    accuracy                           0.97      3000
   macro avg       0.97      0.97      0.97      3000
weighted avg       0.97      0.97      0.97      3000
```

Comentario: El model_3cnn converge a partir de 32 épocas (en lugar de las 50 fijadas) gracias al `early stopping`.

b) Batch Normalization

```
[ ]: # Entrenamiento CNN de 3 bloques con Batch Normalization (batch_norm=True)

     n_cnn_blocks = 3
     cnn_exp = str(n_cnn_blocks) + "CNN_BN_" + str(batch_size) + "_" + str(epochs)
     exp_set.add(cnn_exp)

     if do_training == True:

       model_3cnn_batch_norm = get_cnn_model(n_cnn_blocks , x_train_norm, ␣
       ↪num_clases, dense_size, batch_norm = True)

       model_3cnn_batch_norm.summary()

       model_3cnn_batch_norm.compile(optimizer=Adam(learning_rate=learning_rate),
                                     loss="categorical_crossentropy",
                                     metrics=["accuracy"])

       print("[INFO]: Entrenando la red neuronal " + cnn_exp + "....")

       H = model_3cnn_batch_norm.fit(x_train_norm, y_train_ohe,
                                     batch_size=batch_size,
                                     epochs=epochs,
                                     steps_per_epoch=x_train_norm.shape[0] //␣
       ↪batch_size,
                                     validation_data=(x_val_norm, y_val_ohe),
                                     callbacks=[early_stopping_cbck])

       save_trained_model(model = model_3cnn_batch_norm, history = H, model_name =␣
       ↪cnn_exp)

     else:
       print("[INFO]: Cargando la red convolucional " + cnn_exp + "....")
       model_3cnn_batch_norm = load_keras_model(cnn_exp)
       model_3cnn_batch_norm.summary()
```

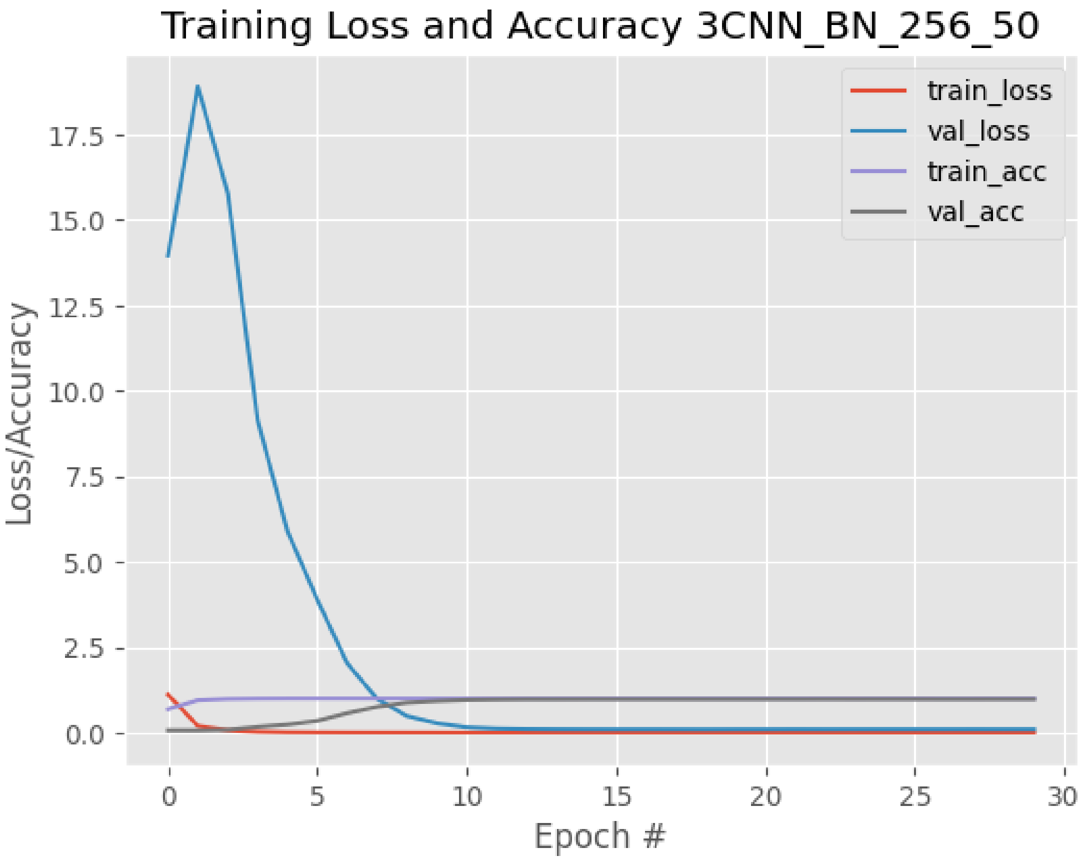
```
H = load_history(cnn_exp)
```

```
Model: "model_1"
-----------------------------------------------------------------
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 75, 75, 3)]       0

 conv2d_3 (Conv2D)           (None, 75, 75, 16)        448

 batch_normalization (Batch  (None, 75, 75, 16)        64
 Normalization)

 max_pooling2d_3 (MaxPoolin  (None, 37, 37, 16)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 37, 37, 32)        4640

 batch_normalization_1 (Bat  (None, 37, 37, 32)        128
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 18, 18, 32)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 18, 18, 64)        18496

 batch_normalization_2 (Bat  (None, 18, 18, 64)        256
 chNormalization)

 max_pooling2d_5 (MaxPoolin  (None, 9, 9, 64)          0
 g2D)

 flatten_1 (Flatten)         (None, 5184)              0

 dense_2 (Dense)             (None, 128)               663680

 dense_3 (Dense)             (None, 15)                1935

=================================================================
Total params: 689647 (2.63 MB)
Trainable params: 689423 (2.63 MB)
Non-trainable params: 224 (896.00 Byte)
-----------------------------------------------------------------
[INFO]: Entrenando la red neuronal 3CNN_BN_256_50…
Epoch 1/50
58/58 [==============================] - 7s 56ms/step - loss: 1.1133 - accuracy:
0.6839 - val_loss: 13.9592 - val_accuracy: 0.0673
```

```
Epoch 2/50
58/58 [==============================] - 2s 39ms/step - loss: 0.1940 - accuracy:
0.9465 - val_loss: 18.9109 - val_accuracy: 0.0667
Epoch 3/50
58/58 [==============================] - 2s 41ms/step - loss: 0.0693 - accuracy:
0.9845 - val_loss: 15.7915 - val_accuracy: 0.0820
Epoch 4/50
58/58 [==============================] - 2s 39ms/step - loss: 0.0268 - accuracy:
0.9953 - val_loss: 9.1517 - val_accuracy: 0.1670
Epoch 5/50
58/58 [==============================] - 3s 44ms/step - loss: 0.0103 - accuracy:
0.9992 - val_loss: 5.8915 - val_accuracy: 0.2350
Epoch 6/50
58/58 [==============================] - 3s 45ms/step - loss: 0.0055 - accuracy:
0.9997 - val_loss: 3.8889 - val_accuracy: 0.3447
Epoch 7/50
58/58 [==============================] - 3s 46ms/step - loss: 0.0030 - accuracy:
1.0000 - val_loss: 2.0262 - val_accuracy: 0.5670
Epoch 8/50
58/58 [==============================] - 2s 41ms/step - loss: 0.0021 - accuracy:
1.0000 - val_loss: 0.9888 - val_accuracy: 0.7523
Epoch 9/50
58/58 [==============================] - 2s 42ms/step - loss: 0.0017 - accuracy:
1.0000 - val_loss: 0.4832 - val_accuracy: 0.8753
Epoch 10/50
58/58 [==============================] - 2s 39ms/step - loss: 0.0014 - accuracy:
1.0000 - val_loss: 0.2755 - val_accuracy: 0.9240
Epoch 11/50
58/58 [==============================] - 2s 40ms/step - loss: 0.0011 - accuracy:
1.0000 - val_loss: 0.1547 - val_accuracy: 0.9567
Epoch 12/50
58/58 [==============================] - 3s 45ms/step - loss: 9.4288e-04 -
accuracy: 1.0000 - val_loss: 0.1171 - val_accuracy: 0.9677
Epoch 13/50
58/58 [==============================] - 3s 46ms/step - loss: 7.6242e-04 -
accuracy: 1.0000 - val_loss: 0.1045 - val_accuracy: 0.9740
Epoch 14/50
58/58 [==============================] - 2s 42ms/step - loss: 7.8235e-04 -
accuracy: 1.0000 - val_loss: 0.0996 - val_accuracy: 0.9740
Epoch 15/50
58/58 [==============================] - 2s 41ms/step - loss: 6.7878e-04 -
accuracy: 1.0000 - val_loss: 0.0977 - val_accuracy: 0.9757
Epoch 16/50
58/58 [==============================] - 2s 40ms/step - loss: 5.7802e-04 -
accuracy: 1.0000 - val_loss: 0.0968 - val_accuracy: 0.9760
Epoch 17/50
58/58 [==============================] - 2s 40ms/step - loss: 4.9050e-04 -
accuracy: 1.0000 - val_loss: 0.0971 - val_accuracy: 0.9763
```

```
Epoch 18/50
58/58 [==============================] - 2s 42ms/step - loss: 4.4036e-04 -
accuracy: 1.0000 - val_loss: 0.0961 - val_accuracy: 0.9767
Epoch 19/50
58/58 [==============================] - 3s 46ms/step - loss: 4.1996e-04 -
accuracy: 1.0000 - val_loss: 0.0951 - val_accuracy: 0.9767
Epoch 20/50
58/58 [==============================] - 3s 45ms/step - loss: 3.5869e-04 -
accuracy: 1.0000 - val_loss: 0.0965 - val_accuracy: 0.9770
Epoch 21/50
58/58 [==============================] - 2s 40ms/step - loss: 3.3961e-04 -
accuracy: 1.0000 - val_loss: 0.0967 - val_accuracy: 0.9770
Epoch 22/50
58/58 [==============================] - 2s 39ms/step - loss: 3.1102e-04 -
accuracy: 1.0000 - val_loss: 0.0983 - val_accuracy: 0.9767
Epoch 23/50
58/58 [==============================] - 2s 40ms/step - loss: 2.8927e-04 -
accuracy: 1.0000 - val_loss: 0.0969 - val_accuracy: 0.9780
Epoch 24/50
58/58 [==============================] - 2s 42ms/step - loss: 2.5080e-04 -
accuracy: 1.0000 - val_loss: 0.0975 - val_accuracy: 0.9773
Epoch 25/50
58/58 [==============================] - 3s 44ms/step - loss: 2.6791e-04 -
accuracy: 1.0000 - val_loss: 0.0978 - val_accuracy: 0.9780
Epoch 26/50
58/58 [==============================] - 3s 46ms/step - loss: 2.1261e-04 -
accuracy: 1.0000 - val_loss: 0.0977 - val_accuracy: 0.9777
Epoch 27/50
58/58 [==============================] - 3s 44ms/step - loss: 1.9148e-04 -
accuracy: 1.0000 - val_loss: 0.0972 - val_accuracy: 0.9770
Epoch 28/50
58/58 [==============================] - 2s 42ms/step - loss: 1.9470e-04 -
accuracy: 1.0000 - val_loss: 0.0982 - val_accuracy: 0.9773
Epoch 29/50
58/58 [==============================] - 2s 40ms/step - loss: 1.7760e-04 -
accuracy: 1.0000 - val_loss: 0.0974 - val_accuracy: 0.9780
Epoch 30/50
58/58 [==============================] - 2s 40ms/step - loss: 1.7409e-04 -
accuracy: 1.0000 - val_loss: 0.0978 - val_accuracy: 0.9780
Saved model to disk
```

```
[ ]: visualize_learning_curve(H, lb = cnn_exp)
```



```
[ ]: evaluate_model(model_3cnn_batch_norm, x_test_norm, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 0s 8ms/step
              precision    recall  f1-score   support

           0       0.98      0.97      0.98       200
           1       0.95      0.94      0.95       200
           2       0.99      1.00      1.00       200
           3       0.99      1.00      1.00       200
           4       0.99      1.00      1.00       200
           5       0.95      0.98      0.97       200
           6       0.98      0.98      0.98       200
           7       0.98      0.94      0.96       200
           8       0.98      0.98      0.98       200
           9       0.98      0.98      0.98       200
          10       0.98      0.99      0.99       200
          11       1.00      1.00      1.00       200
          12       0.97      0.95      0.96       200
          13       0.97      0.97      0.97       200
          14       0.97      0.96      0.97       200

    accuracy                           0.98      3000
   macro avg       0.98      0.98      0.98      3000
weighted avg       0.98      0.98      0.98      3000
```

c) CNN + Data Augmentation + Batch Norm Entrenar modelo CNN incorporando técnicas de aumentación de datos, lo que puede resultar en una convergencia más irregular debido a la diversidad y cantidad incrementada de imágenes. La aumentación de datos ayuda a la CNN a generalizar mejor, contribuyendo a una reducción general en las pérdidas de entrenamiento y validación. Las pérdidas más bajas en validación, en comparación con el entrenamiento, pueden deberse a que la aumentación introduce variabilidad en el conjunto de entrenamiento, mientras que el conjunto de validación, sin aumentar, refleja con mayor precisión la distribución natural de los datos.

```
[ ]: # Entrenamiento CNN de 3 bloques con Data Augmentation y Batch Normalization
     n_cnn_blocks = 3
     cnn_exp = str(n_cnn_blocks) + "CNN_BN_AUG_" + str(batch_size) + "_" +␣
      ↪str(epochs)
     exp_set.add(cnn_exp)

     if do_training == True:

       model_3cnn_aug = get_cnn_model(n_cnn_blocks , x_train_norm,  num_clases,␣
      ↪dense_size, batch_norm = True)

       model_3cnn_aug.summary()

       model_3cnn_aug.compile(optimizer=Adam(learning_rate=learning_rate),
                              loss="categorical_crossentropy",
                              metrics=["accuracy"])

       print("[INFO]: Entrenando la red neuronal " + cnn_exp + " con data␣
      ↪augmentation...")

       H_aug = model_3cnn_aug.fit(train_generator,
                         steps_per_epoch=train_generator.samples // batch_size_aug,
                         validation_data=val_generator,
                         validation_steps=val_generator.samples // batch_size_aug,
                         epochs=epochs,
                         callbacks=[early_stopping_cbck])

       save_trained_model(model = model_3cnn_aug, history = H_aug, model_name =␣
      ↪cnn_exp)

     else:
```

```
print("[INFO]: Cargando la red convolucional " + cnn_exp + "....")
model_3cnn_aug = load_keras_model(cnn_exp)
model_3cnn_aug.summary()
H_aug = load_history(cnn_exp)
```

```
Model: "model_1"
-----------------------------------------------------------------
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 75, 75, 3)]       0

 conv2d_3 (Conv2D)           (None, 75, 75, 16)        448

 batch_normalization_3 (Bat  (None, 75, 75, 16)        64
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 37, 37, 16)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 37, 37, 32)        4640

 batch_normalization_4 (Bat  (None, 37, 37, 32)        128
 chNormalization)

 max_pooling2d_4 (MaxPoolin  (None, 18, 18, 32)        0
 g2D)

 conv2d_5 (Conv2D)           (None, 18, 18, 64)        18496

 batch_normalization_5 (Bat  (None, 18, 18, 64)        256
 chNormalization)

 max_pooling2d_5 (MaxPoolin  (None, 9, 9, 64)          0
 g2D)

 flatten_1 (Flatten)         (None, 5184)              0

 dense_2 (Dense)             (None, 128)               663680

 dense_3 (Dense)             (None, 15)                1935

=================================================================
Total params: 689647 (2.63 MB)
Trainable params: 689423 (2.63 MB)
Non-trainable params: 224 (896.00 Byte)
-----------------------------------------------------------------
[INFO]: Entrenando la red neuronal 3CNN_BN_AUG_256_50 con data augmentation…
```

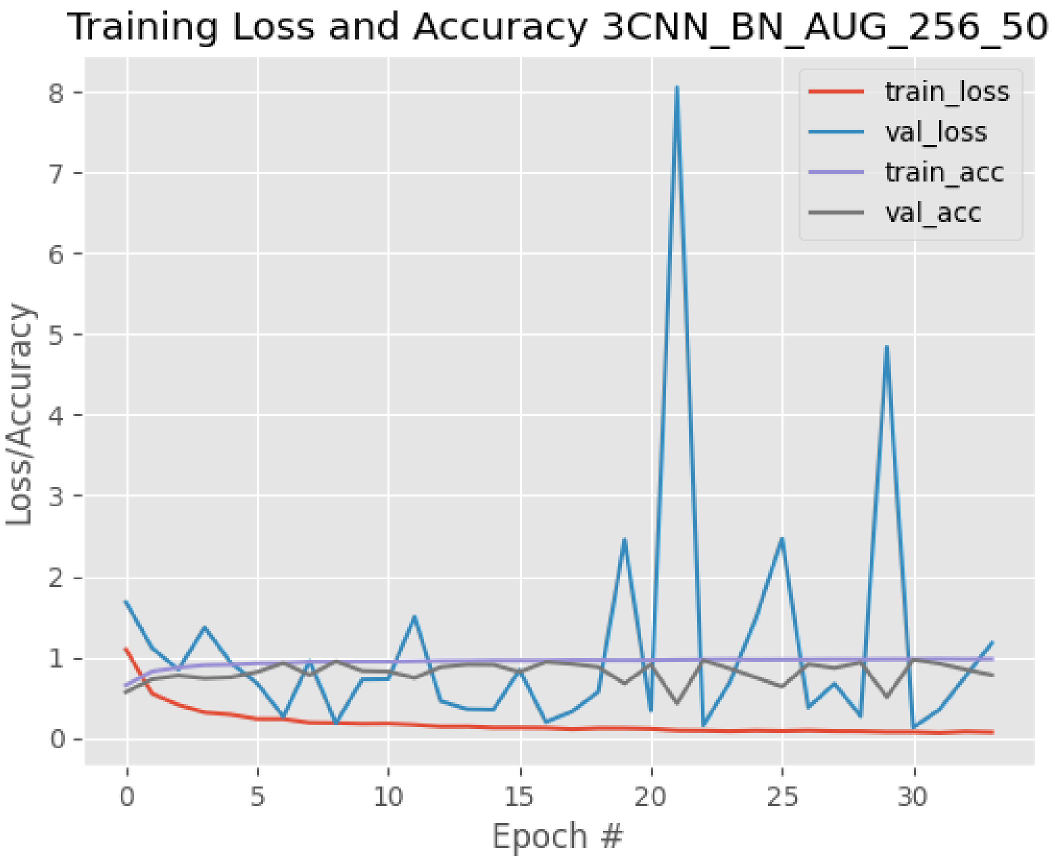
```
Epoch 1/50
468/468 [==============================] - 51s 105ms/step - loss: 1.0956 -
accuracy: 0.6555 - val_loss: 1.6820 - val_accuracy: 0.5699
Epoch 2/50
468/468 [==============================] - 47s 101ms/step - loss: 0.5487 -
accuracy: 0.8246 - val_loss: 1.1115 - val_accuracy: 0.7302
Epoch 3/50
468/468 [==============================] - 49s 104ms/step - loss: 0.4097 -
accuracy: 0.8716 - val_loss: 0.8470 - val_accuracy: 0.7745
Epoch 4/50
468/468 [==============================] - 49s 104ms/step - loss: 0.3166 -
accuracy: 0.9023 - val_loss: 1.3702 - val_accuracy: 0.7399
Epoch 5/50
468/468 [==============================] - 47s 99ms/step - loss: 0.2906 -
accuracy: 0.9080 - val_loss: 0.9292 - val_accuracy: 0.7534
Epoch 6/50
468/468 [==============================] - 50s 107ms/step - loss: 0.2369 -
accuracy: 0.9252 - val_loss: 0.6780 - val_accuracy: 0.8172
Epoch 7/50
468/468 [==============================] - 58s 123ms/step - loss: 0.2362 -
accuracy: 0.9267 - val_loss: 0.2646 - val_accuracy: 0.9278
Epoch 8/50
468/468 [==============================] - 48s 102ms/step - loss: 0.1905 -
accuracy: 0.9417 - val_loss: 0.9540 - val_accuracy: 0.7796
Epoch 9/50
468/468 [==============================] - 47s 101ms/step - loss: 0.1864 -
accuracy: 0.9415 - val_loss: 0.1810 - val_accuracy: 0.9496
Epoch 10/50
468/468 [==============================] - 50s 107ms/step - loss: 0.1773 -
accuracy: 0.9444 - val_loss: 0.7260 - val_accuracy: 0.8306
Epoch 11/50
468/468 [==============================] - 50s 107ms/step - loss: 0.1797 -
accuracy: 0.9439 - val_loss: 0.7303 - val_accuracy: 0.8233
Epoch 12/50
468/468 [==============================] - 48s 102ms/step - loss: 0.1653 -
accuracy: 0.9496 - val_loss: 1.5005 - val_accuracy: 0.7450
Epoch 13/50
468/468 [==============================] - 47s 101ms/step - loss: 0.1428 -
accuracy: 0.9546 - val_loss: 0.4564 - val_accuracy: 0.8780
Epoch 14/50
468/468 [==============================] - 48s 102ms/step - loss: 0.1434 -
accuracy: 0.9563 - val_loss: 0.3570 - val_accuracy: 0.9093
Epoch 15/50
468/468 [==============================] - 47s 101ms/step - loss: 0.1282 -
accuracy: 0.9602 - val_loss: 0.3501 - val_accuracy: 0.9073
Epoch 16/50
468/468 [==============================] - 48s 103ms/step - loss: 0.1284 -
accuracy: 0.9606 - val_loss: 0.8444 - val_accuracy: 0.8179
```

```
Epoch 17/50
468/468 [==============================] - 47s 101ms/step - loss: 0.1260 -
accuracy: 0.9608 - val_loss: 0.1994 - val_accuracy: 0.9412
Epoch 18/50
468/468 [==============================] - 45s 95ms/step - loss: 0.1097 -
accuracy: 0.9649 - val_loss: 0.3287 - val_accuracy: 0.9163
Epoch 19/50
468/468 [==============================] - 48s 102ms/step - loss: 0.1222 -
accuracy: 0.9639 - val_loss: 0.5719 - val_accuracy: 0.8784
Epoch 20/50
468/468 [==============================] - 49s 105ms/step - loss: 0.1211 -
accuracy: 0.9635 - val_loss: 2.4550 - val_accuracy: 0.6754
Epoch 21/50
468/468 [==============================] - 48s 102ms/step - loss: 0.1149 -
accuracy: 0.9641 - val_loss: 0.3384 - val_accuracy: 0.9110
Epoch 22/50
468/468 [==============================] - 49s 105ms/step - loss: 0.0952 -
accuracy: 0.9684 - val_loss: 8.0502 - val_accuracy: 0.4261
Epoch 23/50
468/468 [==============================] - 45s 97ms/step - loss: 0.0929 -
accuracy: 0.9715 - val_loss: 0.1554 - val_accuracy: 0.9607
Epoch 24/50
468/468 [==============================] - 52s 111ms/step - loss: 0.0858 -
accuracy: 0.9745 - val_loss: 0.6842 - val_accuracy: 0.8612
Epoch 25/50
468/468 [==============================] - 52s 110ms/step - loss: 0.0933 -
accuracy: 0.9700 - val_loss: 1.4755 - val_accuracy: 0.7507
Epoch 26/50
468/468 [==============================] - 50s 106ms/step - loss: 0.0879 -
accuracy: 0.9715 - val_loss: 2.4684 - val_accuracy: 0.6368
Epoch 27/50
468/468 [==============================] - 51s 109ms/step - loss: 0.0947 -
accuracy: 0.9719 - val_loss: 0.3771 - val_accuracy: 0.9136
Epoch 28/50
468/468 [==============================] - 52s 110ms/step - loss: 0.0866 -
accuracy: 0.9743 - val_loss: 0.6746 - val_accuracy: 0.8659
Epoch 29/50
468/468 [==============================] - 51s 110ms/step - loss: 0.0854 -
accuracy: 0.9730 - val_loss: 0.2696 - val_accuracy: 0.9348
Epoch 30/50
468/468 [==============================] - 48s 103ms/step - loss: 0.0773 -
accuracy: 0.9759 - val_loss: 4.8395 - val_accuracy: 0.5081
Epoch 31/50
468/468 [==============================] - 51s 110ms/step - loss: 0.0771 -
accuracy: 0.9769 - val_loss: 0.1318 - val_accuracy: 0.9708
Epoch 32/50
468/468 [==============================] - 49s 105ms/step - loss: 0.0662 -
accuracy: 0.9800 - val_loss: 0.3517 - val_accuracy: 0.9207
```

```
Epoch 33/50
468/468 [==============================] - 50s 107ms/step - loss: 0.0822 -
accuracy: 0.9763 - val_loss: 0.7668 - val_accuracy: 0.8468
Epoch 34/50
468/468 [==============================] - 47s 100ms/step - loss: 0.0740 -
accuracy: 0.9785 - val_loss: 1.1760 - val_accuracy: 0.7796
Saved model to disk
```

[ ]:

```
visualize_learning_curve(H_aug, lb = cnn_exp)
```



[ ]:

```
# Función de evaluación.
evaluate_model_aug(model_3cnn_aug, test_generator)
```

```
[INFO]: Evaluando red neuronal…

<ipython-input-22-e5396da28fb4>:27: UserWarning: `Model.predict_generator` is
deprecated and will be removed in a future version. Please use `Model.predict`,
which supports generators.
  predictions = model.predict_generator(generator, steps=len(generator))

              precision    recall  f1-score   support
```

```
           0       0.57      0.93      0.70       200
           1       0.57      0.56      0.57       200
           2       0.97      0.78      0.86       200
           3       0.63      0.83      0.72       200
           4       0.69      0.92      0.79       200
           5       0.65      0.74      0.69       200
           6       0.99      0.84      0.91       200
           7       1.00      0.95      0.98       200
           8       0.79      0.91      0.84       200
           9       0.98      0.74      0.85       200
          10       1.00      0.52      0.68       200
          11       0.89      0.82      0.86       200
          12       0.71      0.93      0.80       200
          13       0.98      0.56      0.72       200
          14       0.95      0.72      0.82       200

    accuracy                           0.78      3000
   macro avg       0.82      0.78      0.79      3000
weighted avg       0.82      0.78      0.79      3000
```

Comentario: El model_3cnn_batch_norma converge a partir de 32 épocas (en lugar de las 50 fijadas) gracias al `early stopping`.

8 6. Estrategia 2: Red pre-entrenada

La segunda estrategia se basa en la utilización de una redes pre-entrenadas con el dataset ImageNet, llevando a cabo tareas de transfer learning y fine-tuning para resolver la tarea de clasificación.

Esta estrategia se divide en tres sub estrategias de entrenamiento:

- **Transfer Learning**: con dataset el original hacemos uso de características de alto nivel entrenadas en un perceptron multicapa de salida (top_model)transfiriendo directamente características de bajo nivel genéricas del *computer vision* dadas por la red prentrenada (base_model), cuyos pesos permancen congelados.
- **Fine Tuning parcial**: se descongela y entrena la red pre-entrenada (base_model) a partir de una capa convolucional dada junto con el perceptron multicapa de salida (top_model).
- **Fine Tuning completo**: se desconjela y entrena la red pre-entrenada (base_model)completamente junto con el perceptron multicapa de salida (top_model).

Se espera el uso de todas las técnicas de regularización mostradas en clase de forma justificada para la mejora del rendimiento de la red neuronal (weight regularization, dropout, batch normalization, data augmentation, etc.).

La función **get_base_model** definida a continuación devuelve el base_model y sus pesos para una arquitectura dada pre-entrenada con el dataset imagenet junto con el pre-procesamiento de los datos de entrada adaptado a la arquitectura, será reutilizada en todos los experimentos de esta sección.

```
[ ]: from tensorflow.keras.applications import vgg16, vgg19, resnet50, xception,␣
      ↪inception_v3, inception_resnet_v2, mobilenet_v2, densenet, nasnet

     # Cargar el base model y preprocesar los datos acorde a la red preentrenada
     def get_base_model(base, x_train, x_val, x_test, weights="imagenet",␣
      ↪include_top=False):
         input_shape = (x_train.shape[1], x_train.shape[2], x_train.shape[3])
         preprocess = None

         if base == 'VGG16':
             base_model = vgg16.VGG16(weights=weights, include_top=include_top,␣
      ↪input_shape=input_shape)
             preprocess = vgg16.preprocess_input

         elif base == 'VGG19':
             base_model = vgg19.VGG19(weights=weights, include_top=include_top,␣
      ↪input_shape=input_shape)
             preprocess = vgg19.preprocess_input

         elif base == 'ResNet50':
             base_model = resnet50.ResNet50(weights=weights,␣
      ↪include_top=include_top, input_shape=input_shape)
             preprocess = resnet50.preprocess_input

         elif base == 'Xception':
             base_model = xception.Xception(weights=weights,␣
      ↪include_top=include_top, input_shape=input_shape)
             preprocess = xception.preprocess_input

         elif base == 'InceptionV3':
             base_model = inception_v3.InceptionV3(weights=weights,␣
      ↪include_top=include_top, input_shape=input_shape)
             preprocess = inception_v3.preprocess_input

         elif base == 'InceptionResNetV2':
             base_model = inception_resnet_v2.InceptionResNetV2(weights=weights,␣
      ↪include_top=include_top, input_shape=input_shape)
             preprocess = inception_resnet_v2.preprocess_input

         elif base == 'MobileNetV2':
             base_model = mobilenet_v2.MobileNetV2(weights=weights,␣
      ↪include_top=include_top, input_shape=input_shape)
             preprocess = mobilenet_v2.preprocess_input

         elif base == 'DenseNet':
```

```
        base_model = densenet.DenseNet121(weights=weights,␣
↪include_top=include_top, input_shape=input_shape)
        preprocess = densenet.preprocess_input

    elif base == 'ResNet':
        base_model = resnet50.ResNet101(weights=weights,␣
↪include_top=include_top, input_shape=input_shape)
        preprocess = resnet50.preprocess_input

    elif base == 'NASNetLarge':
        base_model = nasnet.NASNetLarge(weights=weights,␣
↪include_top=include_top, input_shape=input_shape)
        preprocess = nasnet.preprocess_input

    if preprocess is not None:
        # Aplicar el preprocesamiento a los datos de entrenamiento, validación␣
↪y test
        x_train_preprocessed = preprocess(x_train)
        x_val_preprocessed = preprocess(x_val)
        x_test_preprocessed = preprocess(x_test)
    else:
        raise ValueError(f"Preprocesamiento para {base} no encontrado")

    base_model.summary()

    return base_model, x_train_preprocessed, x_val_preprocessed,␣
↪x_test_preprocessed
```

##**6.1 Transfer Learning**

La función **get_pretrained_model** definida a continuación recibe un base_model, junto con el tamaño de la capa densa del top_model y el numero de clases de salida conectando ambas redes para realizar la tarea de transfer learning. Dado que estamos en una tarea de transfer learning, los pesos del base_model se mantienen congelados.

[ ]:

```
def get_pretrained_model(base_model, dense_size, num_clases):

    # No entrenamos el base model
    base_model.trainable = False

    # Conectar el modelo con el top model
    pre_trained_model = Sequential()
    pre_trained_model.add(base_model)
    pre_trained_model.add(layers.Flatten())
    pre_trained_model.add(layers.Dense(dense_size, activation="relu"))
    pre_trained_model.add(layers.Dense(num_clases, activation="softmax"))
```

```
  pre_trained_model.summary()

  return pre_trained_model
```

###**6.1.1. VGG16**

a) Original

[ ]:
```
# Obtención modelo pre entrenado VGG16

base = 'VGG16'
base_model, x_train_preprocessed, x_val_preprocessed, x_test_preprocessed =␣
 ↪get_base_model(base, x_train, x_val, x_test)

pretrain_exp = base + "_TL_" + str(batch_size) + "_" + str(epochs)
exp_set.add(pretrain_exp)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 2s 0us/step
Model: "vgg16"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 75, 75, 3)]       0

 block1_conv1 (Conv2D)       (None, 75, 75, 64)        1792

 block1_conv2 (Conv2D)       (None, 75, 75, 64)        36928

 block1_pool (MaxPooling2D)  (None, 37, 37, 64)        0

 block2_conv1 (Conv2D)       (None, 37, 37, 128)       73856

 block2_conv2 (Conv2D)       (None, 37, 37, 128)       147584

 block2_pool (MaxPooling2D)  (None, 18, 18, 128)       0

 block3_conv1 (Conv2D)       (None, 18, 18, 256)       295168

 block3_conv2 (Conv2D)       (None, 18, 18, 256)       590080

 block3_conv3 (Conv2D)       (None, 18, 18, 256)       590080

 block3_pool (MaxPooling2D)  (None, 9, 9, 256)         0

 block4_conv1 (Conv2D)       (None, 9, 9, 512)         1180160
 block4_conv2 (Conv2D)       (None, 9, 9, 512)         2359808

 block4_conv3 (Conv2D)       (None, 9, 9, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

```
# Entrenamiento modelo pre entrenado

if do_training == True:

  pre_trained_model = get_pretrained_model(base_model, dense_size, num_clases)

  pre_trained_model.compile(optimizer=Adam(learning_rate=learning_rate),
                            loss="categorical_crossentropy",
                            metrics=["accuracy"])

  print("[INFO]: Entrenando Top Model sobre ", base, " ...")

  H = pre_trained_model.fit(x_train_preprocessed, y_train_ohe,
                            batch_size=batch_size,
                            epochs=epochs,
                            steps_per_epoch=x_train_preprocessed.shape[0] //␣
 ↪batch_size,
                            validation_data=(x_val_preprocessed, y_val_ohe),
                            callbacks=[early_stopping_cbck])

  save_trained_model(model = pre_trained_model, history = H, model_name =␣
 ↪pretrain_exp)

else:
  print("[INFO]: Cargando Top Model sobre " + base + "....")
  pre_trained_model = load_keras_model(pretrain_exp)
  pre_trained_model.summary()
```

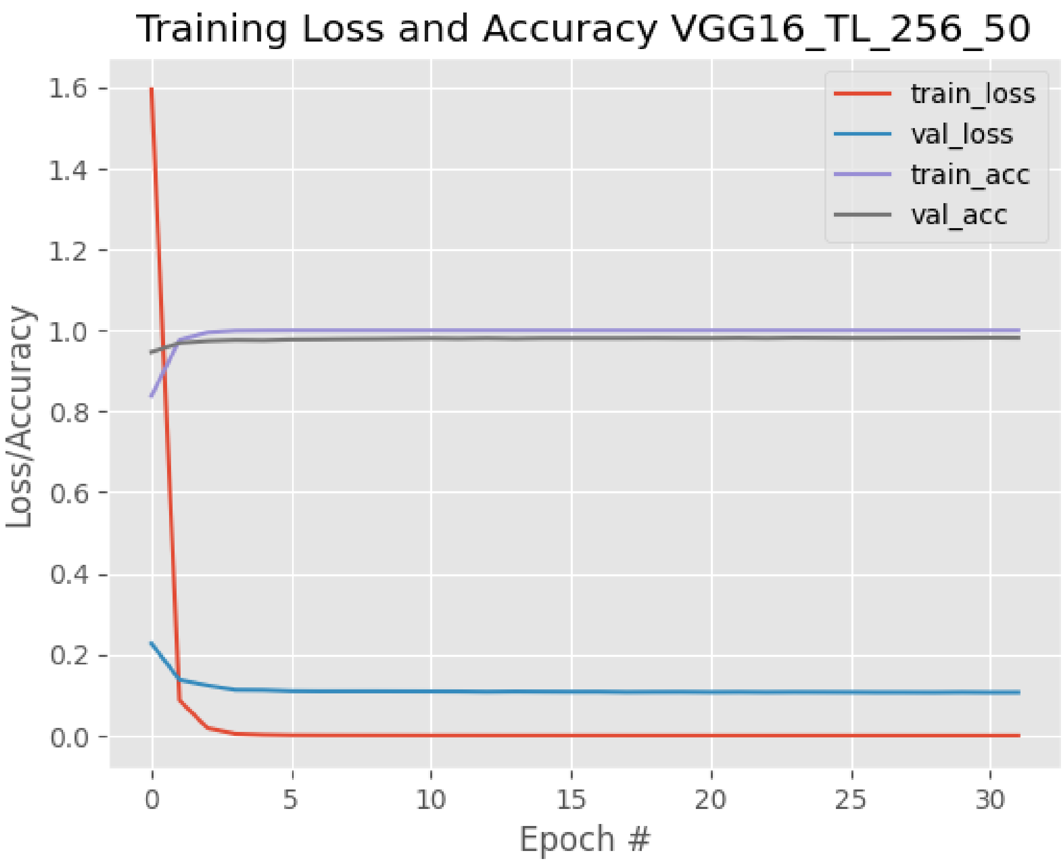
```
H = load_history(pretrain_exp)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 2, 2, 512)         14714688

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 128)               262272

 dense_1 (Dense)             (None, 15)                1935

=================================================================
Total params: 14978895 (57.14 MB)
Trainable params: 264207 (1.01 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
[INFO]: Entrenando Top Model sobre  VGG16  …
Epoch 1/50
58/58 [==============================] - 25s 210ms/step - loss: 1.5944 -
accuracy: 0.8382 - val_loss: 0.2271 - val_accuracy: 0.9467
Epoch 2/50
58/58 [==============================] - 12s 175ms/step - loss: 0.0869 -
accuracy: 0.9764 - val_loss: 0.1376 - val_accuracy: 0.9687
Epoch 3/50
58/58 [==============================] - 10s 173ms/step - loss: 0.0187 -
accuracy: 0.9948 - val_loss: 0.1236 - val_accuracy: 0.9740
Epoch 4/50
58/58 [==============================] - 10s 172ms/step - loss: 0.0044 -
accuracy: 0.9993 - val_loss: 0.1126 - val_accuracy: 0.9763
Epoch 5/50
58/58 [==============================] - 9s 158ms/step - loss: 0.0021 -
accuracy: 0.9998 - val_loss: 0.1121 - val_accuracy: 0.9757
Epoch 6/50
58/58 [==============================] - 9s 155ms/step - loss: 9.6892e-04 -
accuracy: 1.0000 - val_loss: 0.1096 - val_accuracy: 0.9777
Epoch 7/50
58/58 [==============================] - 10s 176ms/step - loss: 7.0305e-04 -
accuracy: 1.0000 - val_loss: 0.1091 - val_accuracy: 0.9780
Epoch 8/50
58/58 [==============================] - 10s 176ms/step - loss: 6.0664e-04 -
accuracy: 1.0000 - val_loss: 0.1091 - val_accuracy: 0.9787
Epoch 9/50
58/58 [==============================] - 10s 175ms/step - loss: 5.0599e-04 -
accuracy: 1.0000 - val_loss: 0.1091 - val_accuracy: 0.9790
```

```
Epoch 10/50
58/58 [==============================] - 9s 160ms/step - loss: 4.3196e-04 -
accuracy: 1.0000 - val_loss: 0.1090 - val_accuracy: 0.9797
Epoch 11/50
58/58 [==============================] - 10s 176ms/step - loss: 3.4579e-04 -
accuracy: 1.0000 - val_loss: 0.1088 - val_accuracy: 0.9803
Epoch 12/50
58/58 [==============================] - 9s 160ms/step - loss: 3.3994e-04 -
accuracy: 1.0000 - val_loss: 0.1087 - val_accuracy: 0.9797
Epoch 13/50
58/58 [==============================] - 10s 180ms/step - loss: 2.8698e-04 -
accuracy: 1.0000 - val_loss: 0.1080 - val_accuracy: 0.9807
Epoch 14/50
58/58 [==============================] - 10s 177ms/step - loss: 2.5208e-04 -
accuracy: 1.0000 - val_loss: 0.1086 - val_accuracy: 0.9797
Epoch 15/50
58/58 [==============================] - 10s 179ms/step - loss: 2.4544e-04 -
accuracy: 1.0000 - val_loss: 0.1082 - val_accuracy: 0.9807
Epoch 16/50
58/58 [==============================] - 9s 162ms/step - loss: 2.1388e-04 -
accuracy: 1.0000 - val_loss: 0.1080 - val_accuracy: 0.9807
Epoch 17/50
58/58 [==============================] - 10s 178ms/step - loss: 2.0132e-04 -
accuracy: 1.0000 - val_loss: 0.1080 - val_accuracy: 0.9807
Epoch 18/50
58/58 [==============================] - 9s 163ms/step - loss: 1.7798e-04 -
accuracy: 1.0000 - val_loss: 0.1072 - val_accuracy: 0.9807
Epoch 19/50
58/58 [==============================] - 9s 159ms/step - loss: 1.5997e-04 -
accuracy: 1.0000 - val_loss: 0.1075 - val_accuracy: 0.9810
Epoch 20/50
58/58 [==============================] - 9s 162ms/step - loss: 1.5110e-04 -
accuracy: 1.0000 - val_loss: 0.1076 - val_accuracy: 0.9807
Epoch 21/50
58/58 [==============================] - 10s 178ms/step - loss: 1.3862e-04 -
accuracy: 1.0000 - val_loss: 0.1071 - val_accuracy: 0.9807
Epoch 22/50
58/58 [==============================] - 9s 159ms/step - loss: 1.3510e-04 -
accuracy: 1.0000 - val_loss: 0.1073 - val_accuracy: 0.9813
Epoch 23/50
58/58 [==============================] - 9s 162ms/step - loss: 1.2755e-04 -
accuracy: 1.0000 - val_loss: 0.1069 - val_accuracy: 0.9807
Epoch 24/50
58/58 [==============================] - 10s 177ms/step - loss: 1.1329e-04 -
accuracy: 1.0000 - val_loss: 0.1072 - val_accuracy: 0.9817
Epoch 25/50
58/58 [==============================] - 10s 179ms/step - loss: 1.0759e-04 -
accuracy: 1.0000 - val_loss: 0.1071 - val_accuracy: 0.9813
```

```
Epoch 26/50
58/58 [==============================] - 10s 180ms/step - loss: 9.9146e-05 -
accuracy: 1.0000 - val_loss: 0.1069 - val_accuracy: 0.9810
Epoch 27/50
58/58 [==============================] - 9s 159ms/step - loss: 9.7297e-05 -
accuracy: 1.0000 - val_loss: 0.1066 - val_accuracy: 0.9810
Epoch 28/50
58/58 [==============================] - 10s 180ms/step - loss: 8.4229e-05 -
accuracy: 1.0000 - val_loss: 0.1065 - val_accuracy: 0.9813
Epoch 29/50
58/58 [==============================] - 9s 161ms/step - loss: 8.3483e-05 -
accuracy: 1.0000 - val_loss: 0.1063 - val_accuracy: 0.9813
Epoch 30/50
58/58 [==============================] - 10s 179ms/step - loss: 8.1674e-05 -
accuracy: 1.0000 - val_loss: 0.1068 - val_accuracy: 0.9817
Epoch 31/50
58/58 [==============================] - 10s 180ms/step - loss: 7.4965e-05 -
accuracy: 1.0000 - val_loss: 0.1064 - val_accuracy: 0.9820
Epoch 32/50
58/58 [==============================] - 10s 177ms/step - loss: 7.1286e-05 -
accuracy: 1.0000 - val_loss: 0.1065 - val_accuracy: 0.9817
Saved model to disk
```

```
[ ]: visualize_learning_curve(H, lb =  pretrain_exp)
```



```
[ ]: # Evaluando modelo pre entrenado

     evaluate_model(pre_trained_model, x_test_preprocessed, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 2s 67ms/step
              precision    recall  f1-score   support

           0       0.99      0.96      0.97       200
           1       0.95      0.95      0.95       200
           2       0.98      0.99      0.99       200
           3       0.97      0.97      0.97       200
           4       1.00      1.00      1.00       200
           5       0.96      0.95      0.96       200
           6       0.98      0.96      0.97       200
           7       0.97      0.97      0.97       200
           8       1.00      0.99      0.99       200
           9       0.97      0.97      0.97       200
          10       0.99      0.99      0.99       200
          11       0.99      0.99      0.99       200
          12       0.96      0.98      0.97       200
          13       0.97      0.98      0.97       200
          14       0.96      0.97      0.97       200

    accuracy                           0.98      3000
   macro avg       0.98      0.98      0.98      3000
weighted avg       0.98      0.98      0.98      3000
```

b) Data Augmentation

[ ]:

```
# Entrenamiento modelo pre entrenado con data augmentation
pretrain_exp = base + "_TL_AUG_" + str(batch_size) + "_" + str(epochs)
exp_set.add(pretrain_exp)

if do_training == True:

  pre_trained_model_aug = get_pretrained_model(base_model, dense_size,␣
 ↪num_clases)

  pre_trained_model_aug.compile(optimizer=Adam(learning_rate=learning_rate),
                                loss="categorical_crossentropy",
                                metrics=["accuracy"])

  print("[INFO]: Entrenando Top Model sobre ", base, " con data augmentation")
  H_aug = pre_trained_model_aug.fit(train_generator,
                                    steps_per_epoch=train_generator.samples //␣
 ↪batch_size_aug,
                                    validation_data=val_generator,
                                    validation_steps=val_generator.samples //␣
 ↪batch_size_aug,
                                    epochs=epochs,
                                    callbacks=[early_stopping_cbck])

  save_trained_model(model = pre_trained_model, history = H, model_name =␣
 ↪pretrain_exp)

else:
  print("[INFO]: Cargando Top Model sobre " +  base, " con data augmentation")
  pre_trained_model_aug = load_keras_model(pretrain_exp)
  pre_trained_model_aug.summary()
  H_aug = load_history(pretrain_exp)
```

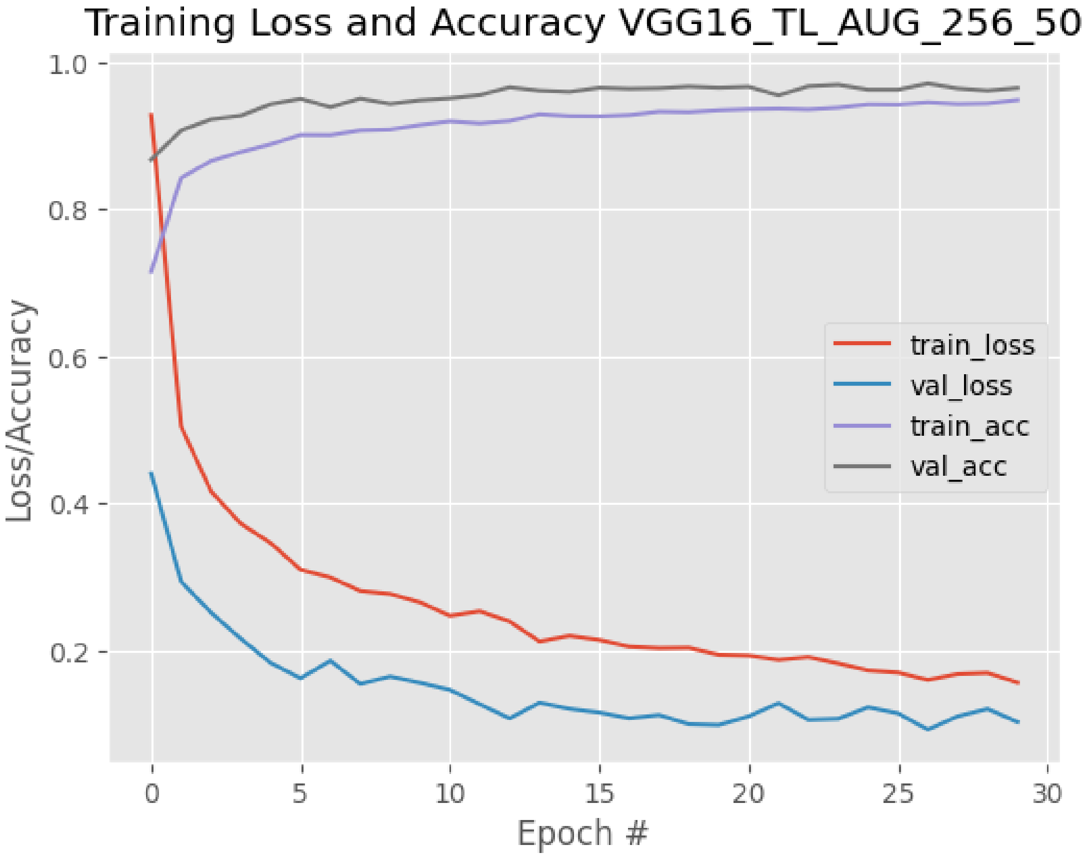
```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 2, 2, 512)         14714688
 flatten_1 (Flatten)        (None, 2048)              0

 dense_2 (Dense)            (None, 128)               262272

 dense_3 (Dense)            (None, 15)                1935

=================================================================
Total params: 14978895 (57.14 MB)
Trainable params: 264207 (1.01 MB)
Non-trainable params: 14714688 (56.13 MB)
-----------------------------------------------------------------
[INFO]: Entrenando Top Model sobre  VGG16  con data augmentation
Epoch 1/50
468/468 [==============================] - 65s 135ms/step - loss: 0.9278 -
accuracy: 0.7161 - val_loss: 0.4412 - val_accuracy: 0.8693
Epoch 2/50
468/468 [==============================] - 53s 113ms/step - loss: 0.5077 -
accuracy: 0.8438 - val_loss: 0.2664 - val_accuracy: 0.9204
Epoch 3/50
468/468 [==============================] - 53s 113ms/step - loss: 0.4121 -
accuracy: 0.8676 - val_loss: 0.2553 - val_accuracy: 0.9217
Epoch 4/50
468/468 [==============================] - 55s 117ms/step - loss: 0.3601 -
accuracy: 0.8885 - val_loss: 0.2021 - val_accuracy: 0.9372
Epoch 5/50
468/468 [==============================] - 55s 119ms/step - loss: 0.3436 -
accuracy: 0.8880 - val_loss: 0.1933 - val_accuracy: 0.9395
Epoch 6/50
468/468 [==============================] - 55s 117ms/step - loss: 0.3107 -
accuracy: 0.9014 - val_loss: 0.1632 - val_accuracy: 0.9499
Epoch 7/50
468/468 [==============================] - 56s 120ms/step - loss: 0.2977 -
accuracy: 0.9010 - val_loss: 0.1698 - val_accuracy: 0.9432
Epoch 8/50
468/468 [==============================] - 57s 122ms/step - loss: 0.2731 -
accuracy: 0.9098 - val_loss: 0.1597 - val_accuracy: 0.9486
Epoch 9/50
468/468 [==============================] - 63s 134ms/step - loss: 0.2689 -
accuracy: 0.9112 - val_loss: 0.1385 - val_accuracy: 0.9570
Epoch 10/50
468/468 [==============================] - 60s 129ms/step - loss: 0.2698 -
accuracy: 0.9115 - val_loss: 0.1576 - val_accuracy: 0.9483
Epoch 11/50
468/468 [==============================] - 59s 125ms/step - loss: 0.2491 -
accuracy: 0.9179 - val_loss: 0.1491 - val_accuracy: 0.9513
Epoch 12/50
468/468 [==============================] - 65s 138ms/step - loss: 0.2466 -
```

```
accuracy: 0.9189 - val_loss: 0.1175 - val_accuracy: 0.9627
Epoch 13/50
468/468 [==============================] - 54s 116ms/step - loss: 0.2324 -
accuracy: 0.9244 - val_loss: 0.1430 - val_accuracy: 0.9513
Epoch 14/50
468/468 [==============================] - 61s 130ms/step - loss: 0.2248 -
accuracy: 0.9249 - val_loss: 0.1426 - val_accuracy: 0.9550
Epoch 15/50
468/468 [==============================] - 66s 140ms/step - loss: 0.2168 -
accuracy: 0.9270 - val_loss: 0.1205 - val_accuracy: 0.9617
Epoch 16/50
468/468 [==============================] - 59s 127ms/step - loss: 0.2169 -
accuracy: 0.9246 - val_loss: 0.1297 - val_accuracy: 0.9593
Epoch 17/50
468/468 [==============================] - 56s 120ms/step - loss: 0.2089 -
accuracy: 0.9321 - val_loss: 0.1142 - val_accuracy: 0.9627
Epoch 18/50
468/468 [==============================] - 54s 116ms/step - loss: 0.2073 -
accuracy: 0.9291 - val_loss: 0.1053 - val_accuracy: 0.9671
Epoch 19/50
468/468 [==============================] - 53s 114ms/step - loss: 0.1937 -
accuracy: 0.9380 - val_loss: 0.1215 - val_accuracy: 0.9620
Epoch 20/50
468/468 [==============================] - 53s 114ms/step - loss: 0.1948 -
accuracy: 0.9364 - val_loss: 0.1008 - val_accuracy: 0.9694
Epoch 21/50
468/468 [==============================] - 54s 115ms/step - loss: 0.1936 -
accuracy: 0.9346 - val_loss: 0.1134 - val_accuracy: 0.9664
Epoch 22/50
468/468 [==============================] - 56s 119ms/step - loss: 0.1920 -
accuracy: 0.9382 - val_loss: 0.0927 - val_accuracy: 0.9684
Epoch 23/50
468/468 [==============================] - 55s 117ms/step - loss: 0.1779 -
accuracy: 0.9405 - val_loss: 0.1084 - val_accuracy: 0.9654
Epoch 24/50
468/468 [==============================] - 54s 115ms/step - loss: 0.1794 -
accuracy: 0.9415 - val_loss: 0.0921 - val_accuracy: 0.9708
Epoch 25/50
468/468 [==============================] - 54s 115ms/step - loss: 0.1685 -
accuracy: 0.9417 - val_loss: 0.1158 - val_accuracy: 0.9644
Epoch 26/50
468/468 [==============================] - 60s 129ms/step - loss: 0.1697 -
accuracy: 0.9410 - val_loss: 0.1145 - val_accuracy: 0.9671
Epoch 27/50
468/468 [==============================] - 80s 170ms/step - loss: 0.1666 -
accuracy: 0.9443 - val_loss: 0.0949 - val_accuracy: 0.9704
Epoch 28/50
468/468 [==============================] - 67s 143ms/step - loss: 0.1599 -
```

```
accuracy: 0.9451 - val_loss: 0.1201 - val_accuracy: 0.9587
Epoch 29/50
468/468 [==============================] - 73s 156ms/step - loss: 0.1637 -
accuracy: 0.9451 - val_loss: 0.0850 - val_accuracy: 0.9741
Epoch 30/50
468/468 [==============================] - 64s 136ms/step - loss: 0.1540 -
accuracy: 0.9482 - val_loss: 0.0902 - val_accuracy: 0.9714
Epoch 31/50
468/468 [==============================] - 69s 147ms/step - loss: 0.1596 -
accuracy: 0.9453 - val_loss: 0.0994 - val_accuracy: 0.9657
Epoch 32/50
468/468 [==============================] - 80s 171ms/step - loss: 0.1551 -
accuracy: 0.9482 - val_loss: 0.0929 - val_accuracy: 0.9721
Saved model to disk
```

[ ]:

```
visualize_learning_curve(H_aug, lb = pretrain_exp)
```



[ ]:

```
# Evaluar el modelo
evaluate_model_aug(pre_trained_model_aug, test_generator)
```

```
[INFO]: Evaluando red neuronal…
```

```
<ipython-input-4-e54c28b45500>:27: UserWarning: `Model.predict_generator` is
deprecated and will be removed in a future version. Please use `Model.predict`,
which supports generators.
  predictions = model.predict_generator(generator, steps=len(generator))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       200
           1       0.98      0.97      0.98       200
           2       1.00      1.00      1.00       200
           3       0.88      0.99      0.93       200
           4       0.96      1.00      0.98       200
           5       0.98      0.94      0.96       200
           6       0.99      1.00      1.00       200
           7       0.99      0.98      0.99       200
           8       0.97      0.96      0.97       200
           9       0.98      0.98      0.98       200
          10       0.99      0.90      0.94       200
          11       0.98      0.99      0.99       200
          12       0.93      0.95      0.94       200
          13       0.98      0.99      0.99       200
          14       0.99      0.94      0.97       200

    accuracy                           0.97      3000
   macro avg       0.97      0.97      0.97      3000
weighted avg       0.97      0.97      0.97      3000
```

###**6.1.2. Xception**

[ ]:

```
# Obtención modelo pre entrenado Xception

base = 'Xception'
base_model, x_train_preprocessed, x_val_preprocessed, x_test_preprocessed =␣
  ↪get_base_model(base, x_train, x_val, x_test)

pretrain_exp = base + "_TL_" + str(batch_size) + "_" + str(epochs)
exp_set.add(pretrain_exp)
```

```
Model: "xception"
________________________________________________________________________________
__________________
 Layer (type)                   Output Shape          Param #     Connected to
================================================================================
==================
 input_4 (InputLayer)           [(None, 75, 75, 3)]   0           []
 block1_conv1 (Conv2D)       (None, 37, 37, 32)           864
['input_4[0][0]']

 block1_conv1_bn (BatchNorm  (None, 37, 37, 32)           128
['block1_conv1[0][0]']
 alization)

 block1_conv1_act (Activati  (None, 37, 37, 32)           0
['block1_conv1_bn[0][0]']
 on)

 block1_conv2 (Conv2D)       (None, 35, 35, 64)           18432
['block1_conv1_act[0][0]']

 block1_conv2_bn (BatchNorm  (None, 35, 35, 64)           256
['block1_conv2[0][0]']
 alization)

 block1_conv2_act (Activati  (None, 35, 35, 64)           0
['block1_conv2_bn[0][0]']
 on)

 block2_sepconv1 (Separable  (None, 35, 35, 128)          8768
['block1_conv2_act[0][0]']
 Conv2D)

 block2_sepconv1_bn (BatchN  (None, 35, 35, 128)          512
['block2_sepconv1[0][0]']
 ormalization)

 block2_sepconv2_act (Activ  (None, 35, 35, 128)          0
['block2_sepconv1_bn[0][0]']
 ation)

 block2_sepconv2 (Separable  (None, 35, 35, 128)          17536
['block2_sepconv2_act[0][0]']
 Conv2D)

 block2_sepconv2_bn (BatchN  (None, 35, 35, 128)          512
['block2_sepconv2[0][0]']
 ormalization)

 conv2d_8 (Conv2D)           (None, 18, 18, 128)          8192
['block1_conv2_act[0][0]']

 block2_pool (MaxPooling2D)  (None, 18, 18, 128)          0
['block2_sepconv2_bn[0][0]']
```

```
 batch_normalization_8 (Bat  (None, 18, 18, 128)         512
['conv2d_8[0][0]']
 chNormalization)

 add_24 (Add)                (None, 18, 18, 128)         0
['block2_pool[0][0]',
'batch_normalization_8[0][0]'
                                                                    ]

 block3_sepconv1_act (Activ  (None, 18, 18, 128)         0
['add_24[0][0]']
 ation)

 block3_sepconv1 (Separable  (None, 18, 18, 256)         33920
['block3_sepconv1_act[0][0]']
 Conv2D)

 block3_sepconv1_bn (BatchN  (None, 18, 18, 256)         1024
['block3_sepconv1[0][0]']
 ormalization)

 block3_sepconv2_act (Activ  (None, 18, 18, 256)         0
['block3_sepconv1_bn[0][0]']
 ation)

 block3_sepconv2 (Separable  (None, 18, 18, 256)         67840
['block3_sepconv2_act[0][0]']
 Conv2D)

 block3_sepconv2_bn (BatchN  (None, 18, 18, 256)         1024
['block3_sepconv2[0][0]']
 ormalization)

 conv2d_9 (Conv2D)           (None, 9, 9, 256)           32768
['add_24[0][0]']

 block3_pool (MaxPooling2D)  (None, 9, 9, 256)           0
['block3_sepconv2_bn[0][0]']

 batch_normalization_9 (Bat  (None, 9, 9, 256)           1024
['conv2d_9[0][0]']
 chNormalization)

 add_25 (Add)                (None, 9, 9, 256)           0
['block3_pool[0][0]',
'batch_normalization_9[0][0]'
                                                                    ]
 block4_sepconv1_act (Activ  (None, 9, 9, 256)           0
['add_25[0][0]']
 ation)

 block4_sepconv1 (Separable  (None, 9, 9, 728)           188672
['block4_sepconv1_act[0][0]']
 Conv2D)

 block4_sepconv1_bn (BatchN  (None, 9, 9, 728)           2912
['block4_sepconv1[0][0]']
 ormalization)

 block4_sepconv2_act (Activ  (None, 9, 9, 728)           0
['block4_sepconv1_bn[0][0]']
 ation)

 block4_sepconv2 (Separable  (None, 9, 9, 728)           536536
['block4_sepconv2_act[0][0]']
 Conv2D)

 block4_sepconv2_bn (BatchN  (None, 9, 9, 728)           2912
['block4_sepconv2[0][0]']
 ormalization)

 conv2d_10 (Conv2D)          (None, 5, 5, 728)           186368
['add_25[0][0]']

 block4_pool (MaxPooling2D)  (None, 5, 5, 728)           0
['block4_sepconv2_bn[0][0]']

 batch_normalization_10 (Ba  (None, 5, 5, 728)           2912
['conv2d_10[0][0]']
 tchNormalization)

 add_26 (Add)                (None, 5, 5, 728)           0
['block4_pool[0][0]',
'batch_normalization_10[0][0]
                                                                     ']

 block5_sepconv1_act (Activ  (None, 5, 5, 728)           0
['add_26[0][0]']
 ation)

 block5_sepconv1 (Separable  (None, 5, 5, 728)           536536
['block5_sepconv1_act[0][0]']
 Conv2D)

 block5_sepconv1_bn (BatchN  (None, 5, 5, 728)           2912
['block5_sepconv1[0][0]']
 ormalization)

 block5_sepconv2_act (Activ  (None, 5, 5, 728)            0
['block5_sepconv1_bn[0][0]']
 ation)

 block5_sepconv2 (Separable  (None, 5, 5, 728)            536536
['block5_sepconv2_act[0][0]']
 Conv2D)

 block5_sepconv2_bn (BatchN  (None, 5, 5, 728)            2912
['block5_sepconv2[0][0]']
 ormalization)

 block5_sepconv3_act (Activ  (None, 5, 5, 728)            0
['block5_sepconv2_bn[0][0]']
 ation)

 block5_sepconv3 (Separable  (None, 5, 5, 728)            536536
['block5_sepconv3_act[0][0]']
 Conv2D)

 block5_sepconv3_bn (BatchN  (None, 5, 5, 728)            2912
['block5_sepconv3[0][0]']
 ormalization)

 add_27 (Add)                (None, 5, 5, 728)            0
['block5_sepconv3_bn[0][0]',
'add_26[0][0]']

 block6_sepconv1_act (Activ  (None, 5, 5, 728)            0
['add_27[0][0]']
 ation)

 block6_sepconv1 (Separable  (None, 5, 5, 728)            536536
['block6_sepconv1_act[0][0]']
 Conv2D)

 block6_sepconv1_bn (BatchN  (None, 5, 5, 728)            2912
['block6_sepconv1[0][0]']
 ormalization)

 block6_sepconv2_act (Activ  (None, 5, 5, 728)            0
['block6_sepconv1_bn[0][0]']
 ation)

 block6_sepconv2 (Separable  (None, 5, 5, 728)            536536
['block6_sepconv2_act[0][0]']
 Conv2D)

 block6_sepconv2_bn (BatchN  (None, 5, 5, 728)            2912
['block6_sepconv2[0][0]']
 ormalization)

 block6_sepconv3_act (Activ  (None, 5, 5, 728)            0
['block6_sepconv2_bn[0][0]']
 ation)

 block6_sepconv3 (Separable  (None, 5, 5, 728)            536536
['block6_sepconv3_act[0][0]']
 Conv2D)

 block6_sepconv3_bn (BatchN  (None, 5, 5, 728)            2912
['block6_sepconv3[0][0]']
 ormalization)

 add_28 (Add)                (None, 5, 5, 728)            0
['block6_sepconv3_bn[0][0]',
'add_27[0][0]']

 block7_sepconv1_act (Activ  (None, 5, 5, 728)            0
['add_28[0][0]']
 ation)

 block7_sepconv1 (Separable  (None, 5, 5, 728)            536536
['block7_sepconv1_act[0][0]']
 Conv2D)

 block7_sepconv1_bn (BatchN  (None, 5, 5, 728)            2912
['block7_sepconv1[0][0]']
 ormalization)

 block7_sepconv2_act (Activ  (None, 5, 5, 728)            0
['block7_sepconv1_bn[0][0]']
 ation)

 block7_sepconv2 (Separable  (None, 5, 5, 728)            536536
['block7_sepconv2_act[0][0]']
 Conv2D)

 block7_sepconv2_bn (BatchN  (None, 5, 5, 728)            2912
['block7_sepconv2[0][0]']
 ormalization)

 block7_sepconv3_act (Activ  (None, 5, 5, 728)            0
['block7_sepconv2_bn[0][0]']
 ation)

 block7_sepconv3 (Separable  (None, 5, 5, 728)            536536
['block7_sepconv3_act[0][0]']
 Conv2D)

 block7_sepconv3_bn (BatchN  (None, 5, 5, 728)            2912
['block7_sepconv3[0][0]']
 ormalization)

 add_29 (Add)                (None, 5, 5, 728)            0
['block7_sepconv3_bn[0][0]',
'add_28[0][0]']

 block8_sepconv1_act (Activ  (None, 5, 5, 728)            0
['add_29[0][0]']
 ation)

 block8_sepconv1 (Separable  (None, 5, 5, 728)            536536
['block8_sepconv1_act[0][0]']
 Conv2D)

 block8_sepconv1_bn (BatchN  (None, 5, 5, 728)            2912
['block8_sepconv1[0][0]']
 ormalization)

 block8_sepconv2_act (Activ  (None, 5, 5, 728)            0
['block8_sepconv1_bn[0][0]']
 ation)

 block8_sepconv2 (Separable  (None, 5, 5, 728)            536536
['block8_sepconv2_act[0][0]']
 Conv2D)

 block8_sepconv2_bn (BatchN  (None, 5, 5, 728)            2912
['block8_sepconv2[0][0]']
 ormalization)

 block8_sepconv3_act (Activ  (None, 5, 5, 728)            0
['block8_sepconv2_bn[0][0]']
 ation)

 block8_sepconv3 (Separable  (None, 5, 5, 728)            536536
['block8_sepconv3_act[0][0]']
 Conv2D)

 block8_sepconv3_bn (BatchN  (None, 5, 5, 728)            2912
['block8_sepconv3[0][0]']
 ormalization)

 add_30 (Add)                (None, 5, 5, 728)            0
['block8_sepconv3_bn[0][0]',
'add_29[0][0]']

 block9_sepconv1_act (Activ  (None, 5, 5, 728)            0
['add_30[0][0]']
 ation)

 block9_sepconv1 (Separable  (None, 5, 5, 728)            536536
['block9_sepconv1_act[0][0]']
 Conv2D)

 block9_sepconv1_bn (BatchN  (None, 5, 5, 728)            2912
['block9_sepconv1[0][0]']
 ormalization)

 block9_sepconv2_act (Activ  (None, 5, 5, 728)            0
['block9_sepconv1_bn[0][0]']
 ation)

 block9_sepconv2 (Separable  (None, 5, 5, 728)            536536
['block9_sepconv2_act[0][0]']
 Conv2D)

 block9_sepconv2_bn (BatchN  (None, 5, 5, 728)            2912
['block9_sepconv2[0][0]']
 ormalization)

 block9_sepconv3_act (Activ  (None, 5, 5, 728)            0
['block9_sepconv2_bn[0][0]']
 ation)

 block9_sepconv3 (Separable  (None, 5, 5, 728)            536536
['block9_sepconv3_act[0][0]']
 Conv2D)

 block9_sepconv3_bn (BatchN  (None, 5, 5, 728)            2912
['block9_sepconv3[0][0]']
 ormalization)

 add_31 (Add)                (None, 5, 5, 728)            0
['block9_sepconv3_bn[0][0]',
'add_30[0][0]']

 block10_sepconv1_act (Acti  (None, 5, 5, 728)            0
['add_31[0][0]']
 vation)

 block10_sepconv1 (Separabl  (None, 5, 5, 728)            536536
['block10_sepconv1_act[0][0]']
 eConv2D)

 block10_sepconv1_bn (Batch  (None, 5, 5, 728)            2912
['block10_sepconv1[0][0]']
 Normalization)

 block10_sepconv2_act (Acti  (None, 5, 5, 728)            0
['block10_sepconv1_bn[0][0]']
 vation)

 block10_sepconv2 (Separabl  (None, 5, 5, 728)            536536
['block10_sepconv2_act[0][0]']
 eConv2D)

 block10_sepconv2_bn (Batch  (None, 5, 5, 728)            2912
['block10_sepconv2[0][0]']
 Normalization)

 block10_sepconv3_act (Acti  (None, 5, 5, 728)            0
['block10_sepconv2_bn[0][0]']
 vation)

 block10_sepconv3 (Separabl  (None, 5, 5, 728)            536536
['block10_sepconv3_act[0][0]']
 eConv2D)

 block10_sepconv3_bn (Batch  (None, 5, 5, 728)            2912
['block10_sepconv3[0][0]']
 Normalization)

 add_32 (Add)                (None, 5, 5, 728)            0
['block10_sepconv3_bn[0][0]',
'add_31[0][0]']

 block11_sepconv1_act (Acti  (None, 5, 5, 728)            0
['add_32[0][0]']
 vation)

 block11_sepconv1 (Separabl  (None, 5, 5, 728)            536536
['block11_sepconv1_act[0][0]']
 eConv2D)

 block11_sepconv1_bn (Batch  (None, 5, 5, 728)            2912
['block11_sepconv1[0][0]']
 Normalization)

 block11_sepconv2_act (Acti  (None, 5, 5, 728)            0
['block11_sepconv1_bn[0][0]']
 vation)

 block11_sepconv2 (Separabl  (None, 5, 5, 728)            536536
['block11_sepconv2_act[0][0]']
 eConv2D)

 block11_sepconv2_bn (Batch  (None, 5, 5, 728)            2912
['block11_sepconv2[0][0]']
 Normalization)

 block11_sepconv3_act (Acti  (None, 5, 5, 728)            0
['block11_sepconv2_bn[0][0]']
 vation)

 block11_sepconv3 (Separabl  (None, 5, 5, 728)            536536
['block11_sepconv3_act[0][0]']
 eConv2D)

 block11_sepconv3_bn (Batch  (None, 5, 5, 728)            2912
['block11_sepconv3[0][0]']
 Normalization)

 add_33 (Add)                (None, 5, 5, 728)            0
['block11_sepconv3_bn[0][0]',
'add_32[0][0]']

 block12_sepconv1_act (Acti  (None, 5, 5, 728)            0
['add_33[0][0]']
 vation)

 block12_sepconv1 (Separabl  (None, 5, 5, 728)            536536
['block12_sepconv1_act[0][0]']
 eConv2D)

 block12_sepconv1_bn (Batch  (None, 5, 5, 728)            2912
['block12_sepconv1[0][0]']
 Normalization)

 block12_sepconv2_act (Acti  (None, 5, 5, 728)            0
['block12_sepconv1_bn[0][0]']
 vation)

 block12_sepconv2 (Separabl  (None, 5, 5, 728)            536536
['block12_sepconv2_act[0][0]']
 eConv2D)

 block12_sepconv2_bn (Batch  (None, 5, 5, 728)            2912
['block12_sepconv2[0][0]']
 Normalization)

 block12_sepconv3_act (Acti  (None, 5, 5, 728)            0
['block12_sepconv2_bn[0][0]']
 vation)

 block12_sepconv3 (Separabl  (None, 5, 5, 728)            536536
['block12_sepconv3_act[0][0]']
 eConv2D)

 block12_sepconv3_bn (Batch  (None, 5, 5, 728)            2912
['block12_sepconv3[0][0]']
 Normalization)

 add_34 (Add)                (None, 5, 5, 728)            0
['block12_sepconv3_bn[0][0]',
'add_33[0][0]']

 block13_sepconv1_act (Acti  (None, 5, 5, 728)            0
['add_34[0][0]']
 vation)

 block13_sepconv1 (Separabl  (None, 5, 5, 728)            536536
['block13_sepconv1_act[0][0]']
 eConv2D)

 block13_sepconv1_bn (Batch  (None, 5, 5, 728)            2912
['block13_sepconv1[0][0]']
 Normalization)

 block13_sepconv2_act (Acti  (None, 5, 5, 728)            0
['block13_sepconv1_bn[0][0]']
 vation)

 block13_sepconv2 (Separabl  (None, 5, 5, 1024)           752024
['block13_sepconv2_act[0][0]']
 eConv2D)

 block13_sepconv2_bn (Batch  (None, 5, 5, 1024)           4096
['block13_sepconv2[0][0]']
 Normalization)

 conv2d_11 (Conv2D)          (None, 3, 3, 1024)           745472
```

```
['add_34[0][0]']

 block13_pool (MaxPooling2D  (None, 3, 3, 1024)           0
['block13_sepconv2_bn[0][0]']
 )

 batch_normalization_11 (Ba  (None, 3, 3, 1024)           4096
['conv2d_11[0][0]']
 tchNormalization)

 add_35 (Add)                (None, 3, 3, 1024)           0
['block13_pool[0][0]',
'batch_normalization_11[0][0]
                                                                    ']

 block14_sepconv1 (Separabl  (None, 3, 3, 1536)           1582080
['add_35[0][0]']
 eConv2D)

 block14_sepconv1_bn (Batch  (None, 3, 3, 1536)           6144
['block14_sepconv1[0][0]']
 Normalization)

 block14_sepconv1_act (Acti  (None, 3, 3, 1536)           0
['block14_sepconv1_bn[0][0]']
 vation)

 block14_sepconv2 (Separabl  (None, 3, 3, 2048)           3159552
['block14_sepconv1_act[0][0]']
 eConv2D)

 block14_sepconv2_bn (Batch  (None, 3, 3, 2048)           8192
['block14_sepconv2[0][0]']
 Normalization)

 block14_sepconv2_act (Acti  (None, 3, 3, 2048)           0
['block14_sepconv2_bn[0][0]']
 vation)

================================================================================
==================
Total params: 20861480 (79.58 MB)
Trainable params: 20806952 (79.37 MB)
Non-trainable params: 54528 (213.00 KB)
________________________________________________________________________________
__________________
```

```
[ ]: # Entrenamiento modelo pre entrenado

     if do_training == True:

       pre_trained_model = get_pretrained_model(base_model, dense_size, num_clases)

       pre_trained_model.compile(optimizer=Adam(learning_rate=learning_rate),
                                 loss="categorical_crossentropy",
                                 metrics=["accuracy"])

       print("[INFO]: Entrenando Top Model sobre ", base, " ...")

       H = pre_trained_model.fit(x_train_preprocessed, y_train_ohe,
                                 batch_size=batch_size,
                                 epochs=epochs,
                                 steps_per_epoch=x_train_preprocessed.shape[0] //␣
      ↪batch_size,
                                 validation_data=(x_val_preprocessed, y_val_ohe),
                                 callbacks=[early_stopping_cbck])

       save_trained_model(model = pre_trained_model, history = H, model_name =␣
      ↪pretrain_exp)

     else:
       print("[INFO]: Cargando Top Model sobre " + base + "....")
       pre_trained_model = load_keras_model(pretrain_exp)
       pre_trained_model.summary()
       H = load_history(pretrain_exp)
```

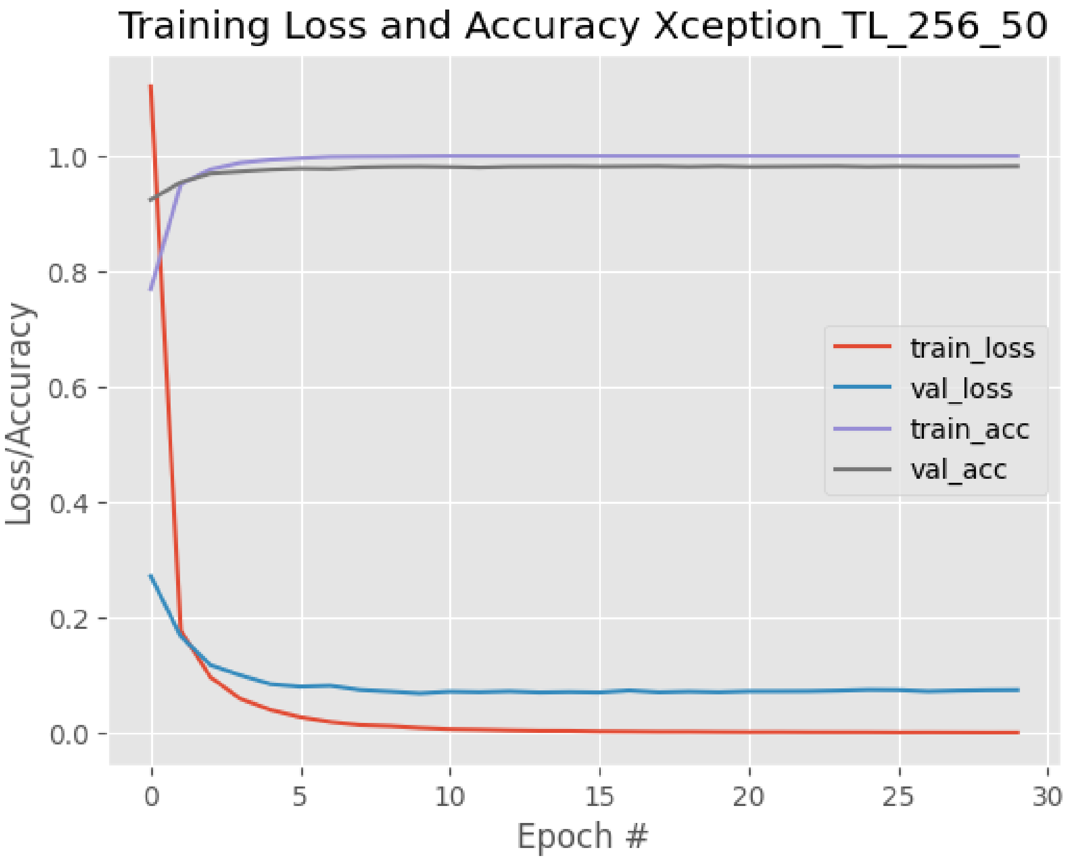
```
Model: "sequential_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 xception (Functional)       (None, 3, 3, 2048)        20861480

 flatten_2 (Flatten)         (None, 18432)             0

 dense_4 (Dense)             (None, 128)               2359424

 dense_5 (Dense)             (None, 15)                1935

=================================================================
Total params: 23222839 (88.59 MB)
Trainable params: 2361359 (9.01 MB)
Non-trainable params: 20861480 (79.58 MB)
_________________________________________________________________
[INFO]: Entrenando Top Model sobre  Xception  …
```

```
Epoch 1/50
58/58 [==============================] - 17s 204ms/step - loss: 1.1204 -
accuracy: 0.7695 - val_loss: 0.2719 - val_accuracy: 0.9243
Epoch 2/50
58/58 [==============================] - 11s 171ms/step - loss: 0.1775 -
accuracy: 0.9510 - val_loss: 0.1677 - val_accuracy: 0.9543
Epoch 3/50
58/58 [==============================] - 10s 169ms/step - loss: 0.0965 -
accuracy: 0.9770 - val_loss: 0.1178 - val_accuracy: 0.9697
Epoch 4/50
58/58 [==============================] - 9s 157ms/step - loss: 0.0593 -
accuracy: 0.9883 - val_loss: 0.1001 - val_accuracy: 0.9730
Epoch 5/50
58/58 [==============================] - 9s 155ms/step - loss: 0.0402 -
accuracy: 0.9934 - val_loss: 0.0848 - val_accuracy: 0.9763
Epoch 6/50
58/58 [==============================] - 10s 175ms/step - loss: 0.0276 -
accuracy: 0.9962 - val_loss: 0.0803 - val_accuracy: 0.9783
Epoch 7/50
58/58 [==============================] - 9s 152ms/step - loss: 0.0193 -
accuracy: 0.9986 - val_loss: 0.0820 - val_accuracy: 0.9777
Epoch 8/50
58/58 [==============================] - 10s 178ms/step - loss: 0.0140 -
accuracy: 0.9991 - val_loss: 0.0748 - val_accuracy: 0.9803
Epoch 9/50
58/58 [==============================] - 10s 176ms/step - loss: 0.0121 -
accuracy: 0.9993 - val_loss: 0.0720 - val_accuracy: 0.9813
Epoch 10/50
58/58 [==============================] - 9s 160ms/step - loss: 0.0089 -
accuracy: 0.9997 - val_loss: 0.0689 - val_accuracy: 0.9817
Epoch 11/50
58/58 [==============================] - 10s 181ms/step - loss: 0.0066 -
accuracy: 1.0000 - val_loss: 0.0719 - val_accuracy: 0.9810
Epoch 12/50
58/58 [==============================] - 10s 174ms/step - loss: 0.0059 -
accuracy: 1.0000 - val_loss: 0.0709 - val_accuracy: 0.9800
Epoch 13/50
58/58 [==============================] - 9s 155ms/step - loss: 0.0048 -
accuracy: 1.0000 - val_loss: 0.0723 - val_accuracy: 0.9813
Epoch 14/50
58/58 [==============================] - 10s 177ms/step - loss: 0.0040 -
accuracy: 1.0000 - val_loss: 0.0704 - val_accuracy: 0.9817
Epoch 15/50
58/58 [==============================] - 9s 154ms/step - loss: 0.0039 -
accuracy: 1.0000 - val_loss: 0.0712 - val_accuracy: 0.9820
Epoch 16/50
58/58 [==============================] - 10s 177ms/step - loss: 0.0030 -
accuracy: 1.0000 - val_loss: 0.0704 - val_accuracy: 0.9817
```

```
Epoch 17/50
58/58 [==============================] - 10s 173ms/step - loss: 0.0027 -
accuracy: 1.0000 - val_loss: 0.0738 - val_accuracy: 0.9820
Epoch 18/50
58/58 [==============================] - 10s 174ms/step - loss: 0.0023 -
accuracy: 1.0000 - val_loss: 0.0707 - val_accuracy: 0.9823
Epoch 19/50
58/58 [==============================] - 10s 177ms/step - loss: 0.0023 -
accuracy: 1.0000 - val_loss: 0.0719 - val_accuracy: 0.9813
Epoch 20/50
58/58 [==============================] - 10s 173ms/step - loss: 0.0019 -
accuracy: 1.0000 - val_loss: 0.0708 - val_accuracy: 0.9823
Epoch 21/50
58/58 [==============================] - 10s 176ms/step - loss: 0.0017 -
accuracy: 1.0000 - val_loss: 0.0723 - val_accuracy: 0.9813
Epoch 22/50
58/58 [==============================] - 10s 175ms/step - loss: 0.0017 -
accuracy: 1.0000 - val_loss: 0.0724 - val_accuracy: 0.9817
Epoch 23/50
58/58 [==============================] - 9s 155ms/step - loss: 0.0015 -
accuracy: 1.0000 - val_loss: 0.0725 - val_accuracy: 0.9820
Epoch 24/50
58/58 [==============================] - 10s 178ms/step - loss: 0.0014 -
accuracy: 1.0000 - val_loss: 0.0736 - val_accuracy: 0.9823
Epoch 25/50
58/58 [==============================] - 10s 173ms/step - loss: 0.0013 -
accuracy: 1.0000 - val_loss: 0.0749 - val_accuracy: 0.9813
Epoch 26/50
58/58 [==============================] - 9s 155ms/step - loss: 0.0010 -
accuracy: 1.0000 - val_loss: 0.0746 - val_accuracy: 0.9820
Epoch 27/50
58/58 [==============================] - 9s 160ms/step - loss: 0.0011 -
accuracy: 1.0000 - val_loss: 0.0723 - val_accuracy: 0.9817
Epoch 28/50
58/58 [==============================] - 10s 175ms/step - loss: 9.5904e-04 -
accuracy: 1.0000 - val_loss: 0.0736 - val_accuracy: 0.9817
Epoch 29/50
58/58 [==============================] - 10s 177ms/step - loss: 8.2753e-04 -
accuracy: 1.0000 - val_loss: 0.0743 - val_accuracy: 0.9820
Epoch 30/50
58/58 [==============================] - 9s 155ms/step - loss: 8.6573e-04 -
accuracy: 1.0000 - val_loss: 0.0746 - val_accuracy: 0.9823
Saved model to disk
```

[ ]:

```
visualize_learning_curve(H, lb =  pretrain_exp)
```



```
[ ]: # Evaluando modelo pre entrenado

     evaluate_model(pre_trained_model, x_test_preprocessed, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 4s 90ms/step
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       200
           1       0.98      0.98      0.98       200
           2       1.00      0.97      0.98       200
           3       0.99      0.97      0.98       200
           4       1.00      1.00      1.00       200
           5       0.97      0.96      0.97       200
           6       0.99      0.97      0.98       200
           7       0.96      0.99      0.97       200
           8       0.98      0.98      0.98       200
           9       0.95      0.98      0.97       200
          10       0.99      0.99      0.99       200
          11       0.99      0.99      0.99       200
          12       0.99      0.97      0.98       200
          13       0.98      0.97      0.98       200
          14       0.97      0.98      0.98       200

    accuracy                           0.98      3000
   macro avg       0.98      0.98      0.98      3000
weighted avg       0.98      0.98      0.98      3000
```

###**6.1.3. InceptionV3**

[ ]:

```python
# Obtención modelo pre entrenado InceptionV3

base = 'InceptionV3'
base_model, x_train_preprocessed, x_val_preprocessed, x_test_preprocessed = get_base_model(base, x_train, x_val, x_test)

pretrain_exp = base + "_TL_" + str(batch_size) + "_" + str(epochs)
exp_set.add(pretrain_exp)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicatio
ns/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 3s 0us/step
Model: "inception_v3"
________________________________________________________________________________
__________________
 Layer (type)                Output Shape                 Param #   Connected to
================================================================================
==================
 input_5 (InputLayer)        [(None, 75, 75, 3)]          0         []

 conv2d_12 (Conv2D)          (None, 37, 37, 32)           864
['input_5[0][0]']

 batch_normalization_12 (Ba  (None, 37, 37, 32)           96
['conv2d_12[0][0]']
 tchNormalization)

 activation (Activation)     (None, 37, 37, 32)           0
['batch_normalization_12[0][0]
                                                                      ']

 conv2d_13 (Conv2D)          (None, 35, 35, 32)           9216
['activation[0][0]']

 batch_normalization_13 (Ba  (None, 35, 35, 32)           96
['conv2d_13[0][0]']
 tchNormalization)
 activation_1 (Activation)   (None, 35, 35, 32)           0
['batch_normalization_13[0][0]
                                                                     ']

 conv2d_14 (Conv2D)          (None, 35, 35, 64)           18432
['activation_1[0][0]']

 batch_normalization_14 (Ba  (None, 35, 35, 64)           192
['conv2d_14[0][0]']
 tchNormalization)

 activation_2 (Activation)   (None, 35, 35, 64)           0
['batch_normalization_14[0][0]
                                                                     ']

 max_pooling2d (MaxPooling2  (None, 17, 17, 64)           0
['activation_2[0][0]']
 D)

 conv2d_15 (Conv2D)          (None, 17, 17, 80)           5120
['max_pooling2d[0][0]']

 batch_normalization_15 (Ba  (None, 17, 17, 80)           240
['conv2d_15[0][0]']
 tchNormalization)

 activation_3 (Activation)   (None, 17, 17, 80)           0
['batch_normalization_15[0][0]
                                                                     ']

 conv2d_16 (Conv2D)          (None, 15, 15, 192)          138240
['activation_3[0][0]']

 batch_normalization_16 (Ba  (None, 15, 15, 192)          576
['conv2d_16[0][0]']
 tchNormalization)

 activation_4 (Activation)   (None, 15, 15, 192)          0
['batch_normalization_16[0][0]
                                                                     ']

 max_pooling2d_1 (MaxPoolin  (None, 7, 7, 192)            0
['activation_4[0][0]']
 g2D)

 conv2d_20 (Conv2D)          (None, 7, 7, 64)             12288
['max_pooling2d_1[0][0]']
```

```
 batch_normalization_20 (Ba  (None, 7, 7, 64)             192
['conv2d_20[0][0]']
 tchNormalization)

 activation_8 (Activation)   (None, 7, 7, 64)             0
['batch_normalization_20[0][0]
                                                                    ']

 conv2d_18 (Conv2D)          (None, 7, 7, 48)             9216
['max_pooling2d_1[0][0]']

 conv2d_21 (Conv2D)          (None, 7, 7, 96)             55296
['activation_8[0][0]']

 batch_normalization_18 (Ba  (None, 7, 7, 48)             144
['conv2d_18[0][0]']
 tchNormalization)

 batch_normalization_21 (Ba  (None, 7, 7, 96)             288
['conv2d_21[0][0]']
 tchNormalization)

 activation_6 (Activation)   (None, 7, 7, 48)             0
['batch_normalization_18[0][0]
                                                                    ']

 activation_9 (Activation)   (None, 7, 7, 96)             0
['batch_normalization_21[0][0]
                                                                    ']

 average_pooling2d (Average  (None, 7, 7, 192)            0
['max_pooling2d_1[0][0]']
 Pooling2D)

 conv2d_17 (Conv2D)          (None, 7, 7, 64)             12288
['max_pooling2d_1[0][0]']

 conv2d_19 (Conv2D)          (None, 7, 7, 64)             76800
['activation_6[0][0]']

 conv2d_22 (Conv2D)          (None, 7, 7, 96)             82944
['activation_9[0][0]']

 conv2d_23 (Conv2D)          (None, 7, 7, 32)             6144
['average_pooling2d[0][0]']

 batch_normalization_17 (Ba  (None, 7, 7, 64)             192
['conv2d_17[0][0]']
 tchNormalization)

 batch_normalization_19 (Ba  (None, 7, 7, 64)             192
['conv2d_19[0][0]']
 tchNormalization)

 batch_normalization_22 (Ba  (None, 7, 7, 96)             288
['conv2d_22[0][0]']
 tchNormalization)

 batch_normalization_23 (Ba  (None, 7, 7, 32)             96
['conv2d_23[0][0]']
 tchNormalization)

 activation_5 (Activation)   (None, 7, 7, 64)             0
['batch_normalization_17[0][0]
                                                                     ']

 activation_7 (Activation)   (None, 7, 7, 64)             0
['batch_normalization_19[0][0]
                                                                     ']

 activation_10 (Activation)  (None, 7, 7, 96)             0
['batch_normalization_22[0][0]
                                                                     ']

 activation_11 (Activation)  (None, 7, 7, 32)             0
['batch_normalization_23[0][0]
                                                                     ']

 mixed0 (Concatenate)        (None, 7, 7, 256)            0
['activation_5[0][0]',
'activation_7[0][0]',
'activation_10[0][0]',
'activation_11[0][0]']

 conv2d_27 (Conv2D)          (None, 7, 7, 64)             16384
['mixed0[0][0]']

 batch_normalization_27 (Ba  (None, 7, 7, 64)             192
['conv2d_27[0][0]']
 tchNormalization)

 activation_15 (Activation)  (None, 7, 7, 64)             0
['batch_normalization_27[0][0]
                                                                     ']
 conv2d_25 (Conv2D)          (None, 7, 7, 48)             12288
['mixed0[0][0]']

 conv2d_28 (Conv2D)          (None, 7, 7, 96)             55296
['activation_15[0][0]']

 batch_normalization_25 (Ba  (None, 7, 7, 48)             144
['conv2d_25[0][0]']
 tchNormalization)

 batch_normalization_28 (Ba  (None, 7, 7, 96)             288
['conv2d_28[0][0]']
 tchNormalization)

 activation_13 (Activation)  (None, 7, 7, 48)             0
['batch_normalization_25[0][0]
                                                                    ']

 activation_16 (Activation)  (None, 7, 7, 96)             0
['batch_normalization_28[0][0]
                                                                    ']

 average_pooling2d_1 (Avera  (None, 7, 7, 256)            0
['mixed0[0][0]']
 gePooling2D)

 conv2d_24 (Conv2D)          (None, 7, 7, 64)             16384
['mixed0[0][0]']

 conv2d_26 (Conv2D)          (None, 7, 7, 64)             76800
['activation_13[0][0]']

 conv2d_29 (Conv2D)          (None, 7, 7, 96)             82944
['activation_16[0][0]']

 conv2d_30 (Conv2D)          (None, 7, 7, 64)             16384
['average_pooling2d_1[0][0]']

 batch_normalization_24 (Ba  (None, 7, 7, 64)             192
['conv2d_24[0][0]']
 tchNormalization)

 batch_normalization_26 (Ba  (None, 7, 7, 64)             192
['conv2d_26[0][0]']
 tchNormalization)

 batch_normalization_29 (Ba  (None, 7, 7, 96)             288
['conv2d_29[0][0]']
```

```
 tchNormalization)

 batch_normalization_30 (Ba  (None, 7, 7, 64)             192
['conv2d_30[0][0]']
 tchNormalization)

 activation_12 (Activation)  (None, 7, 7, 64)             0
['batch_normalization_24[0][0]
                                                                     ']

 activation_14 (Activation)  (None, 7, 7, 64)             0
['batch_normalization_26[0][0]
                                                                     ']

 activation_17 (Activation)  (None, 7, 7, 96)             0
['batch_normalization_29[0][0]
                                                                     ']

 activation_18 (Activation)  (None, 7, 7, 64)             0
['batch_normalization_30[0][0]
                                                                     ']

 mixed1 (Concatenate)        (None, 7, 7, 288)            0
['activation_12[0][0]',
'activation_14[0][0]',
'activation_17[0][0]',
'activation_18[0][0]']

 conv2d_34 (Conv2D)          (None, 7, 7, 64)             18432
['mixed1[0][0]']

 batch_normalization_34 (Ba  (None, 7, 7, 64)             192
['conv2d_34[0][0]']
 tchNormalization)

 activation_22 (Activation)  (None, 7, 7, 64)             0
['batch_normalization_34[0][0]
                                                                     ']

 conv2d_32 (Conv2D)          (None, 7, 7, 48)             13824
['mixed1[0][0]']

 conv2d_35 (Conv2D)          (None, 7, 7, 96)             55296
['activation_22[0][0]']

 batch_normalization_32 (Ba  (None, 7, 7, 48)             144
['conv2d_32[0][0]']
 tchNormalization)
 batch_normalization_35 (Ba  (None, 7, 7, 96)            288
['conv2d_35[0][0]']
 tchNormalization)

 activation_20 (Activation)  (None, 7, 7, 48)            0
['batch_normalization_32[0][0]
                                                                  ']

 activation_23 (Activation)  (None, 7, 7, 96)            0
['batch_normalization_35[0][0]
                                                                  ']

 average_pooling2d_2 (Avera  (None, 7, 7, 288)           0
['mixed1[0][0]']
 gePooling2D)

 conv2d_31 (Conv2D)          (None, 7, 7, 64)            18432
['mixed1[0][0]']

 conv2d_33 (Conv2D)          (None, 7, 7, 64)            76800
['activation_20[0][0]']

 conv2d_36 (Conv2D)          (None, 7, 7, 96)            82944
['activation_23[0][0]']

 conv2d_37 (Conv2D)          (None, 7, 7, 64)            18432
['average_pooling2d_2[0][0]']

 batch_normalization_31 (Ba  (None, 7, 7, 64)            192
['conv2d_31[0][0]']
 tchNormalization)

 batch_normalization_33 (Ba  (None, 7, 7, 64)            192
['conv2d_33[0][0]']
 tchNormalization)

 batch_normalization_36 (Ba  (None, 7, 7, 96)            288
['conv2d_36[0][0]']
 tchNormalization)

 batch_normalization_37 (Ba  (None, 7, 7, 64)            192
['conv2d_37[0][0]']
 tchNormalization)

 activation_19 (Activation)  (None, 7, 7, 64)            0
['batch_normalization_31[0][0]
                                                                  ']
 activation_21 (Activation)  (None, 7, 7, 64)              0
['batch_normalization_33[0][0]
                                                                     ']

 activation_24 (Activation)  (None, 7, 7, 96)              0
['batch_normalization_36[0][0]
                                                                     ']

 activation_25 (Activation)  (None, 7, 7, 64)              0
['batch_normalization_37[0][0]
                                                                     ']

 mixed2 (Concatenate)        (None, 7, 7, 288)             0
['activation_19[0][0]',
'activation_21[0][0]',
'activation_24[0][0]',
'activation_25[0][0]']

 conv2d_39 (Conv2D)          (None, 7, 7, 64)              18432
['mixed2[0][0]']

 batch_normalization_39 (Ba  (None, 7, 7, 64)              192
['conv2d_39[0][0]']
 tchNormalization)

 activation_27 (Activation)  (None, 7, 7, 64)              0
['batch_normalization_39[0][0]
                                                                     ']

 conv2d_40 (Conv2D)          (None, 7, 7, 96)              55296
['activation_27[0][0]']

 batch_normalization_40 (Ba  (None, 7, 7, 96)              288
['conv2d_40[0][0]']
 tchNormalization)

 activation_28 (Activation)  (None, 7, 7, 96)              0
['batch_normalization_40[0][0]
                                                                     ']

 conv2d_38 (Conv2D)          (None, 3, 3, 384)             995328
['mixed2[0][0]']

 conv2d_41 (Conv2D)          (None, 3, 3, 96)              82944
['activation_28[0][0]']

 batch_normalization_38 (Ba  (None, 3, 3, 384)             1152
['conv2d_38[0][0]']
 tchNormalization)

 batch_normalization_41 (Ba  (None, 3, 3, 96)             288
['conv2d_41[0][0]']
 tchNormalization)

 activation_26 (Activation)  (None, 3, 3, 384)            0
['batch_normalization_38[0][0]
                                                                    ']

 activation_29 (Activation)  (None, 3, 3, 96)             0
['batch_normalization_41[0][0]
                                                                    ']

 max_pooling2d_2 (MaxPoolin  (None, 3, 3, 288)            0
['mixed2[0][0]']
 g2D)

 mixed3 (Concatenate)        (None, 3, 3, 768)            0
['activation_26[0][0]',
'activation_29[0][0]',
'max_pooling2d_2[0][0]']

 conv2d_46 (Conv2D)          (None, 3, 3, 128)            98304
['mixed3[0][0]']

 batch_normalization_46 (Ba  (None, 3, 3, 128)            384
['conv2d_46[0][0]']
 tchNormalization)

 activation_34 (Activation)  (None, 3, 3, 128)            0
['batch_normalization_46[0][0]
                                                                    ']

 conv2d_47 (Conv2D)          (None, 3, 3, 128)            114688
['activation_34[0][0]']

 batch_normalization_47 (Ba  (None, 3, 3, 128)            384
['conv2d_47[0][0]']
 tchNormalization)

 activation_35 (Activation)  (None, 3, 3, 128)            0
['batch_normalization_47[0][0]
                                                                    ']

 conv2d_43 (Conv2D)          (None, 3, 3, 128)            98304
['mixed3[0][0]']
```

```
 conv2d_48 (Conv2D)          (None, 3, 3, 128)            114688
['activation_35[0][0]']

 batch_normalization_43 (Ba  (None, 3, 3, 128)            384
['conv2d_43[0][0]']
 tchNormalization)

 batch_normalization_48 (Ba  (None, 3, 3, 128)            384
['conv2d_48[0][0]']
 tchNormalization)

 activation_31 (Activation)  (None, 3, 3, 128)            0
['batch_normalization_43[0][0]
                                                                    ']

 activation_36 (Activation)  (None, 3, 3, 128)            0
['batch_normalization_48[0][0]
                                                                    ']

 conv2d_44 (Conv2D)          (None, 3, 3, 128)            114688
['activation_31[0][0]']

 conv2d_49 (Conv2D)          (None, 3, 3, 128)            114688
['activation_36[0][0]']

 batch_normalization_44 (Ba  (None, 3, 3, 128)            384
['conv2d_44[0][0]']
 tchNormalization)

 batch_normalization_49 (Ba  (None, 3, 3, 128)            384
['conv2d_49[0][0]']
 tchNormalization)

 activation_32 (Activation)  (None, 3, 3, 128)            0
['batch_normalization_44[0][0]
                                                                    ']

 activation_37 (Activation)  (None, 3, 3, 128)            0
['batch_normalization_49[0][0]
                                                                    ']

 average_pooling2d_3 (Avera  (None, 3, 3, 768)            0
['mixed3[0][0]']
 gePooling2D)

 conv2d_42 (Conv2D)          (None, 3, 3, 192)            147456
['mixed3[0][0]']
```

```
 conv2d_45 (Conv2D)          (None, 3, 3, 192)            172032
['activation_32[0][0]']

 conv2d_50 (Conv2D)          (None, 3, 3, 192)            172032
['activation_37[0][0]']

 conv2d_51 (Conv2D)          (None, 3, 3, 192)            147456
['average_pooling2d_3[0][0]']

 batch_normalization_42 (Ba  (None, 3, 3, 192)            576
['conv2d_42[0][0]']
 tchNormalization)

 batch_normalization_45 (Ba  (None, 3, 3, 192)            576
['conv2d_45[0][0]']
 tchNormalization)

 batch_normalization_50 (Ba  (None, 3, 3, 192)            576
['conv2d_50[0][0]']
 tchNormalization)

 batch_normalization_51 (Ba  (None, 3, 3, 192)            576
['conv2d_51[0][0]']
 tchNormalization)

 activation_30 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_42[0][0]
                                                                    ']

 activation_33 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_45[0][0]
                                                                    ']

 activation_38 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_50[0][0]
                                                                    ']

 activation_39 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_51[0][0]
                                                                    ']

 mixed4 (Concatenate)        (None, 3, 3, 768)            0
['activation_30[0][0]',
'activation_33[0][0]',
'activation_38[0][0]',
'activation_39[0][0]']
```

```
 conv2d_56 (Conv2D)          (None, 3, 3, 160)             122880
['mixed4[0][0]']

 batch_normalization_56 (Ba  (None, 3, 3, 160)             480
['conv2d_56[0][0]']
 tchNormalization)

 activation_44 (Activation)  (None, 3, 3, 160)             0
['batch_normalization_56[0][0]
                                                                     ']

 conv2d_57 (Conv2D)          (None, 3, 3, 160)             179200
['activation_44[0][0]']

 batch_normalization_57 (Ba  (None, 3, 3, 160)             480
['conv2d_57[0][0]']
 tchNormalization)

 activation_45 (Activation)  (None, 3, 3, 160)             0
['batch_normalization_57[0][0]
                                                                     ']

 conv2d_53 (Conv2D)          (None, 3, 3, 160)             122880
['mixed4[0][0]']

 conv2d_58 (Conv2D)          (None, 3, 3, 160)             179200
['activation_45[0][0]']

 batch_normalization_53 (Ba  (None, 3, 3, 160)             480
['conv2d_53[0][0]']
 tchNormalization)

 batch_normalization_58 (Ba  (None, 3, 3, 160)             480
['conv2d_58[0][0]']
 tchNormalization)

 activation_41 (Activation)  (None, 3, 3, 160)             0
['batch_normalization_53[0][0]
                                                                     ']

 activation_46 (Activation)  (None, 3, 3, 160)             0
['batch_normalization_58[0][0]
                                                                     ']

 conv2d_54 (Conv2D)          (None, 3, 3, 160)             179200
['activation_41[0][0]']

 conv2d_59 (Conv2D)          (None, 3, 3, 160)             179200
```

```
['activation_46[0][0]']

 batch_normalization_54 (Ba  (None, 3, 3, 160)            480
['conv2d_54[0][0]']
 tchNormalization)

 batch_normalization_59 (Ba  (None, 3, 3, 160)            480
['conv2d_59[0][0]']
 tchNormalization)

 activation_42 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_54[0][0]
                                                                      ']

 activation_47 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_59[0][0]
                                                                      ']

 average_pooling2d_4 (Avera  (None, 3, 3, 768)            0
['mixed4[0][0]']
 gePooling2D)

 conv2d_52 (Conv2D)          (None, 3, 3, 192)            147456
['mixed4[0][0]']

 conv2d_55 (Conv2D)          (None, 3, 3, 192)            215040
['activation_42[0][0]']

 conv2d_60 (Conv2D)          (None, 3, 3, 192)            215040
['activation_47[0][0]']

 conv2d_61 (Conv2D)          (None, 3, 3, 192)            147456
['average_pooling2d_4[0][0]']

 batch_normalization_52 (Ba  (None, 3, 3, 192)            576
['conv2d_52[0][0]']
 tchNormalization)

 batch_normalization_55 (Ba  (None, 3, 3, 192)            576
['conv2d_55[0][0]']
 tchNormalization)

 batch_normalization_60 (Ba  (None, 3, 3, 192)            576
['conv2d_60[0][0]']
 tchNormalization)

 batch_normalization_61 (Ba  (None, 3, 3, 192)            576
['conv2d_61[0][0]']
```

```
 tchNormalization)

 activation_40 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_52[0][0]
                                                                     ']

 activation_43 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_55[0][0]
                                                                     ']

 activation_48 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_60[0][0]
                                                                     ']

 activation_49 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_61[0][0]
                                                                     ']

 mixed5 (Concatenate)        (None, 3, 3, 768)            0
['activation_40[0][0]',
'activation_43[0][0]',
'activation_48[0][0]',
'activation_49[0][0]']

 conv2d_66 (Conv2D)          (None, 3, 3, 160)            122880
['mixed5[0][0]']

 batch_normalization_66 (Ba  (None, 3, 3, 160)            480
['conv2d_66[0][0]']
 tchNormalization)

 activation_54 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_66[0][0]
                                                                     ']

 conv2d_67 (Conv2D)          (None, 3, 3, 160)            179200
['activation_54[0][0]']

 batch_normalization_67 (Ba  (None, 3, 3, 160)            480
['conv2d_67[0][0]']
 tchNormalization)

 activation_55 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_67[0][0]
                                                                     ']

 conv2d_63 (Conv2D)          (None, 3, 3, 160)            122880
['mixed5[0][0]']
```

```
 conv2d_68 (Conv2D)          (None, 3, 3, 160)            179200
['activation_55[0][0]']

 batch_normalization_63 (Ba  (None, 3, 3, 160)            480
['conv2d_63[0][0]']
 tchNormalization)

 batch_normalization_68 (Ba  (None, 3, 3, 160)            480
['conv2d_68[0][0]']
 tchNormalization)

 activation_51 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_63[0][0]
                                                                     ']

 activation_56 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_68[0][0]
                                                                     ']

 conv2d_64 (Conv2D)          (None, 3, 3, 160)            179200
['activation_51[0][0]']

 conv2d_69 (Conv2D)          (None, 3, 3, 160)            179200
['activation_56[0][0]']

 batch_normalization_64 (Ba  (None, 3, 3, 160)            480
['conv2d_64[0][0]']
 tchNormalization)

 batch_normalization_69 (Ba  (None, 3, 3, 160)            480
['conv2d_69[0][0]']
 tchNormalization)

 activation_52 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_64[0][0]
                                                                     ']

 activation_57 (Activation)  (None, 3, 3, 160)            0
['batch_normalization_69[0][0]
                                                                     ']

 average_pooling2d_5 (Avera  (None, 3, 3, 768)            0
['mixed5[0][0]']
 gePooling2D)

 conv2d_62 (Conv2D)          (None, 3, 3, 192)            147456
['mixed5[0][0]']
```

```
 conv2d_65 (Conv2D)          (None, 3, 3, 192)           215040
['activation_52[0][0]']

 conv2d_70 (Conv2D)          (None, 3, 3, 192)           215040
['activation_57[0][0]']

 conv2d_71 (Conv2D)          (None, 3, 3, 192)           147456
['average_pooling2d_5[0][0]']

 batch_normalization_62 (Ba  (None, 3, 3, 192)           576
['conv2d_62[0][0]']
 tchNormalization)

 batch_normalization_65 (Ba  (None, 3, 3, 192)           576
['conv2d_65[0][0]']
 tchNormalization)

 batch_normalization_70 (Ba  (None, 3, 3, 192)           576
['conv2d_70[0][0]']
 tchNormalization)

 batch_normalization_71 (Ba  (None, 3, 3, 192)           576
['conv2d_71[0][0]']
 tchNormalization)

 activation_50 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_62[0][0]
                                                                    ']

 activation_53 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_65[0][0]
                                                                    ']

 activation_58 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_70[0][0]
                                                                    ']

 activation_59 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_71[0][0]
                                                                    ']

 mixed6 (Concatenate)        (None, 3, 3, 768)           0
['activation_50[0][0]',
'activation_53[0][0]',
'activation_58[0][0]',
'activation_59[0][0]']
```

```
 conv2d_76 (Conv2D)           (None, 3, 3, 192)           147456
['mixed6[0][0]']

 batch_normalization_76 (Ba  (None, 3, 3, 192)           576
['conv2d_76[0][0]']
 tchNormalization)

 activation_64 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_76[0][0]
                                                                     ']

 conv2d_77 (Conv2D)           (None, 3, 3, 192)           258048
['activation_64[0][0]']

 batch_normalization_77 (Ba  (None, 3, 3, 192)           576
['conv2d_77[0][0]']
 tchNormalization)

 activation_65 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_77[0][0]
                                                                     ']

 conv2d_73 (Conv2D)           (None, 3, 3, 192)           147456
['mixed6[0][0]']

 conv2d_78 (Conv2D)           (None, 3, 3, 192)           258048
['activation_65[0][0]']

 batch_normalization_73 (Ba  (None, 3, 3, 192)           576
['conv2d_73[0][0]']
 tchNormalization)

 batch_normalization_78 (Ba  (None, 3, 3, 192)           576
['conv2d_78[0][0]']
 tchNormalization)

 activation_61 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_73[0][0]
                                                                     ']

 activation_66 (Activation)  (None, 3, 3, 192)           0
['batch_normalization_78[0][0]
                                                                     ']

 conv2d_74 (Conv2D)           (None, 3, 3, 192)           258048
['activation_61[0][0]']

 conv2d_79 (Conv2D)           (None, 3, 3, 192)           258048
```

```
['activation_66[0][0]']

 batch_normalization_74 (Ba  (None, 3, 3, 192)            576
['conv2d_74[0][0]']
 tchNormalization)

 batch_normalization_79 (Ba  (None, 3, 3, 192)            576
['conv2d_79[0][0]']
 tchNormalization)

 activation_62 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_74[0][0]
                                                                    ']

 activation_67 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_79[0][0]
                                                                    ']

 average_pooling2d_6 (Avera  (None, 3, 3, 768)            0
['mixed6[0][0]']
 gePooling2D)

 conv2d_72 (Conv2D)          (None, 3, 3, 192)            147456
['mixed6[0][0]']

 conv2d_75 (Conv2D)          (None, 3, 3, 192)            258048
['activation_62[0][0]']

 conv2d_80 (Conv2D)          (None, 3, 3, 192)            258048
['activation_67[0][0]']

 conv2d_81 (Conv2D)          (None, 3, 3, 192)            147456
['average_pooling2d_6[0][0]']

 batch_normalization_72 (Ba  (None, 3, 3, 192)            576
['conv2d_72[0][0]']
 tchNormalization)

 batch_normalization_75 (Ba  (None, 3, 3, 192)            576
['conv2d_75[0][0]']
 tchNormalization)

 batch_normalization_80 (Ba  (None, 3, 3, 192)            576
['conv2d_80[0][0]']
 tchNormalization)

 batch_normalization_81 (Ba  (None, 3, 3, 192)            576
['conv2d_81[0][0]']
```

```
 tchNormalization)

 activation_60 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_72[0][0]
                                                                    ']

 activation_63 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_75[0][0]
                                                                    ']

 activation_68 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_80[0][0]
                                                                    ']

 activation_69 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_81[0][0]
                                                                    ']

 mixed7 (Concatenate)        (None, 3, 3, 768)            0
['activation_60[0][0]',
'activation_63[0][0]',
'activation_68[0][0]',
'activation_69[0][0]']

 conv2d_84 (Conv2D)          (None, 3, 3, 192)            147456
['mixed7[0][0]']

 batch_normalization_84 (Ba  (None, 3, 3, 192)            576
['conv2d_84[0][0]']
 tchNormalization)

 activation_72 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_84[0][0]
                                                                    ']

 conv2d_85 (Conv2D)          (None, 3, 3, 192)            258048
['activation_72[0][0]']

 batch_normalization_85 (Ba  (None, 3, 3, 192)            576
['conv2d_85[0][0]']
 tchNormalization)

 activation_73 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_85[0][0]
                                                                    ']

 conv2d_82 (Conv2D)          (None, 3, 3, 192)            147456
['mixed7[0][0]']
```

```
 conv2d_86 (Conv2D)          (None, 3, 3, 192)            258048
['activation_73[0][0]']

 batch_normalization_82 (Ba  (None, 3, 3, 192)            576
['conv2d_82[0][0]']
 tchNormalization)

 batch_normalization_86 (Ba  (None, 3, 3, 192)            576
['conv2d_86[0][0]']
 tchNormalization)

 activation_70 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_82[0][0]
                                                                     ']

 activation_74 (Activation)  (None, 3, 3, 192)            0
['batch_normalization_86[0][0]
                                                                     ']

 conv2d_83 (Conv2D)          (None, 1, 1, 320)            552960
['activation_70[0][0]']

 conv2d_87 (Conv2D)          (None, 1, 1, 192)            331776
['activation_74[0][0]']

 batch_normalization_83 (Ba  (None, 1, 1, 320)            960
['conv2d_83[0][0]']
 tchNormalization)

 batch_normalization_87 (Ba  (None, 1, 1, 192)            576
['conv2d_87[0][0]']
 tchNormalization)

 activation_71 (Activation)  (None, 1, 1, 320)            0
['batch_normalization_83[0][0]
                                                                     ']

 activation_75 (Activation)  (None, 1, 1, 192)            0
['batch_normalization_87[0][0]
                                                                     ']

 max_pooling2d_3 (MaxPoolin  (None, 1, 1, 768)            0
['mixed7[0][0]']
 g2D)

 mixed8 (Concatenate)        (None, 1, 1, 1280)           0
['activation_71[0][0]',
'activation_75[0][0]',
'max_pooling2d_3[0][0]']

 conv2d_92 (Conv2D)          (None, 1, 1, 448)            573440
['mixed8[0][0]']

 batch_normalization_92 (Ba  (None, 1, 1, 448)            1344
['conv2d_92[0][0]']
 tchNormalization)

 activation_80 (Activation)  (None, 1, 1, 448)            0
['batch_normalization_92[0][0]
                                                                     ']

 conv2d_89 (Conv2D)          (None, 1, 1, 384)            491520
['mixed8[0][0]']

 conv2d_93 (Conv2D)          (None, 1, 1, 384)            1548288
['activation_80[0][0]']

 batch_normalization_89 (Ba  (None, 1, 1, 384)            1152
['conv2d_89[0][0]']
 tchNormalization)

 batch_normalization_93 (Ba  (None, 1, 1, 384)            1152
['conv2d_93[0][0]']
 tchNormalization)

 activation_77 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_89[0][0]
                                                                     ']

 activation_81 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_93[0][0]
                                                                     ']

 conv2d_90 (Conv2D)          (None, 1, 1, 384)            442368
['activation_77[0][0]']

 conv2d_91 (Conv2D)          (None, 1, 1, 384)            442368
['activation_77[0][0]']

 conv2d_94 (Conv2D)          (None, 1, 1, 384)            442368
['activation_81[0][0]']

 conv2d_95 (Conv2D)          (None, 1, 1, 384)            442368
['activation_81[0][0]']
```

```
 average_pooling2d_7 (Avera  (None, 1, 1, 1280)           0
['mixed8[0][0]']
 gePooling2D)

 conv2d_88 (Conv2D)          (None, 1, 1, 320)            409600
['mixed8[0][0]']

 batch_normalization_90 (Ba  (None, 1, 1, 384)            1152
['conv2d_90[0][0]']
 tchNormalization)

 batch_normalization_91 (Ba  (None, 1, 1, 384)            1152
['conv2d_91[0][0]']
 tchNormalization)

 batch_normalization_94 (Ba  (None, 1, 1, 384)            1152
['conv2d_94[0][0]']
 tchNormalization)

 batch_normalization_95 (Ba  (None, 1, 1, 384)            1152
['conv2d_95[0][0]']
 tchNormalization)

 conv2d_96 (Conv2D)          (None, 1, 1, 192)            245760
['average_pooling2d_7[0][0]']

 batch_normalization_88 (Ba  (None, 1, 1, 320)            960
['conv2d_88[0][0]']
 tchNormalization)

 activation_78 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_90[0][0]
                                                                    ']

 activation_79 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_91[0][0]
                                                                    ']

 activation_82 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_94[0][0]
                                                                    ']

 activation_83 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_95[0][0]
                                                                    ']

 batch_normalization_96 (Ba  (None, 1, 1, 192)            576
['conv2d_96[0][0]']
```

```
 tchNormalization)

 activation_76 (Activation)  (None, 1, 1, 320)            0
['batch_normalization_88[0][0]
                                                                    ']

 mixed9_0 (Concatenate)      (None, 1, 1, 768)            0
['activation_78[0][0]',
'activation_79[0][0]']

 concatenate (Concatenate)   (None, 1, 1, 768)            0
['activation_82[0][0]',
'activation_83[0][0]']

 activation_84 (Activation)  (None, 1, 1, 192)            0
['batch_normalization_96[0][0]
                                                                    ']

 mixed9 (Concatenate)        (None, 1, 1, 2048)           0
['activation_76[0][0]',
'mixed9_0[0][0]',
'concatenate[0][0]',
'activation_84[0][0]']

 conv2d_101 (Conv2D)         (None, 1, 1, 448)            917504
['mixed9[0][0]']

 batch_normalization_101 (B  (None, 1, 1, 448)            1344
['conv2d_101[0][0]']
 atchNormalization)

 activation_89 (Activation)  (None, 1, 1, 448)            0
['batch_normalization_101[0][0
                                                                    ]']

 conv2d_98 (Conv2D)          (None, 1, 1, 384)            786432
['mixed9[0][0]']

 conv2d_102 (Conv2D)         (None, 1, 1, 384)            1548288
['activation_89[0][0]']

 batch_normalization_98 (Ba  (None, 1, 1, 384)            1152
['conv2d_98[0][0]']
 tchNormalization)

 batch_normalization_102 (B  (None, 1, 1, 384)            1152
['conv2d_102[0][0]']
 atchNormalization)
 activation_86 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_98[0][0]
                                                                    ']

 activation_90 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_102[0][0
                                                                    ]']

 conv2d_99 (Conv2D)          (None, 1, 1, 384)            442368
['activation_86[0][0]']

 conv2d_100 (Conv2D)         (None, 1, 1, 384)            442368
['activation_86[0][0]']

 conv2d_103 (Conv2D)         (None, 1, 1, 384)            442368
['activation_90[0][0]']

 conv2d_104 (Conv2D)         (None, 1, 1, 384)            442368
['activation_90[0][0]']

 average_pooling2d_8 (Avera  (None, 1, 1, 2048)           0
['mixed9[0][0]']
 gePooling2D)

 conv2d_97 (Conv2D)          (None, 1, 1, 320)            655360
['mixed9[0][0]']

 batch_normalization_99 (Ba  (None, 1, 1, 384)            1152
['conv2d_99[0][0]']
 tchNormalization)

 batch_normalization_100 (B  (None, 1, 1, 384)            1152
['conv2d_100[0][0]']
 atchNormalization)

 batch_normalization_103 (B  (None, 1, 1, 384)            1152
['conv2d_103[0][0]']
 atchNormalization)

 batch_normalization_104 (B  (None, 1, 1, 384)            1152
['conv2d_104[0][0]']
 atchNormalization)

 conv2d_105 (Conv2D)         (None, 1, 1, 192)            393216
['average_pooling2d_8[0][0]']

 batch_normalization_97 (Ba  (None, 1, 1, 320)            960
['conv2d_97[0][0]']
 tchNormalization)

 activation_87 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_99[0][0]
                                                                    ']

 activation_88 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_100[0][0
                                                                    ]']

 activation_91 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_103[0][0
                                                                    ]']

 activation_92 (Activation)  (None, 1, 1, 384)            0
['batch_normalization_104[0][0
                                                                    ]']

 batch_normalization_105 (B  (None, 1, 1, 192)            576
['conv2d_105[0][0]']
 atchNormalization)

 activation_85 (Activation)  (None, 1, 1, 320)            0
['batch_normalization_97[0][0]
                                                                    ']

 mixed9_1 (Concatenate)      (None, 1, 1, 768)            0
['activation_87[0][0]',
'activation_88[0][0]']

 concatenate_1 (Concatenate  (None, 1, 1, 768)            0
['activation_91[0][0]',
 )
'activation_92[0][0]']

 activation_93 (Activation)  (None, 1, 1, 192)            0
['batch_normalization_105[0][0
                                                                    ]']

 mixed10 (Concatenate)       (None, 1, 1, 2048)           0
['activation_85[0][0]',
'mixed9_1[0][0]',
'concatenate_1[0][0]',
'activation_93[0][0]']

================================================================================
==================
Total params: 21802784 (83.17 MB)
Trainable params: 21768352 (83.04 MB)
Non-trainable params: 34432 (134.50 KB)
--------------------------------------------------------------------------------
------------------
```

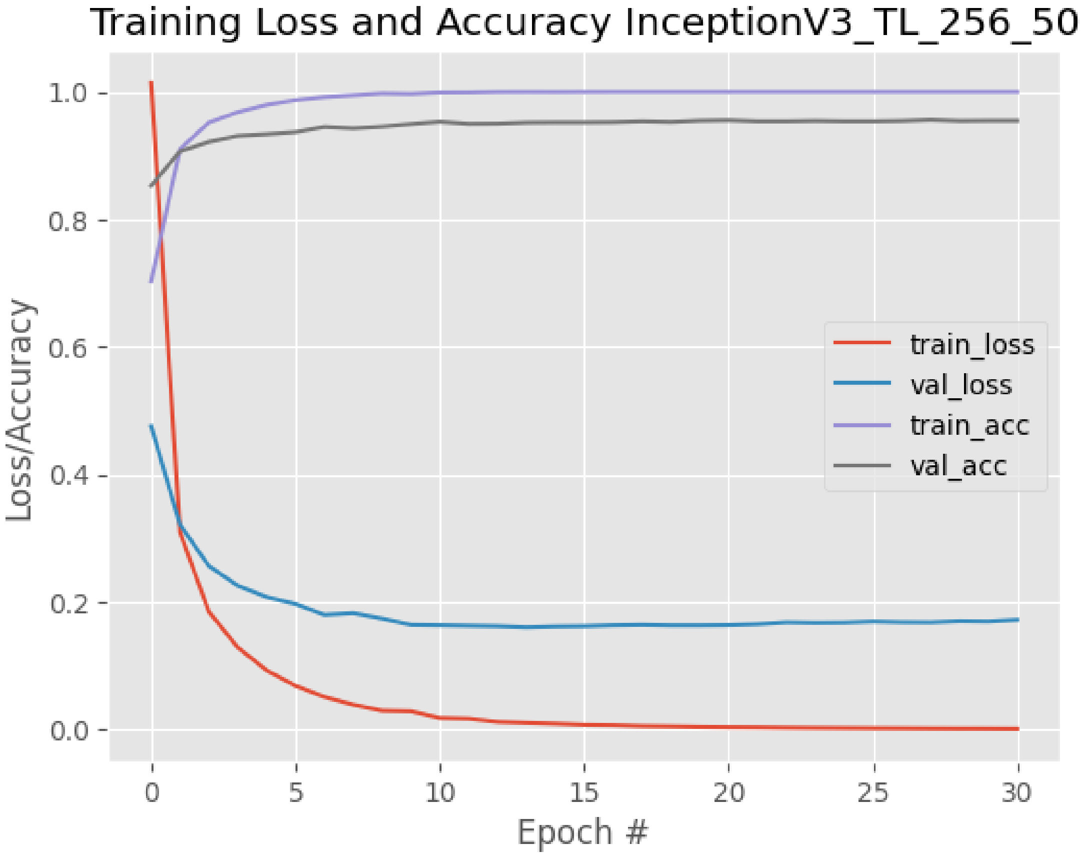
```
[ ]: # Entrenamiento modelo pre entrenado

     if do_training == True:

       pre_trained_model = get_pretrained_model(base_model, dense_size, num_clases)

       pre_trained_model.compile(optimizer=Adam(learning_rate=learning_rate),
                                 loss="categorical_crossentropy",
                                 metrics=["accuracy"])

       print("[INFO]: Entrenando Top Model sobre ", base, " ...")

       H = pre_trained_model.fit(x_train_preprocessed, y_train_ohe,
                                 batch_size=batch_size,
                                 epochs=epochs,
                                 steps_per_epoch=x_train_preprocessed.shape[0] //␣
      ↪batch_size,
                                 validation_data=(x_val_preprocessed, y_val_ohe),
                                 callbacks=[early_stopping_cbck])

       save_trained_model(model = pre_trained_model, history = H, model_name =␣
      ↪pretrain_exp)

     else:
       print("[INFO]: Cargando Top Model sobre " + base + "....")
       pre_trained_model = load_keras_model(pretrain_exp)
       pre_trained_model.summary()
       H = load_history(pretrain_exp)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 1, 1, 2048)        21802784

 flatten_3 (Flatten)         (None, 2048)              0

 dense_6 (Dense)             (None, 128)               262272

 dense_7 (Dense)             (None, 15)                1935

=================================================================
Total params: 22066991 (84.18 MB)
Trainable params: 264207 (1.01 MB)
Non-trainable params: 21802784 (83.17 MB)
________________________________________________________________
[INFO]: Entrenando Top Model sobre  InceptionV3  …
Epoch 1/50
58/58 [==============================] - 15s 132ms/step - loss: 1.0142 -
accuracy: 0.7035 - val_loss: 0.4759 - val_accuracy: 0.8530
Epoch 2/50
58/58 [==============================] - 4s 61ms/step - loss: 0.3093 - accuracy:
0.9105 - val_loss: 0.3214 - val_accuracy: 0.9070
Epoch 3/50
58/58 [==============================] - 4s 65ms/step - loss: 0.1849 - accuracy:
0.9522 - val_loss: 0.2566 - val_accuracy: 0.9217
Epoch 4/50
58/58 [==============================] - 4s 65ms/step - loss: 0.1290 - accuracy:
0.9683 - val_loss: 0.2257 - val_accuracy: 0.9310
Epoch 5/50
58/58 [==============================] - 4s 61ms/step - loss: 0.0931 - accuracy:
0.9797 - val_loss: 0.2081 - val_accuracy: 0.9337
Epoch 6/50
58/58 [==============================] - 4s 63ms/step - loss: 0.0688 - accuracy:
0.9872 - val_loss: 0.1971 - val_accuracy: 0.9370
Epoch 7/50
58/58 [==============================] - 4s 65ms/step - loss: 0.0515 - accuracy:
0.9919 - val_loss: 0.1800 - val_accuracy: 0.9453
Epoch 8/50
58/58 [==============================] - 4s 68ms/step - loss: 0.0391 - accuracy:
0.9946 - val_loss: 0.1827 - val_accuracy: 0.9430
Epoch 9/50
58/58 [==============================] - 3s 60ms/step - loss: 0.0299 - accuracy:
0.9974 - val_loss: 0.1742 - val_accuracy: 0.9457
Epoch 10/50
58/58 [==============================] - 4s 62ms/step - loss: 0.0290 - accuracy:
0.9967 - val_loss: 0.1644 - val_accuracy: 0.9497
Epoch 11/50
58/58 [==============================] - 4s 64ms/step - loss: 0.0185 - accuracy:
0.9990 - val_loss: 0.1638 - val_accuracy: 0.9533
Epoch 12/50
58/58 [==============================] - 4s 70ms/step - loss: 0.0174 - accuracy:
0.9991 - val_loss: 0.1630 - val_accuracy: 0.9500
Epoch 13/50
58/58 [==============================] - 4s 63ms/step - loss: 0.0123 - accuracy:
0.9998 - val_loss: 0.1623 - val_accuracy: 0.9503
Epoch 14/50
58/58 [==============================] - 4s 63ms/step - loss: 0.0108 - accuracy:
0.9999 - val_loss: 0.1607 - val_accuracy: 0.9520
Epoch 15/50
```

```
58/58 [==============================] - 4s 64ms/step - loss: 0.0095 - accuracy:
0.9998 - val_loss: 0.1618 - val_accuracy: 0.9523
Epoch 16/50
58/58 [==============================] - 4s 70ms/step - loss: 0.0076 - accuracy:
0.9999 - val_loss: 0.1623 - val_accuracy: 0.9523
Epoch 17/50
58/58 [==============================] - 4s 63ms/step - loss: 0.0071 - accuracy:
1.0000 - val_loss: 0.1638 - val_accuracy: 0.9527
Epoch 18/50
58/58 [==============================] - 4s 63ms/step - loss: 0.0058 - accuracy:
1.0000 - val_loss: 0.1645 - val_accuracy: 0.9540
Epoch 19/50
58/58 [==============================] - 4s 62ms/step - loss: 0.0052 - accuracy:
1.0000 - val_loss: 0.1636 - val_accuracy: 0.9530
Epoch 20/50
58/58 [==============================] - 4s 69ms/step - loss: 0.0045 - accuracy:
1.0000 - val_loss: 0.1636 - val_accuracy: 0.9550
Epoch 21/50
58/58 [==============================] - 4s 64ms/step - loss: 0.0041 - accuracy:
1.0000 - val_loss: 0.1640 - val_accuracy: 0.9557
Epoch 22/50
58/58 [==============================] - 4s 62ms/step - loss: 0.0039 - accuracy:
1.0000 - val_loss: 0.1654 - val_accuracy: 0.9540
Epoch 23/50
58/58 [==============================] - 4s 63ms/step - loss: 0.0034 - accuracy:
1.0000 - val_loss: 0.1683 - val_accuracy: 0.9540
Epoch 24/50
58/58 [==============================] - 4s 71ms/step - loss: 0.0030 - accuracy:
1.0000 - val_loss: 0.1676 - val_accuracy: 0.9547
Epoch 25/50
58/58 [==============================] - 4s 64ms/step - loss: 0.0028 - accuracy:
1.0000 - val_loss: 0.1679 - val_accuracy: 0.9540
Epoch 26/50
58/58 [==============================] - 4s 64ms/step - loss: 0.0025 - accuracy:
1.0000 - val_loss: 0.1696 - val_accuracy: 0.9540
Epoch 27/50
58/58 [==============================] - 4s 64ms/step - loss: 0.0023 - accuracy:
1.0000 - val_loss: 0.1686 - val_accuracy: 0.9547
Epoch 28/50
58/58 [==============================] - 4s 70ms/step - loss: 0.0021 - accuracy:
1.0000 - val_loss: 0.1683 - val_accuracy: 0.9563
Epoch 29/50
58/58 [==============================] - 4s 65ms/step - loss: 0.0020 - accuracy:
1.0000 - val_loss: 0.1702 - val_accuracy: 0.9547
Epoch 30/50
58/58 [==============================] - 4s 62ms/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 0.1697 - val_accuracy: 0.9550
Epoch 31/50
```

```
58/58 [==============================] - 4s 62ms/step - loss: 0.0017 - accuracy:
1.0000 - val_loss: 0.1723 - val_accuracy: 0.9550
Saved model to disk
```

```
[ ]: visualize_learning_curve(H, lb = pretrain_exp)
```



```
[ ]: # Evaluando modelo pre entrenado

     evaluate_model(pre_trained_model, x_test_preprocessed, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 4s 67ms/step
              precision    recall  f1-score   support

           0       0.95      0.95      0.95       200
           1       0.96      0.94      0.95       200
           2       0.98      0.98      0.98       200
           3       0.96      0.96      0.96       200
           4       0.97      0.99      0.98       200
           5       0.92      0.94      0.93       200
           6       0.97      0.94      0.95       200
           7       0.92      0.94      0.93       200
           8       0.96      0.93      0.94       200
           9       0.93      0.95      0.94       200
          10       0.98      0.97      0.98       200
          11       0.98      0.98      0.98       200
          12       0.95      0.94      0.95       200
          13       0.96      0.95      0.96       200
          14       0.94      0.95      0.95       200

    accuracy                           0.96      3000
   macro avg       0.96      0.96      0.96      3000
weighted avg       0.96      0.96      0.96      3000
```

###**6.1.4. ResNet50**

[ ]:

```python
# Obtención modelo pre entrenado InceptionV3

base = 'ResNet50'
base_model, x_train_preprocessed, x_val_preprocessed, x_test_preprocessed =␣
 ↪get_base_model(base, x_train, x_val, x_test)

pretrain_exp = base + "_TL_" + str(batch_size) + "_" + str(epochs)
exp_set.add(pretrain_exp)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 3s 0us/step
Model: "resnet50"
________________________________________________________________________________
__________________
 Layer (type)                Output Shape                 Param #   Connected to
================================================================================
==================
 input_4 (InputLayer)        [(None, 75, 75, 3)]          0         []

 conv1_pad (ZeroPadding2D)   (None, 81, 81, 3)            0
['input_4[0][0]']

 conv1_conv (Conv2D)         (None, 38, 38, 64)           9472
['conv1_pad[0][0]']

 conv1_bn (BatchNormalizati  (None, 38, 38, 64)           256
['conv1_conv[0][0]']
 on)

 conv1_relu (Activation)     (None, 38, 38, 64)           0
```

```
['conv1_bn[0][0]']

 pool1_pad (ZeroPadding2D)   (None, 40, 40, 64)           0
['conv1_relu[0][0]']

 pool1_pool (MaxPooling2D)   (None, 19, 19, 64)           0
['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2  (None, 19, 19, 64)           4160
['pool1_pool[0][0]']
 D)

 conv2_block1_1_bn (BatchNo  (None, 19, 19, 64)           256
['conv2_block1_1_conv[0][0]']
 rmalization)

 conv2_block1_1_relu (Activ  (None, 19, 19, 64)           0
['conv2_block1_1_bn[0][0]']
 ation)

 conv2_block1_2_conv (Conv2  (None, 19, 19, 64)           36928
['conv2_block1_1_relu[0][0]']
 D)

 conv2_block1_2_bn (BatchNo  (None, 19, 19, 64)           256
['conv2_block1_2_conv[0][0]']
 rmalization)

 conv2_block1_2_relu (Activ  (None, 19, 19, 64)           0
['conv2_block1_2_bn[0][0]']
 ation)

 conv2_block1_0_conv (Conv2  (None, 19, 19, 256)          16640
['pool1_pool[0][0]']
 D)

 conv2_block1_3_conv (Conv2  (None, 19, 19, 256)          16640
['conv2_block1_2_relu[0][0]']
 D)

 conv2_block1_0_bn (BatchNo  (None, 19, 19, 256)          1024
['conv2_block1_0_conv[0][0]']
 rmalization)

 conv2_block1_3_bn (BatchNo  (None, 19, 19, 256)          1024
['conv2_block1_3_conv[0][0]']
 rmalization)
 conv2_block1_add (Add)      (None, 19, 19, 256)          0
['conv2_block1_0_bn[0][0]',
'conv2_block1_3_bn[0][0]']

 conv2_block1_out (Activati  (None, 19, 19, 256)          0
['conv2_block1_add[0][0]']
 on)

 conv2_block2_1_conv (Conv2  (None, 19, 19, 64)           16448
['conv2_block1_out[0][0]']
 D)

 conv2_block2_1_bn (BatchNo  (None, 19, 19, 64)           256
['conv2_block2_1_conv[0][0]']
 rmalization)

 conv2_block2_1_relu (Activ  (None, 19, 19, 64)           0
['conv2_block2_1_bn[0][0]']
 ation)

 conv2_block2_2_conv (Conv2  (None, 19, 19, 64)           36928
['conv2_block2_1_relu[0][0]']
 D)

 conv2_block2_2_bn (BatchNo  (None, 19, 19, 64)           256
['conv2_block2_2_conv[0][0]']
 rmalization)

 conv2_block2_2_relu (Activ  (None, 19, 19, 64)           0
['conv2_block2_2_bn[0][0]']
 ation)

 conv2_block2_3_conv (Conv2  (None, 19, 19, 256)          16640
['conv2_block2_2_relu[0][0]']
 D)

 conv2_block2_3_bn (BatchNo  (None, 19, 19, 256)          1024
['conv2_block2_3_conv[0][0]']
 rmalization)

 conv2_block2_add (Add)      (None, 19, 19, 256)          0
['conv2_block1_out[0][0]',
'conv2_block2_3_bn[0][0]']

 conv2_block2_out (Activati  (None, 19, 19, 256)          0
['conv2_block2_add[0][0]']
 on)
 conv2_block3_1_conv (Conv2  (None, 19, 19, 64)           16448
['conv2_block2_out[0][0]']
 D)

 conv2_block3_1_bn (BatchNo  (None, 19, 19, 64)           256
['conv2_block3_1_conv[0][0]']
 rmalization)

 conv2_block3_1_relu (Activ  (None, 19, 19, 64)           0
['conv2_block3_1_bn[0][0]']
 ation)

 conv2_block3_2_conv (Conv2  (None, 19, 19, 64)           36928
['conv2_block3_1_relu[0][0]']
 D)

 conv2_block3_2_bn (BatchNo  (None, 19, 19, 64)           256
['conv2_block3_2_conv[0][0]']
 rmalization)

 conv2_block3_2_relu (Activ  (None, 19, 19, 64)           0
['conv2_block3_2_bn[0][0]']
 ation)

 conv2_block3_3_conv (Conv2  (None, 19, 19, 256)          16640
['conv2_block3_2_relu[0][0]']
 D)

 conv2_block3_3_bn (BatchNo  (None, 19, 19, 256)          1024
['conv2_block3_3_conv[0][0]']
 rmalization)

 conv2_block3_add (Add)      (None, 19, 19, 256)          0
['conv2_block2_out[0][0]',
'conv2_block3_3_bn[0][0]']

 conv2_block3_out (Activati  (None, 19, 19, 256)          0
['conv2_block3_add[0][0]']
 on)

 conv3_block1_1_conv (Conv2  (None, 10, 10, 128)          32896
['conv2_block3_out[0][0]']
 D)

 conv3_block1_1_bn (BatchNo  (None, 10, 10, 128)          512
['conv3_block1_1_conv[0][0]']
 rmalization)
 conv3_block1_1_relu (Activ  (None, 10, 10, 128)          0
['conv3_block1_1_bn[0][0]']
 ation)

 conv3_block1_2_conv (Conv2  (None, 10, 10, 128)          147584
['conv3_block1_1_relu[0][0]']
 D)

 conv3_block1_2_bn (BatchNo  (None, 10, 10, 128)          512
['conv3_block1_2_conv[0][0]']
 rmalization)

 conv3_block1_2_relu (Activ  (None, 10, 10, 128)          0
['conv3_block1_2_bn[0][0]']
 ation)

 conv3_block1_0_conv (Conv2  (None, 10, 10, 512)          131584
['conv2_block3_out[0][0]']
 D)

 conv3_block1_3_conv (Conv2  (None, 10, 10, 512)          66048
['conv3_block1_2_relu[0][0]']
 D)

 conv3_block1_0_bn (BatchNo  (None, 10, 10, 512)          2048
['conv3_block1_0_conv[0][0]']
 rmalization)

 conv3_block1_3_bn (BatchNo  (None, 10, 10, 512)          2048
['conv3_block1_3_conv[0][0]']
 rmalization)

 conv3_block1_add (Add)      (None, 10, 10, 512)          0
['conv3_block1_0_bn[0][0]',
'conv3_block1_3_bn[0][0]']

 conv3_block1_out (Activati  (None, 10, 10, 512)          0
['conv3_block1_add[0][0]']
 on)

 conv3_block2_1_conv (Conv2  (None, 10, 10, 128)          65664
['conv3_block1_out[0][0]']
 D)

 conv3_block2_1_bn (BatchNo  (None, 10, 10, 128)          512
['conv3_block2_1_conv[0][0]']
 rmalization)
 conv3_block2_1_relu (Activ  (None, 10, 10, 128)         0
['conv3_block2_1_bn[0][0]']
 ation)

 conv3_block2_2_conv (Conv2  (None, 10, 10, 128)         147584
['conv3_block2_1_relu[0][0]']
 D)

 conv3_block2_2_bn (BatchNo  (None, 10, 10, 128)         512
['conv3_block2_2_conv[0][0]']
 rmalization)

 conv3_block2_2_relu (Activ  (None, 10, 10, 128)         0
['conv3_block2_2_bn[0][0]']
 ation)

 conv3_block2_3_conv (Conv2  (None, 10, 10, 512)         66048
['conv3_block2_2_relu[0][0]']
 D)

 conv3_block2_3_bn (BatchNo  (None, 10, 10, 512)         2048
['conv3_block2_3_conv[0][0]']
 rmalization)

 conv3_block2_add (Add)      (None, 10, 10, 512)         0
['conv3_block1_out[0][0]',
'conv3_block2_3_bn[0][0]']

 conv3_block2_out (Activati  (None, 10, 10, 512)         0
['conv3_block2_add[0][0]']
 on)

 conv3_block3_1_conv (Conv2  (None, 10, 10, 128)         65664
['conv3_block2_out[0][0]']
 D)

 conv3_block3_1_bn (BatchNo  (None, 10, 10, 128)         512
['conv3_block3_1_conv[0][0]']
 rmalization)

 conv3_block3_1_relu (Activ  (None, 10, 10, 128)         0
['conv3_block3_1_bn[0][0]']
 ation)

 conv3_block3_2_conv (Conv2  (None, 10, 10, 128)         147584
['conv3_block3_1_relu[0][0]']
 D)
 conv3_block3_2_bn (BatchNo  (None, 10, 10, 128)         512
['conv3_block3_2_conv[0][0]']
 rmalization)

 conv3_block3_2_relu (Activ  (None, 10, 10, 128)         0
['conv3_block3_2_bn[0][0]']
 ation)

 conv3_block3_3_conv (Conv2  (None, 10, 10, 512)         66048
['conv3_block3_2_relu[0][0]']
 D)

 conv3_block3_3_bn (BatchNo  (None, 10, 10, 512)         2048
['conv3_block3_3_conv[0][0]']
 rmalization)

 conv3_block3_add (Add)      (None, 10, 10, 512)         0
['conv3_block2_out[0][0]',
'conv3_block3_3_bn[0][0]']

 conv3_block3_out (Activati  (None, 10, 10, 512)         0
['conv3_block3_add[0][0]']
 on)

 conv3_block4_1_conv (Conv2  (None, 10, 10, 128)         65664
['conv3_block3_out[0][0]']
 D)

 conv3_block4_1_bn (BatchNo  (None, 10, 10, 128)         512
['conv3_block4_1_conv[0][0]']
 rmalization)

 conv3_block4_1_relu (Activ  (None, 10, 10, 128)         0
['conv3_block4_1_bn[0][0]']
 ation)

 conv3_block4_2_conv (Conv2  (None, 10, 10, 128)         147584
['conv3_block4_1_relu[0][0]']
 D)

 conv3_block4_2_bn (BatchNo  (None, 10, 10, 128)         512
['conv3_block4_2_conv[0][0]']
 rmalization)

 conv3_block4_2_relu (Activ  (None, 10, 10, 128)         0
['conv3_block4_2_bn[0][0]']
 ation)
 conv3_block4_3_conv (Conv2  (None, 10, 10, 512)          66048
['conv3_block4_2_relu[0][0]']
 D)

 conv3_block4_3_bn (BatchNo  (None, 10, 10, 512)          2048
['conv3_block4_3_conv[0][0]']
 rmalization)

 conv3_block4_add (Add)      (None, 10, 10, 512)          0
['conv3_block3_out[0][0]',
'conv3_block4_3_bn[0][0]']

 conv3_block4_out (Activati  (None, 10, 10, 512)          0
['conv3_block4_add[0][0]']
 on)

 conv4_block1_1_conv (Conv2  (None, 5, 5, 256)            131328
['conv3_block4_out[0][0]']
 D)

 conv4_block1_1_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block1_1_conv[0][0]']
 rmalization)

 conv4_block1_1_relu (Activ  (None, 5, 5, 256)            0
['conv4_block1_1_bn[0][0]']
 ation)

 conv4_block1_2_conv (Conv2  (None, 5, 5, 256)            590080
['conv4_block1_1_relu[0][0]']
 D)

 conv4_block1_2_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block1_2_conv[0][0]']
 rmalization)

 conv4_block1_2_relu (Activ  (None, 5, 5, 256)            0
['conv4_block1_2_bn[0][0]']
 ation)

 conv4_block1_0_conv (Conv2  (None, 5, 5, 1024)           525312
['conv3_block4_out[0][0]']
 D)

 conv4_block1_3_conv (Conv2  (None, 5, 5, 1024)           263168
['conv4_block1_2_relu[0][0]']
 D)
 conv4_block1_0_bn (BatchNo  (None, 5, 5, 1024)           4096
['conv4_block1_0_conv[0][0]']
 rmalization)

 conv4_block1_3_bn (BatchNo  (None, 5, 5, 1024)           4096
['conv4_block1_3_conv[0][0]']
 rmalization)

 conv4_block1_add (Add)      (None, 5, 5, 1024)           0
['conv4_block1_0_bn[0][0]',
'conv4_block1_3_bn[0][0]']

 conv4_block1_out (Activati  (None, 5, 5, 1024)           0
['conv4_block1_add[0][0]']
 on)

 conv4_block2_1_conv (Conv2  (None, 5, 5, 256)            262400
['conv4_block1_out[0][0]']
 D)

 conv4_block2_1_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block2_1_conv[0][0]']
 rmalization)

 conv4_block2_1_relu (Activ  (None, 5, 5, 256)            0
['conv4_block2_1_bn[0][0]']
 ation)

 conv4_block2_2_conv (Conv2  (None, 5, 5, 256)            590080
['conv4_block2_1_relu[0][0]']
 D)

 conv4_block2_2_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block2_2_conv[0][0]']
 rmalization)

 conv4_block2_2_relu (Activ  (None, 5, 5, 256)            0
['conv4_block2_2_bn[0][0]']
 ation)

 conv4_block2_3_conv (Conv2  (None, 5, 5, 1024)           263168
['conv4_block2_2_relu[0][0]']
 D)

 conv4_block2_3_bn (BatchNo  (None, 5, 5, 1024)           4096
['conv4_block2_3_conv[0][0]']
 rmalization)
 conv4_block2_add (Add)      (None, 5, 5, 1024)         0
['conv4_block1_out[0][0]',
'conv4_block2_3_bn[0][0]']

 conv4_block2_out (Activati  (None, 5, 5, 1024)         0
['conv4_block2_add[0][0]']
 on)

 conv4_block3_1_conv (Conv2  (None, 5, 5, 256)          262400
['conv4_block2_out[0][0]']
 D)

 conv4_block3_1_bn (BatchNo  (None, 5, 5, 256)          1024
['conv4_block3_1_conv[0][0]']
 rmalization)

 conv4_block3_1_relu (Activ  (None, 5, 5, 256)          0
['conv4_block3_1_bn[0][0]']
 ation)

 conv4_block3_2_conv (Conv2  (None, 5, 5, 256)          590080
['conv4_block3_1_relu[0][0]']
 D)

 conv4_block3_2_bn (BatchNo  (None, 5, 5, 256)          1024
['conv4_block3_2_conv[0][0]']
 rmalization)

 conv4_block3_2_relu (Activ  (None, 5, 5, 256)          0
['conv4_block3_2_bn[0][0]']
 ation)

 conv4_block3_3_conv (Conv2  (None, 5, 5, 1024)         263168
['conv4_block3_2_relu[0][0]']
 D)

 conv4_block3_3_bn (BatchNo  (None, 5, 5, 1024)         4096
['conv4_block3_3_conv[0][0]']
 rmalization)

 conv4_block3_add (Add)      (None, 5, 5, 1024)         0
['conv4_block2_out[0][0]',
'conv4_block3_3_bn[0][0]']

 conv4_block3_out (Activati  (None, 5, 5, 1024)         0
['conv4_block3_add[0][0]']
 on)
 conv4_block4_1_conv (Conv2  (None, 5, 5, 256)            262400
['conv4_block3_out[0][0]']
 D)

 conv4_block4_1_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block4_1_conv[0][0]']
 rmalization)

 conv4_block4_1_relu (Activ  (None, 5, 5, 256)            0
['conv4_block4_1_bn[0][0]']
 ation)

 conv4_block4_2_conv (Conv2  (None, 5, 5, 256)            590080
['conv4_block4_1_relu[0][0]']
 D)

 conv4_block4_2_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block4_2_conv[0][0]']
 rmalization)

 conv4_block4_2_relu (Activ  (None, 5, 5, 256)            0
['conv4_block4_2_bn[0][0]']
 ation)

 conv4_block4_3_conv (Conv2  (None, 5, 5, 1024)           263168
['conv4_block4_2_relu[0][0]']
 D)

 conv4_block4_3_bn (BatchNo  (None, 5, 5, 1024)           4096
['conv4_block4_3_conv[0][0]']
 rmalization)

 conv4_block4_add (Add)      (None, 5, 5, 1024)           0
['conv4_block3_out[0][0]',
'conv4_block4_3_bn[0][0]']

 conv4_block4_out (Activati  (None, 5, 5, 1024)           0
['conv4_block4_add[0][0]']
 on)

 conv4_block5_1_conv (Conv2  (None, 5, 5, 256)            262400
['conv4_block4_out[0][0]']
 D)

 conv4_block5_1_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block5_1_conv[0][0]']
 rmalization)
 conv4_block5_1_relu (Activ  (None, 5, 5, 256)            0
['conv4_block5_1_bn[0][0]']
 ation)

 conv4_block5_2_conv (Conv2  (None, 5, 5, 256)            590080
['conv4_block5_1_relu[0][0]']
 D)

 conv4_block5_2_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block5_2_conv[0][0]']
 rmalization)

 conv4_block5_2_relu (Activ  (None, 5, 5, 256)            0
['conv4_block5_2_bn[0][0]']
 ation)

 conv4_block5_3_conv (Conv2  (None, 5, 5, 1024)           263168
['conv4_block5_2_relu[0][0]']
 D)

 conv4_block5_3_bn (BatchNo  (None, 5, 5, 1024)           4096
['conv4_block5_3_conv[0][0]']
 rmalization)

 conv4_block5_add (Add)      (None, 5, 5, 1024)           0
['conv4_block4_out[0][0]',
'conv4_block5_3_bn[0][0]']

 conv4_block5_out (Activati  (None, 5, 5, 1024)           0
['conv4_block5_add[0][0]']
 on)

 conv4_block6_1_conv (Conv2  (None, 5, 5, 256)            262400
['conv4_block5_out[0][0]']
 D)

 conv4_block6_1_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block6_1_conv[0][0]']
 rmalization)

 conv4_block6_1_relu (Activ  (None, 5, 5, 256)            0
['conv4_block6_1_bn[0][0]']
 ation)

 conv4_block6_2_conv (Conv2  (None, 5, 5, 256)            590080
['conv4_block6_1_relu[0][0]']
 D)
 conv4_block6_2_bn (BatchNo  (None, 5, 5, 256)            1024
['conv4_block6_2_conv[0][0]']
 rmalization)

 conv4_block6_2_relu (Activ  (None, 5, 5, 256)            0
['conv4_block6_2_bn[0][0]']
 ation)

 conv4_block6_3_conv (Conv2  (None, 5, 5, 1024)           263168
['conv4_block6_2_relu[0][0]']
 D)

 conv4_block6_3_bn (BatchNo  (None, 5, 5, 1024)           4096
['conv4_block6_3_conv[0][0]']
 rmalization)

 conv4_block6_add (Add)      (None, 5, 5, 1024)           0
['conv4_block5_out[0][0]',
'conv4_block6_3_bn[0][0]']

 conv4_block6_out (Activati  (None, 5, 5, 1024)           0
['conv4_block6_add[0][0]']
 on)

 conv5_block1_1_conv (Conv2  (None, 3, 3, 512)            524800
['conv4_block6_out[0][0]']
 D)

 conv5_block1_1_bn (BatchNo  (None, 3, 3, 512)            2048
['conv5_block1_1_conv[0][0]']
 rmalization)

 conv5_block1_1_relu (Activ  (None, 3, 3, 512)            0
['conv5_block1_1_bn[0][0]']
 ation)

 conv5_block1_2_conv (Conv2  (None, 3, 3, 512)            2359808
['conv5_block1_1_relu[0][0]']
 D)

 conv5_block1_2_bn (BatchNo  (None, 3, 3, 512)            2048
['conv5_block1_2_conv[0][0]']
 rmalization)

 conv5_block1_2_relu (Activ  (None, 3, 3, 512)            0
['conv5_block1_2_bn[0][0]']
 ation)
 conv5_block1_0_conv (Conv2  (None, 3, 3, 2048)          2099200
['conv4_block6_out[0][0]']
 D)

 conv5_block1_3_conv (Conv2  (None, 3, 3, 2048)          1050624
['conv5_block1_2_relu[0][0]']
 D)

 conv5_block1_0_bn (BatchNo  (None, 3, 3, 2048)          8192
['conv5_block1_0_conv[0][0]']
 rmalization)

 conv5_block1_3_bn (BatchNo  (None, 3, 3, 2048)          8192
['conv5_block1_3_conv[0][0]']
 rmalization)

 conv5_block1_add (Add)      (None, 3, 3, 2048)          0
['conv5_block1_0_bn[0][0]',
'conv5_block1_3_bn[0][0]']

 conv5_block1_out (Activati  (None, 3, 3, 2048)          0
['conv5_block1_add[0][0]']
 on)

 conv5_block2_1_conv (Conv2  (None, 3, 3, 512)           1049088
['conv5_block1_out[0][0]']
 D)

 conv5_block2_1_bn (BatchNo  (None, 3, 3, 512)           2048
['conv5_block2_1_conv[0][0]']
 rmalization)

 conv5_block2_1_relu (Activ  (None, 3, 3, 512)           0
['conv5_block2_1_bn[0][0]']
 ation)

 conv5_block2_2_conv (Conv2  (None, 3, 3, 512)           2359808
['conv5_block2_1_relu[0][0]']
 D)

 conv5_block2_2_bn (BatchNo  (None, 3, 3, 512)           2048
['conv5_block2_2_conv[0][0]']
 rmalization)

 conv5_block2_2_relu (Activ  (None, 3, 3, 512)           0
['conv5_block2_2_bn[0][0]']
 ation)
 conv5_block2_3_conv (Conv2  (None, 3, 3, 2048)           1050624
['conv5_block2_2_relu[0][0]']
 D)

 conv5_block2_3_bn (BatchNo  (None, 3, 3, 2048)           8192
['conv5_block2_3_conv[0][0]']
 rmalization)

 conv5_block2_add (Add)      (None, 3, 3, 2048)           0
['conv5_block1_out[0][0]',
'conv5_block2_3_bn[0][0]']

 conv5_block2_out (Activati  (None, 3, 3, 2048)           0
['conv5_block2_add[0][0]']
 on)

 conv5_block3_1_conv (Conv2  (None, 3, 3, 512)            1049088
['conv5_block2_out[0][0]']
 D)

 conv5_block3_1_bn (BatchNo  (None, 3, 3, 512)            2048
['conv5_block3_1_conv[0][0]']
 rmalization)

 conv5_block3_1_relu (Activ  (None, 3, 3, 512)            0
['conv5_block3_1_bn[0][0]']
 ation)

 conv5_block3_2_conv (Conv2  (None, 3, 3, 512)            2359808
['conv5_block3_1_relu[0][0]']
 D)

 conv5_block3_2_bn (BatchNo  (None, 3, 3, 512)            2048
['conv5_block3_2_conv[0][0]']
 rmalization)

 conv5_block3_2_relu (Activ  (None, 3, 3, 512)            0
['conv5_block3_2_bn[0][0]']
 ation)

 conv5_block3_3_conv (Conv2  (None, 3, 3, 2048)           1050624
['conv5_block3_2_relu[0][0]']
 D)

 conv5_block3_3_bn (BatchNo  (None, 3, 3, 2048)           8192
['conv5_block3_3_conv[0][0]']
 rmalization)
 conv5_block3_add (Add)      (None, 3, 3, 2048)           0
['conv5_block2_out[0][0]',
'conv5_block3_3_bn[0][0]']

 conv5_block3_out (Activati  (None, 3, 3, 2048)           0
['conv5_block3_add[0][0]']
 on)

=================================================================================
==================
Total params: 23587712 (89.98 MB)
Trainable params: 23534592 (89.78 MB)
Non-trainable params: 53120 (207.50 KB)
---------------------------------------------------------------------------------
------------------
```

[ ]:

```python
# Entrenamiento modelo pre entrenado

if do_training == True:

  pre_trained_model = get_pretrained_model(base_model, dense_size, num_clases)

  pre_trained_model.compile(optimizer=Adam(learning_rate=learning_rate),
                            loss="categorical_crossentropy",
                            metrics=["accuracy"])

  print("[INFO]: Entrenando Top Model sobre ", base, " ...")
  H = pre_trained_model.fit(x_train_preprocessed, y_train_ohe,
                            batch_size=batch_size,
                            epochs=epochs,
                            steps_per_epoch=x_train_preprocessed.shape[0] //␣
 ↪batch_size,
                            validation_data=(x_val_preprocessed, y_val_ohe),
                            callbacks=[early_stopping_cbck])

  save_trained_model(model = pre_trained_model, history = H, model_name =␣
 ↪pretrain_exp)

else:
  print("[INFO]: Cargando Top Model sobre " + base + "....")
  pre_trained_model = load_keras_model(pretrain_exp)
  pre_trained_model.summary()
  H = load_history(pretrain_exp)
```

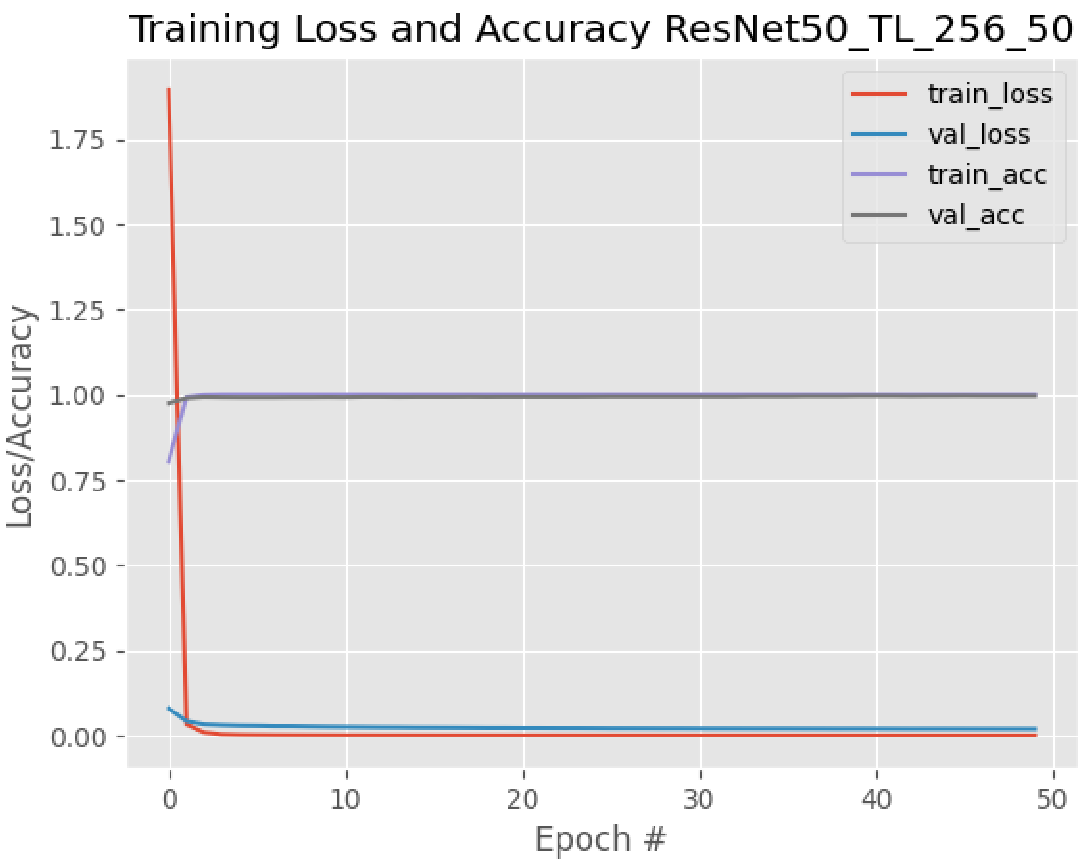
```
Model: "sequential_2"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 3, 3, 2048)        23587712

 flatten_2 (Flatten)         (None, 18432)             0

 dense_4 (Dense)             (None, 128)               2359424

 dense_5 (Dense)             (None, 15)                1935

=================================================================
Total params: 25949071 (98.99 MB)
Trainable params: 2361359 (9.01 MB)
Non-trainable params: 23587712 (89.98 MB)
_________________________________________________________________
[INFO]: Entrenando Top Model sobre  ResNet50  …
Epoch 1/50
58/58 [==============================] - 18s 198ms/step - loss: 1.8946 -
accuracy: 0.8045 - val_loss: 0.0792 - val_accuracy: 0.9743
Epoch 2/50
58/58 [==============================] - 9s 137ms/step - loss: 0.0326 -
accuracy: 0.9917 - val_loss: 0.0419 - val_accuracy: 0.9887
Epoch 3/50
58/58 [==============================] - 8s 143ms/step - loss: 0.0091 -
accuracy: 0.9985 - val_loss: 0.0327 - val_accuracy: 0.9920
Epoch 4/50
58/58 [==============================] - 8s 138ms/step - loss: 0.0034 -
accuracy: 0.9999 - val_loss: 0.0306 - val_accuracy: 0.9913
Epoch 5/50
58/58 [==============================] - 8s 144ms/step - loss: 0.0021 -
accuracy: 1.0000 - val_loss: 0.0291 - val_accuracy: 0.9910
Epoch 6/50
58/58 [==============================] - 8s 139ms/step - loss: 0.0016 -
accuracy: 1.0000 - val_loss: 0.0286 - val_accuracy: 0.9910
Epoch 7/50
58/58 [==============================] - 8s 147ms/step - loss: 0.0012 -
accuracy: 1.0000 - val_loss: 0.0275 - val_accuracy: 0.9910
Epoch 8/50
58/58 [==============================] - 8s 139ms/step - loss: 9.9686e-04 -
accuracy: 1.0000 - val_loss: 0.0273 - val_accuracy: 0.9913
Epoch 9/50
58/58 [==============================] - 8s 144ms/step - loss: 8.4769e-04 -
accuracy: 1.0000 - val_loss: 0.0264 - val_accuracy: 0.9913
Epoch 10/50
58/58 [==============================] - 8s 140ms/step - loss: 7.2824e-04 -
accuracy: 1.0000 - val_loss: 0.0257 - val_accuracy: 0.9917
Epoch 11/50
58/58 [==============================] - 10s 167ms/step - loss: 5.8659e-04 -
accuracy: 1.0000 - val_loss: 0.0253 - val_accuracy: 0.9913
Epoch 12/50
```

```
58/58 [==============================] - 9s 161ms/step - loss: 5.7396e-04 -
accuracy: 1.0000 - val_loss: 0.0248 - val_accuracy: 0.9923
Epoch 13/50
58/58 [==============================] - 8s 144ms/step - loss: 4.6750e-04 -
accuracy: 1.0000 - val_loss: 0.0243 - val_accuracy: 0.9923
Epoch 14/50
58/58 [==============================] - 8s 142ms/step - loss: 4.3054e-04 -
accuracy: 1.0000 - val_loss: 0.0241 - val_accuracy: 0.9920
Epoch 15/50
58/58 [==============================] - 8s 144ms/step - loss: 3.8517e-04 -
accuracy: 1.0000 - val_loss: 0.0235 - val_accuracy: 0.9923
Epoch 16/50
58/58 [==============================] - 8s 142ms/step - loss: 3.4770e-04 -
accuracy: 1.0000 - val_loss: 0.0234 - val_accuracy: 0.9923
Epoch 17/50
58/58 [==============================] - 10s 165ms/step - loss: 3.1323e-04 -
accuracy: 1.0000 - val_loss: 0.0231 - val_accuracy: 0.9923
Epoch 18/50
58/58 [==============================] - 8s 143ms/step - loss: 2.8126e-04 -
accuracy: 1.0000 - val_loss: 0.0228 - val_accuracy: 0.9927
Epoch 19/50
58/58 [==============================] - 8s 142ms/step - loss: 2.6310e-04 -
accuracy: 1.0000 - val_loss: 0.0224 - val_accuracy: 0.9923
Epoch 20/50
58/58 [==============================] - 9s 163ms/step - loss: 2.3990e-04 -
accuracy: 1.0000 - val_loss: 0.0224 - val_accuracy: 0.9927
Epoch 21/50
58/58 [==============================] - 9s 163ms/step - loss: 2.2265e-04 -
accuracy: 1.0000 - val_loss: 0.0220 - val_accuracy: 0.9927
Epoch 22/50
58/58 [==============================] - 8s 145ms/step - loss: 2.0301e-04 -
accuracy: 1.0000 - val_loss: 0.0220 - val_accuracy: 0.9927
Epoch 23/50
58/58 [==============================] - 9s 162ms/step - loss: 1.8979e-04 -
accuracy: 1.0000 - val_loss: 0.0217 - val_accuracy: 0.9927
Epoch 24/50
58/58 [==============================] - 8s 145ms/step - loss: 1.7258e-04 -
accuracy: 1.0000 - val_loss: 0.0217 - val_accuracy: 0.9927
Epoch 25/50
58/58 [==============================] - 9s 162ms/step - loss: 1.6544e-04 -
accuracy: 1.0000 - val_loss: 0.0215 - val_accuracy: 0.9930
Epoch 26/50
58/58 [==============================] - 10s 168ms/step - loss: 1.5540e-04 -
accuracy: 1.0000 - val_loss: 0.0212 - val_accuracy: 0.9933
Epoch 27/50
58/58 [==============================] - 8s 141ms/step - loss: 1.3819e-04 -
accuracy: 1.0000 - val_loss: 0.0212 - val_accuracy: 0.9933
Epoch 28/50
```

```
58/58 [==============================] - 10s 166ms/step - loss: 1.3360e-04 -
accuracy: 1.0000 - val_loss: 0.0211 - val_accuracy: 0.9933
Epoch 29/50
58/58 [==============================] - 8s 141ms/step - loss: 1.2380e-04 -
accuracy: 1.0000 - val_loss: 0.0209 - val_accuracy: 0.9933
Epoch 30/50
58/58 [==============================] - 10s 165ms/step - loss: 1.1808e-04 -
accuracy: 1.0000 - val_loss: 0.0209 - val_accuracy: 0.9933
Epoch 31/50
58/58 [==============================] - 9s 163ms/step - loss: 1.1203e-04 -
accuracy: 1.0000 - val_loss: 0.0207 - val_accuracy: 0.9933
Epoch 32/50
58/58 [==============================] - 8s 141ms/step - loss: 1.0162e-04 -
accuracy: 1.0000 - val_loss: 0.0206 - val_accuracy: 0.9933
Epoch 33/50
58/58 [==============================] - 10s 166ms/step - loss: 9.9303e-05 -
accuracy: 1.0000 - val_loss: 0.0206 - val_accuracy: 0.9933
Epoch 34/50
58/58 [==============================] - 8s 140ms/step - loss: 9.4755e-05 -
accuracy: 1.0000 - val_loss: 0.0205 - val_accuracy: 0.9940
Epoch 35/50
58/58 [==============================] - 8s 146ms/step - loss: 8.6138e-05 -
accuracy: 1.0000 - val_loss: 0.0205 - val_accuracy: 0.9940
Epoch 36/50
58/58 [==============================] - 8s 140ms/step - loss: 8.5267e-05 -
accuracy: 1.0000 - val_loss: 0.0204 - val_accuracy: 0.9940
Epoch 37/50
58/58 [==============================] - 8s 145ms/step - loss: 8.0007e-05 -
accuracy: 1.0000 - val_loss: 0.0203 - val_accuracy: 0.9947
Epoch 38/50
58/58 [==============================] - 9s 162ms/step - loss: 7.3624e-05 -
accuracy: 1.0000 - val_loss: 0.0202 - val_accuracy: 0.9947
Epoch 39/50
58/58 [==============================] - 10s 167ms/step - loss: 7.1760e-05 -
accuracy: 1.0000 - val_loss: 0.0200 - val_accuracy: 0.9947
Epoch 40/50
58/58 [==============================] - 8s 141ms/step - loss: 6.7313e-05 -
accuracy: 1.0000 - val_loss: 0.0201 - val_accuracy: 0.9953
Epoch 41/50
58/58 [==============================] - 10s 167ms/step - loss: 6.4556e-05 -
accuracy: 1.0000 - val_loss: 0.0200 - val_accuracy: 0.9953
Epoch 42/50
58/58 [==============================] - 9s 164ms/step - loss: 6.2110e-05 -
accuracy: 1.0000 - val_loss: 0.0199 - val_accuracy: 0.9953
Epoch 43/50
58/58 [==============================] - 8s 142ms/step - loss: 5.8975e-05 -
accuracy: 1.0000 - val_loss: 0.0197 - val_accuracy: 0.9950
Epoch 44/50
```

```
58/58 [==============================] - 8s 144ms/step - loss: 5.5954e-05 -
accuracy: 1.0000 - val_loss: 0.0198 - val_accuracy: 0.9953
Epoch 45/50
58/58 [==============================] - 8s 143ms/step - loss: 5.3744e-05 -
accuracy: 1.0000 - val_loss: 0.0197 - val_accuracy: 0.9953
Epoch 46/50
58/58 [==============================] - 9s 164ms/step - loss: 5.2896e-05 -
accuracy: 1.0000 - val_loss: 0.0197 - val_accuracy: 0.9957
Epoch 47/50
58/58 [==============================] - 8s 140ms/step - loss: 4.9087e-05 -
accuracy: 1.0000 - val_loss: 0.0196 - val_accuracy: 0.9953
Epoch 48/50
58/58 [==============================] - 10s 166ms/step - loss: 4.6338e-05 -
accuracy: 1.0000 - val_loss: 0.0195 - val_accuracy: 0.9953
Epoch 49/50
58/58 [==============================] - 9s 162ms/step - loss: 4.4400e-05 -
accuracy: 1.0000 - val_loss: 0.0195 - val_accuracy: 0.9953
Epoch 50/50
58/58 [==============================] - 8s 145ms/step - loss: 4.4051e-05 -
accuracy: 1.0000 - val_loss: 0.0194 - val_accuracy: 0.9953
Saved model to disk
```

```
[ ]: visualize_learning_curve(H, lb = pretrain_exp)
```



[ ]:

```
evaluate_model(pre_trained_model, x_test_preprocessed, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 4s 84ms/step
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       200
           1       0.99      0.99      0.99       200
           2       1.00      0.99      0.99       200
           3       0.99      1.00      1.00       200
           4       0.99      0.99      0.99       200
           5       1.00      0.99      0.99       200
           6       0.99      0.99      0.99       200
           7       0.99      0.99      0.99       200
           8       0.99      0.99      0.99       200
           9       0.98      0.98      0.98       200
          10       1.00      1.00      1.00       200
          11       1.00      1.00      1.00       200
          12       0.99      0.99      0.99       200
          13       0.99      0.99      0.99       200
          14       0.98      0.99      0.99       200

    accuracy                           0.99      3000
   macro avg       0.99      0.99      0.99      3000
weighted avg       0.99      0.99      0.99      3000
```

COMENTARIO ResNet50: Con la arquitectura ResNet50 es de vital importancia preprocesar los datos de entrada en coherencia con esta arquitectura tal y como se hace mediante la función get_base_model. En base a otras pruebas realizadas, este procesamiento supone pasar de un score de 0.72 (sin preprocesar x_train), al 0.99 obtenido en el anterior experimento (el mejore score obtenido en el proyecto).

##6.2 Fine Tuning parcial

La función **get_fine_tuned_model** definida a continuación recibe un base_model, junto con el tamaño de la capa densa entrenable del top_model, el numero de clases de salida, y la primera capa entrenable del base_model, conectando ambas redes para realizar la tarea de fine tuning. Dado que estamos en una tarea de fine tuning parcial, se entrenarán los pesos del base_model a partir de la primera capa entrenable hasta el top_model, manteniendose el resto de pesos del base_model congelados.

[ ]:

```
def get_fine_tuned_model(base_model, first_trainable_layer, dense_size, num_clases):

  for layer in base_model.layers:
    if layer.name == first_trainable_layer:
      break
    layer.trainable = False
    print("Capa " + layer.name + " congelada...")

  fine_tuned_model = Sequential()
  fine_tuned_model.add(base_model)
  fine_tuned_model.add(layers.Flatten())
  fine_tuned_model.add(layers.Dense(dense_size, activation="relu"))
  fine_tuned_model.add(layers.Dense(num_clases, activation="softmax"))

  fine_tuned_model.summary()

  return fine_tuned_model
```

###6.2.1. VGG16

[ ]:

```
# Elegir modelo base
base = 'VGG16'
first_trainable_layer = "block4_conv1"

# Cargar modelo base y preprocesar datos
```

```
base_model, x_train_preprocessed, x_val_preprocessed, x_test_preprocessed =␣
 ↪get_base_model(base, x_train, x_val, x_test)

finetuning_exp = base + "_FT_" + first_trainable_layer + "_" + str(batch_size)␣
 ↪+ "_" + str(epochs)
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 75, 75, 3)]       0

 block1_conv1 (Conv2D)       (None, 75, 75, 64)        1792

 block1_conv2 (Conv2D)       (None, 75, 75, 64)        36928

 block1_pool (MaxPooling2D)  (None, 37, 37, 64)        0

 block2_conv1 (Conv2D)       (None, 37, 37, 128)       73856

 block2_conv2 (Conv2D)       (None, 37, 37, 128)       147584

 block2_pool (MaxPooling2D)  (None, 18, 18, 128)       0

 block3_conv1 (Conv2D)       (None, 18, 18, 256)       295168

 block3_conv2 (Conv2D)       (None, 18, 18, 256)       590080

 block3_conv3 (Conv2D)       (None, 18, 18, 256)       590080

 block3_pool (MaxPooling2D)  (None, 9, 9, 256)         0

 block4_conv1 (Conv2D)       (None, 9, 9, 512)         1180160

 block4_conv2 (Conv2D)       (None, 9, 9, 512)         2359808

 block4_conv3 (Conv2D)       (None, 9, 9, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0
```

```
=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

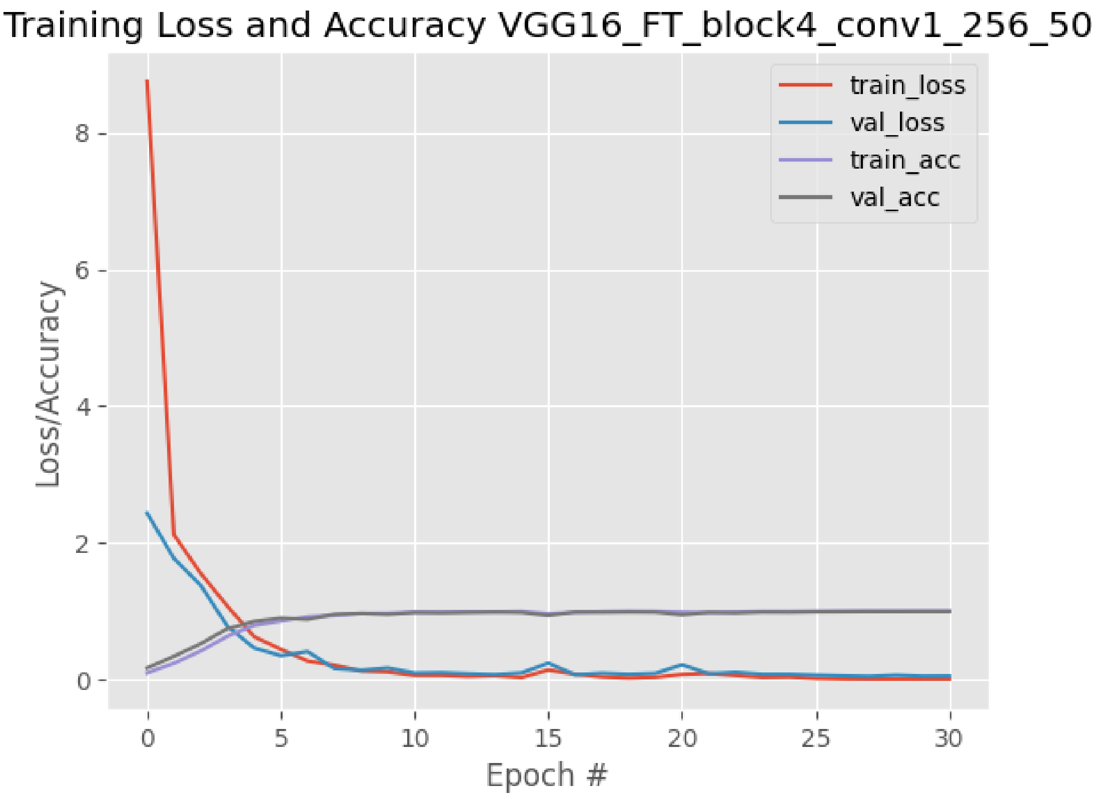
```python
# Compilamos el modelo y entrenamos

if do_training == True:

  fine_tuned_model = get_fine_tuned_model(base_model, first_trainable_layer,␣
 ↪dense_size, num_clases)

  fine_tuned_model.compile(optimizer=Adam(learning_rate=learning_rate),
                           loss="categorical_crossentropy",
                           metrics=["accuracy"])

  print("[INFO]: Entrenando " + base + " desde " + first_trainable_layer + " + ␣
 ↪Top Model ...")
  H = fine_tuned_model.fit(x_train_preprocessed, y_train_ohe,
                            batch_size=batch_size,
                            epochs=epochs,
                            steps_per_epoch=x_train_preprocessed.shape[0] //␣
 ↪batch_size,
                            validation_data=(x_val_preprocessed, y_val_ohe),
                            callbacks=[early_stopping_cbck])

  save_trained_model(model = fine_tuned_model, history = H, model_name =␣
 ↪finetuning_exp)

else:
  print("[INFO]: Cargando " + base + " desde " + first_trainable_layer + " + ␣
 ↪Top Model ...")
  fine_tuned_model = load_keras_model(finetuning_exp)
  fine_tuned_model.summary()
  H = load_history(finetuning_exp)
```

```
Capa input_2 congelada…
Capa block1_conv1 congelada…
Capa block1_conv2 congelada…
Capa block1_pool congelada…
Capa block2_conv1 congelada…
Capa block2_conv2 congelada…
Capa block2_pool congelada…
Capa block3_conv1 congelada…
Capa block3_conv2 congelada…
Capa block3_conv3 congelada…
Capa block3_pool congelada…
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 2, 2, 512)         14714688

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 128)               262272

 dense_1 (Dense)             (None, 15)                1935

=================================================================
Total params: 14978895 (57.14 MB)
Trainable params: 13243407 (50.52 MB)
Non-trainable params: 1735488 (6.62 MB)
_________________________________________________________________
[INFO]: Entrenando VGG16 desde block4_conv1 +  Top Model …
Epoch 1/50
58/58 [==============================] - 35s 292ms/step - loss: 8.7502 -
accuracy: 0.0921 - val_loss: 2.4305 - val_accuracy: 0.1643
Epoch 2/50
58/58 [==============================] - 16s 230ms/step - loss: 2.1153 -
accuracy: 0.2324 - val_loss: 1.7692 - val_accuracy: 0.3343
Epoch 3/50
58/58 [==============================] - 14s 246ms/step - loss: 1.5507 -
accuracy: 0.4124 - val_loss: 1.3756 - val_accuracy: 0.5183
Epoch 4/50
58/58 [==============================] - 13s 229ms/step - loss: 1.0694 -
accuracy: 0.6275 - val_loss: 0.7758 - val_accuracy: 0.7400
Epoch 5/50
58/58 [==============================] - 13s 230ms/step - loss: 0.6212 -
accuracy: 0.7897 - val_loss: 0.4567 - val_accuracy: 0.8447
Epoch 6/50
58/58 [==============================] - 13s 232ms/step - loss: 0.4380 -
accuracy: 0.8516 - val_loss: 0.3445 - val_accuracy: 0.8943
Epoch 7/50
58/58 [==============================] - 15s 260ms/step - loss: 0.2661 -
accuracy: 0.9133 - val_loss: 0.4044 - val_accuracy: 0.8743
Epoch 8/50
58/58 [==============================] - 15s 253ms/step - loss: 0.2056 -
accuracy: 0.9343 - val_loss: 0.1565 - val_accuracy: 0.9507
Epoch 9/50
58/58 [==============================] - 15s 253ms/step - loss: 0.1202 -
accuracy: 0.9632 - val_loss: 0.1313 - val_accuracy: 0.9637
Epoch 10/50
```

```
58/58 [==============================] - 14s 240ms/step - loss: 0.1078 -
accuracy: 0.9648 - val_loss: 0.1649 - val_accuracy: 0.9500
Epoch 11/50
58/58 [==============================] - 15s 260ms/step - loss: 0.0562 -
accuracy: 0.9827 - val_loss: 0.0892 - val_accuracy: 0.9743
Epoch 12/50
58/58 [==============================] - 15s 261ms/step - loss: 0.0555 -
accuracy: 0.9830 - val_loss: 0.0949 - val_accuracy: 0.9707
Epoch 13/50
58/58 [==============================] - 14s 242ms/step - loss: 0.0413 -
accuracy: 0.9870 - val_loss: 0.0805 - val_accuracy: 0.9760
Epoch 14/50
58/58 [==============================] - 14s 240ms/step - loss: 0.0518 -
accuracy: 0.9835 - val_loss: 0.0683 - val_accuracy: 0.9833
Epoch 15/50
58/58 [==============================] - 14s 236ms/step - loss: 0.0249 -
accuracy: 0.9917 - val_loss: 0.0921 - val_accuracy: 0.9760
Epoch 16/50
58/58 [==============================] - 15s 255ms/step - loss: 0.1351 -
accuracy: 0.9589 - val_loss: 0.2352 - val_accuracy: 0.9330
Epoch 17/50
58/58 [==============================] - 14s 241ms/step - loss: 0.0742 -
accuracy: 0.9763 - val_loss: 0.0647 - val_accuracy: 0.9833
Epoch 18/50
58/58 [==============================] - 14s 234ms/step - loss: 0.0337 -
accuracy: 0.9895 - val_loss: 0.0881 - val_accuracy: 0.9790
Epoch 19/50
58/58 [==============================] - 14s 236ms/step - loss: 0.0170 -
accuracy: 0.9948 - val_loss: 0.0730 - val_accuracy: 0.9833
Epoch 20/50
58/58 [==============================] - 15s 254ms/step - loss: 0.0283 -
accuracy: 0.9918 - val_loss: 0.0852 - val_accuracy: 0.9810
Epoch 21/50
58/58 [==============================] - 14s 239ms/step - loss: 0.0708 -
accuracy: 0.9820 - val_loss: 0.2112 - val_accuracy: 0.9427
Epoch 22/50
58/58 [==============================] - 15s 258ms/step - loss: 0.0856 -
accuracy: 0.9753 - val_loss: 0.0797 - val_accuracy: 0.9780
Epoch 23/50
58/58 [==============================] - 14s 235ms/step - loss: 0.0546 -
accuracy: 0.9847 - val_loss: 0.1011 - val_accuracy: 0.9697
Epoch 24/50
58/58 [==============================] - 15s 252ms/step - loss: 0.0271 -
accuracy: 0.9915 - val_loss: 0.0732 - val_accuracy: 0.9820
Epoch 25/50
58/58 [==============================] - 14s 235ms/step - loss: 0.0319 -
accuracy: 0.9910 - val_loss: 0.0734 - val_accuracy: 0.9803
Epoch 26/50
```

```
58/58 [==============================] - 14s 236ms/step - loss: 0.0119 -
accuracy: 0.9959 - val_loss: 0.0605 - val_accuracy: 0.9863
Epoch 27/50
58/58 [==============================] - 15s 262ms/step - loss: 0.0030 -
accuracy: 0.9991 - val_loss: 0.0508 - val_accuracy: 0.9873
Epoch 28/50
58/58 [==============================] - 14s 236ms/step - loss: 3.8840e-04 -
accuracy: 1.0000 - val_loss: 0.0430 - val_accuracy: 0.9893
Epoch 29/50
58/58 [==============================] - 15s 255ms/step - loss: 6.1954e-04 -
accuracy: 0.9999 - val_loss: 0.0599 - val_accuracy: 0.9887
Epoch 30/50
58/58 [==============================] - 15s 259ms/step - loss: 3.5416e-04 -
accuracy: 0.9999 - val_loss: 0.0457 - val_accuracy: 0.9903
Epoch 31/50
58/58 [==============================] - 14s 239ms/step - loss: 1.4188e-04 -
accuracy: 0.9999 - val_loss: 0.0474 - val_accuracy: 0.9900
Saved model to disk
```

```
[ ]: visualize_learning_curve(H, lb = finetuning_exp)
```



```
[ ]: evaluate_model(fine_tuned_model, x_test_preprocessed, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 5s 102ms/step
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       200
           1       0.99      0.99      0.99       200
           2       1.00      1.00      1.00       200
           3       1.00      0.99      1.00       200
           4       0.99      1.00      1.00       200
           5       0.98      0.98      0.98       200
           6       1.00      0.97      0.98       200
           7       0.98      0.98      0.98       200
           8       1.00      0.98      0.99       200
           9       0.99      0.99      0.99       200
          10       1.00      1.00      1.00       200
          11       1.00      1.00      1.00       200
          12       0.98      0.98      0.98       200
          13       0.99      0.99      0.99       200
          14       0.96      0.99      0.98       200

    accuracy                           0.99      3000
   macro avg       0.99      0.99      0.99      3000
weighted avg       0.99      0.99      0.99      3000
```

COMENTARIO fine tuning parcial VGG16: se observa una convergencia más lenta que mediante la técnica *transfer leanrning*.

8.1 6.3 Fine Tuning Completo

La función **get_fine_tuned_model_full** definida a continuación recibe un base_model, junto con el tamaño de la capa densa entrenable del top_model y el numero de clases de salida conectando ambas redes para realizar la tarea de fine tuning. Dado que estamos en una tarea de fine tuning completo, se entrenarán los todos pesos del base_model a partir de la primera capa entrenable hasta el top_model.

[ ]:

```python
def get_fine_tuned_model_full(base_model, dense_size, num_clases):

  base_model.trainable = True

  fine_tuned_model = Sequential()
  fine_tuned_model.add(base_model)
  fine_tuned_model.add(layers.Flatten())
  fine_tuned_model.add(layers.Dense(dense_size, activation="relu"))
  fine_tuned_model.add(layers.Dense(num_clases, activation="softmax"))

  fine_tuned_model.summary()
  return fine_tuned_model
```

###**6.3.1. VGG16**

[ ]:
```
# Elegir modelo base
base = 'VGG16'
# Cargar modelo base y preprocesar datos acorde al modelo base
base_model, x_train_preprocessed, x_val_preprocessed, x_test_preprocessed = get_base_model(base, x_train, x_val, x_test)

finetuning_exp = base + "_FULL_" + str(batch_size) + "_" + str(epochs)
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_3 (InputLayer)        [(None, 75, 75, 3)]       0

 block1_conv1 (Conv2D)       (None, 75, 75, 64)        1792

 block1_conv2 (Conv2D)       (None, 75, 75, 64)        36928

 block1_pool (MaxPooling2D)  (None, 37, 37, 64)        0

 block2_conv1 (Conv2D)       (None, 37, 37, 128)       73856

 block2_conv2 (Conv2D)       (None, 37, 37, 128)       147584

 block2_pool (MaxPooling2D)  (None, 18, 18, 128)       0

 block3_conv1 (Conv2D)       (None, 18, 18, 256)       295168

 block3_conv2 (Conv2D)       (None, 18, 18, 256)       590080

 block3_conv3 (Conv2D)       (None, 18, 18, 256)       590080

 block3_pool (MaxPooling2D)  (None, 9, 9, 256)         0

 block4_conv1 (Conv2D)       (None, 9, 9, 512)         1180160

 block4_conv2 (Conv2D)       (None, 9, 9, 512)         2359808

 block4_conv3 (Conv2D)       (None, 9, 9, 512)         2359808

 block4_pool (MaxPooling2D)  (None, 4, 4, 512)         0

 block5_conv1 (Conv2D)       (None, 4, 4, 512)         2359808
 block5_conv2 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_conv3 (Conv2D)       (None, 4, 4, 512)         2359808

 block5_pool (MaxPooling2D)  (None, 2, 2, 512)         0

=================================================================
Total params: 14714688 (56.13 MB)
Trainable params: 14714688 (56.13 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:
```
# Entrenamos la red
if do_training == True:

# Cargar modelo base
  fine_tuned_model = get_fine_tuned_model_full(base_model, dense_size,␣
 ↪num_clases)

  fine_tuned_model.compile(optimizer=Adam(learning_rate=learning_rate),
                        loss="categorical_crossentropy",
                        metrics=["accuracy"])

  print("[INFO]: Entrenando " + base + " completo + Top Model...")

  H = fine_tuned_model.fit(x_train_preprocessed, y_train_ohe,
                         batch_size=batch_size,
                         epochs=epochs,
                         steps_per_epoch=x_train_preprocessed.shape[0] //␣
 ↪batch_size,
                         validation_data=(x_val_preprocessed, y_val_ohe),
                         callbacks=[early_stopping_cbck])

  save_trained_model(model = fine_tuned_model, history = H, model_name =␣
 ↪finetuning_exp)

else:
  print("[INFO]: Entrenando " + base + " completo + Top Model...")
  fine_tuned_model = load_keras_model(finetuning_exp)
  fine_tuned_model.summary()
  H = load_history(finetuning_exp)
```

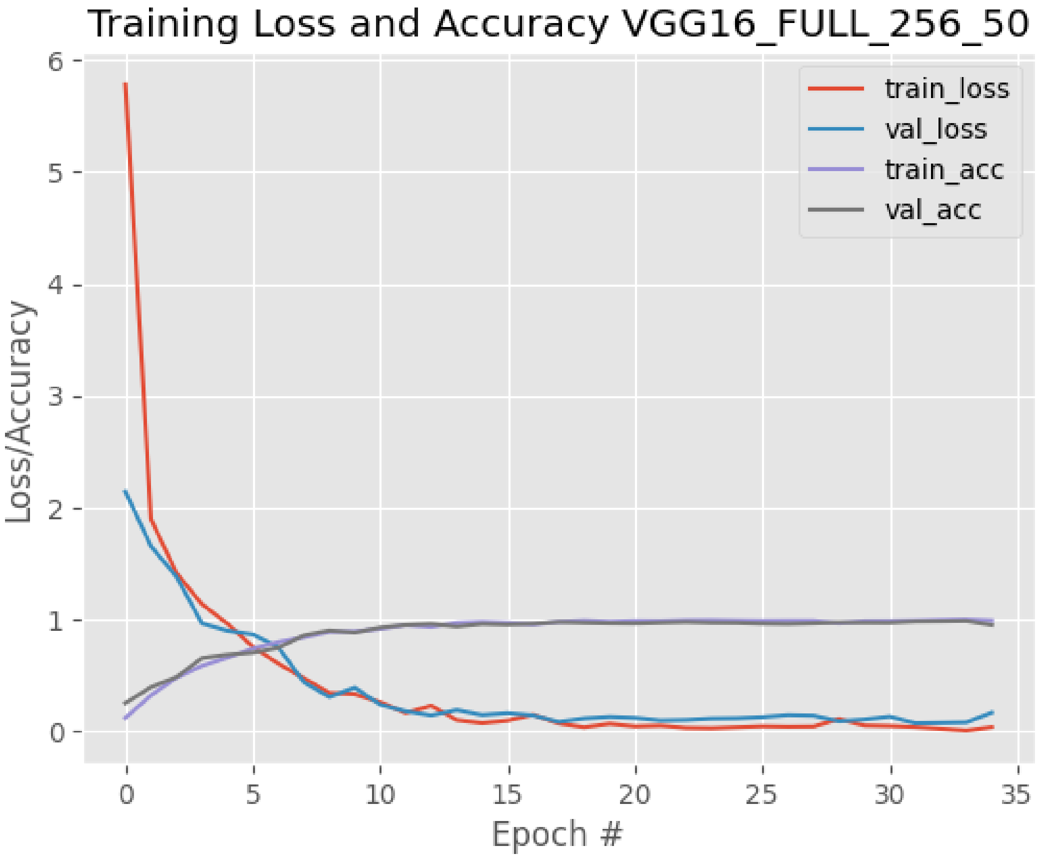
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 2, 2, 512)         14714688
 flatten_1 (Flatten)         (None, 2048)              0

 dense_2 (Dense)             (None, 128)               262272

 dense_3 (Dense)             (None, 15)                1935

=================================================================
Total params: 14978895 (57.14 MB)
Trainable params: 14978895 (57.14 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
[INFO]: Entrenando VGG16 completo + Top Model…
Epoch 1/50
58/58 [==============================] - 37s 456ms/step - loss: 5.7823 -
accuracy: 0.1195 - val_loss: 2.1403 - val_accuracy: 0.2517
Epoch 2/50
58/58 [==============================] - 28s 435ms/step - loss: 1.8964 -
accuracy: 0.3200 - val_loss: 1.6537 - val_accuracy: 0.3930
Epoch 3/50
58/58 [==============================] - 24s 421ms/step - loss: 1.4141 -
accuracy: 0.4776 - val_loss: 1.3816 - val_accuracy: 0.4827
Epoch 4/50
58/58 [==============================] - 25s 440ms/step - loss: 1.1352 -
accuracy: 0.5837 - val_loss: 0.9642 - val_accuracy: 0.6527
Epoch 5/50
58/58 [==============================] - 24s 419ms/step - loss: 0.9618 -
accuracy: 0.6567 - val_loss: 0.8981 - val_accuracy: 0.6837
Epoch 6/50
58/58 [==============================] - 26s 444ms/step - loss: 0.7539 -
accuracy: 0.7384 - val_loss: 0.8652 - val_accuracy: 0.7023
Epoch 7/50
58/58 [==============================] - 25s 439ms/step - loss: 0.6035 -
accuracy: 0.7971 - val_loss: 0.7504 - val_accuracy: 0.7460
Epoch 8/50
58/58 [==============================] - 25s 438ms/step - loss: 0.4731 -
accuracy: 0.8400 - val_loss: 0.4375 - val_accuracy: 0.8560
Epoch 9/50
58/58 [==============================] - 25s 440ms/step - loss: 0.3407 -
accuracy: 0.8874 - val_loss: 0.3057 - val_accuracy: 0.8993
Epoch 10/50
58/58 [==============================] - 25s 440ms/step - loss: 0.3309 -
accuracy: 0.8907 - val_loss: 0.3890 - val_accuracy: 0.8813
Epoch 11/50
58/58 [==============================] - 24s 420ms/step - loss: 0.2586 -
accuracy: 0.9147 - val_loss: 0.2364 - val_accuracy: 0.9263
Epoch 12/50
58/58 [==============================] - 25s 439ms/step - loss: 0.1628 -
```

```
accuracy: 0.9489 - val_loss: 0.1780 - val_accuracy: 0.9497
Epoch 13/50
58/58 [==============================] - 25s 439ms/step - loss: 0.2273 -
accuracy: 0.9322 - val_loss: 0.1409 - val_accuracy: 0.9590
Epoch 14/50
58/58 [==============================] - 26s 441ms/step - loss: 0.0997 -
accuracy: 0.9691 - val_loss: 0.1890 - val_accuracy: 0.9360
Epoch 15/50
58/58 [==============================] - 25s 440ms/step - loss: 0.0725 -
accuracy: 0.9773 - val_loss: 0.1447 - val_accuracy: 0.9593
Epoch 16/50
58/58 [==============================] - 25s 439ms/step - loss: 0.0953 -
accuracy: 0.9697 - val_loss: 0.1612 - val_accuracy: 0.9533
Epoch 17/50
58/58 [==============================] - 25s 440ms/step - loss: 0.1459 -
accuracy: 0.9543 - val_loss: 0.1405 - val_accuracy: 0.9617
Epoch 18/50
58/58 [==============================] - 24s 420ms/step - loss: 0.0707 -
accuracy: 0.9780 - val_loss: 0.0834 - val_accuracy: 0.9760
Epoch 19/50
58/58 [==============================] - 24s 421ms/step - loss: 0.0347 -
accuracy: 0.9888 - val_loss: 0.1138 - val_accuracy: 0.9700
Epoch 20/50
58/58 [==============================] - 25s 439ms/step - loss: 0.0666 -
accuracy: 0.9790 - val_loss: 0.1281 - val_accuracy: 0.9663
Epoch 21/50
58/58 [==============================] - 26s 446ms/step - loss: 0.0425 -
accuracy: 0.9861 - val_loss: 0.1190 - val_accuracy: 0.9643
Epoch 22/50
58/58 [==============================] - 24s 421ms/step - loss: 0.0493 -
accuracy: 0.9845 - val_loss: 0.0960 - val_accuracy: 0.9707
Epoch 23/50
58/58 [==============================] - 24s 422ms/step - loss: 0.0291 -
accuracy: 0.9913 - val_loss: 0.1004 - val_accuracy: 0.9757
Epoch 24/50
58/58 [==============================] - 24s 420ms/step - loss: 0.0262 -
accuracy: 0.9923 - val_loss: 0.1126 - val_accuracy: 0.9703
Epoch 25/50
58/58 [==============================] - 26s 441ms/step - loss: 0.0351 -
accuracy: 0.9890 - val_loss: 0.1154 - val_accuracy: 0.9693
Epoch 26/50
58/58 [==============================] - 24s 420ms/step - loss: 0.0432 -
accuracy: 0.9864 - val_loss: 0.1255 - val_accuracy: 0.9627
Epoch 27/50
58/58 [==============================] - 26s 442ms/step - loss: 0.0394 -
accuracy: 0.9876 - val_loss: 0.1439 - val_accuracy: 0.9600
Epoch 28/50
58/58 [==============================] - 24s 420ms/step - loss: 0.0410 -
```

```
accuracy: 0.9874 - val_loss: 0.1389 - val_accuracy: 0.9657
Epoch 29/50
58/58 [==============================] - 25s 440ms/step - loss: 0.1094 -
accuracy: 0.9655 - val_loss: 0.0893 - val_accuracy: 0.9733
Epoch 30/50
58/58 [==============================] - 24s 420ms/step - loss: 0.0513 -
accuracy: 0.9845 - val_loss: 0.1047 - val_accuracy: 0.9710
Epoch 31/50
58/58 [==============================] - 26s 441ms/step - loss: 0.0461 -
accuracy: 0.9849 - val_loss: 0.1303 - val_accuracy: 0.9703
Epoch 32/50
58/58 [==============================] - 25s 439ms/step - loss: 0.0366 -
accuracy: 0.9886 - val_loss: 0.0713 - val_accuracy: 0.9810
Epoch 33/50
58/58 [==============================] - 25s 430ms/step - loss: 0.0217 -
accuracy: 0.9944 - val_loss: 0.0748 - val_accuracy: 0.9827
Epoch 34/50
58/58 [==============================] - 26s 442ms/step - loss: 0.0053 -
accuracy: 0.9983 - val_loss: 0.0790 - val_accuracy: 0.9860
Epoch 35/50
58/58 [==============================] - 25s 440ms/step - loss: 0.0364 -
accuracy: 0.9896 - val_loss: 0.1649 - val_accuracy: 0.9500
Saved model to disk
```

[ ]:

```
visualize_learning_curve(H, lb = finetuning_exp)
```



```
[ ]: evaluate_model(fine_tuned_model, x_test_preprocessed, y_test)
```

```
[INFO]: Evaluando red neuronal…
24/24 [==============================] - 2s 61ms/step
              precision    recall  f1-score   support

           0       0.90      0.96      0.93       200
           1       0.96      0.95      0.95       200
           2       0.97      1.00      0.98       200
           3       0.94      1.00      0.97       200
           4       0.92      0.98      0.95       200
           5       0.97      0.95      0.96       200
           6       0.94      0.95      0.95       200
           7       0.88      0.96      0.92       200
           8       0.97      0.94      0.96       200
           9       0.95      0.79      0.86       200
          10       0.97      0.97      0.97       200
          11       1.00      1.00      1.00       200
          12       0.98      0.86      0.92       200
          13       0.97      0.96      0.96       200
          14       0.97      0.97      0.97       200

    accuracy                           0.95      3000
   macro avg       0.95      0.95      0.95      3000
weighted avg       0.95      0.95      0.95      3000
```

COMENTARIO fine tuning completo VGG16: se observa una convergencia más lenta que mediante la técnica *transfer leanrning* y *fine tuning* parcial.

9 7. Análisis de Resultados

En este proyecto se han desarrollado dos estrategias de clasificación de imágenes del dataset de vegetable-image-dataset.

En la estrategia 1 se han explorado arquitecturas basadas en perceptron multicapa (MLP) y redes neuronales convolucionales (CNN) combinadas con técnicas de regularización y data augmentation, obteniendo los siguientes resultados en términos de *precision, recall y f1-score* en el conjunto de test.

1. Perceptron multicapa

	precision	recall	f1-score
MLP_BASE	0.72	0.69	0.69
MLP_DROP	0.46	0.42	0.40
MLP_REG	0.53	0.51	0.51
MLP_AUG	0.75	0.72	0.71

2. Redes convolucionales

	precision	recall	f1-score
1_CNN	0.87	0.86	0.86
2_CNN	0.95	0.95	0.95
3_CNN	0.97	0.97	0.97
3_CNN_BN	0.98	0.98	0.98
3_CNN_BN_AUG	0.82	0.78	0.79

3. Transfer Learning con red pre-entrenada

	precision	recall	f1-score
VGG16	0.96	0.97	0.97

```
VGG16_AUG       0.97      0.97      0.97

XCEPTION        0.98      0.98      0.98

INCEPTIONV3     0.96      0.96      0.96

RESNET50        0.99      0.99      0.99
```

4. Fine Tuning parcial con red pre-entrenada

```
              precision  recall  f1-score

VGG16          0.99      0.99      0.99
```

5. Fine Tuning completo con red pre-entrenada

```
              precision  recall  f1-score

VGG16         0.95      0.95      0.95
```

Nomenclatura

```
*_REG: Regularización L1,L2
*_DROP: Dropout
*_BN: Batch normalization
*_AUG: Data augmentation (flow_from_directory)
```

En base a las pruebas realizadas, las arquitecturas de red convolucional *from scratch* (estrategia 1) se pueden obtener muy buenos resultados en esta tarea de clasificación, pero consideramos que la estrategia 2 en su variante de **transfer learning** (en especial **ResNet50**) es la más robusta en términos de curva de aprendizaje, resultados y eficiencia computacional.

Al centrarnos en la comparativa *transfer learning Vs fine tuning* sobre **VGG16**, se verifica que no existen problemas de translación de dominio o *domain shift.*

El uso de técnicas de regularización no conduce a mejores resultados en las redes CNN, no se se han detectado problemas de *overfitting* y su aplicación puede conducir rápidamente al *underfitting.* Sobre el MLP se aprecia una mejoría en la capacidad de generalización empleando la técnica *data augmentation.*

10 8. Conclusiones

Considerando los resultados obtenidos muestran que la Estrategia 2, que consiste en la utilización de redes neuronales preentrenadas para nuestra tarea específica, resulta ser más eficiente en términos de recursos. A diferencia de la Estrategia 1, la Estrategia 2 evita el esfuerzo de construir una red desde cero, logrando alcanzar resultados satisfactorios en un número significativamente menor de épocas (por ejemplo, 20 épocas en el caso de la ResNet) en comparación con una red entrenada completamente 'from scratch' de la CNN + Batch Normalization + Data Augmentation (34 épocas).

11 9. Exportación de Resultados

A continuación se exporta el notebook a PDF para su presentación

[3]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic
```

```
Reading package lists… Done
Building dependency tree… Done
Reading state information… Done
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base
  texlive-binaries texlive-latex-base texlive-latex-extra
  texlive-latex-recommended texlive-pictures tipa xfonts-encodings
  xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java
  poppler-utils ghostscript fonts-japanese-mincho | fonts-ipafont-mincho
  fonts-japanese-gothic | fonts-ipafont-gothic fonts-arphic-ukai
  fonts-arphic-uming fonts-nanum ri ruby-dev bundler debhelper gv
  | postscript-viewer perl-tk xpdf | pdf-viewer xzdec
  texlive-fonts-recommended-doc texlive-latex-base-doc python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl
  texlive-latex-extra-doc texlive-latex-recommended-doc texlive-luatex
  texlive-pstricks dot2tex prerex texlive-pictures-doc vprerex
  default-jre-headless tipa-doc
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 libapache-pom-java libcommons-logging-java
  libcommons-parent-java libfontbox-java libfontenc1 libgs9 libgs9-common
  libidn12 libijs-0.35 libjbig2dec0 libkpathsea6 libpdfbox-java libptexenc1
  libruby3.0 libsynctex2 libteckit0 libtexlua53 libtexluajit2 libwoff1
  libzzip-0-13 lmodern poppler-data preview-latex-style rake ruby
  ruby-net-telnet ruby-rubygems ruby-webrick ruby-xmlrpc ruby3.0
  rubygems-integration t1utils teckit tex-common tex-gyre texlive-base
  texlive-binaries texlive-fonts-recommended texlive-latex-base
  texlive-latex-extra texlive-latex-recommended texlive-pictures
  texlive-plain-generic texlive-xetex tipa xfonts-encodings xfonts-utils
0 upgraded, 54 newly installed, 0 to remove and 10 not upgraded.
Need to get 182 MB of archives.
```

```
After this operation, 571 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1build1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-lato all 2.0-2.1
[2,696 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 poppler-data all
0.4.11-1 [2,171 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-common all 6.17
[33.7 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-urw-base35 all
20200910-1 [6,367 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9-common
all 9.55.0~dfsg1-0ubuntu5.5 [752 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libidn12 amd64
1.38-4ubuntu1 [60.0 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy/main amd64 libijs-0.35 amd64
0.35-15build2 [16.5 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy/main amd64 libjbig2dec0 amd64
0.19-3build2 [64.7 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgs9 amd64
9.55.0~dfsg1-0ubuntu5.5 [5,030 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libkpathsea6
amd64 2021.20210626.59705-1ubuntu0.1 [60.3 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwoff1 amd64
1.0.2-1build4 [45.2 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/universe amd64 dvisvgm amd64
2.13.1-1 [1,221 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-lmodern all
2.004.5-6.1 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu jammy/main amd64 fonts-noto-mono all
20201225-1build1 [397 kB]
Get:16 http://archive.ubuntu.com/ubuntu jammy/universe amd64 fonts-texgyre all
20180621-3.1 [10.2 MB]
Get:17 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:18 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:19 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:20 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64
1:1.1.4-1build3 [14.7 kB]
Get:21 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libptexenc1
amd64 2021.20210626.59705-1ubuntu0.1 [39.1 kB]
Get:22 http://archive.ubuntu.com/ubuntu jammy/main amd64 rubygems-integration
all 1.18 [5,336 B]
Get:23 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby3.0 amd64
3.0.2-7ubuntu2.4 [50.1 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-rubygems all
```

```
3.3.5-2 [228 kB]
Get:25 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby amd64 1:3.0~exp1
[5,100 B]
Get:26 http://archive.ubuntu.com/ubuntu jammy/main amd64 rake all 13.0.6-2 [61.7
kB]
Get:27 http://archive.ubuntu.com/ubuntu jammy/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:28 http://archive.ubuntu.com/ubuntu jammy/universe amd64 ruby-webrick all
1.7.0-3 [51.8 kB]
Get:29 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ruby-xmlrpc all
0.3.2-1ubuntu0.1 [24.9 kB]
Get:30 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libruby3.0
amd64 3.0.2-7ubuntu2.4 [5,113 kB]
Get:31 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libsynctex2
amd64 2021.20210626.59705-1ubuntu0.1 [55.5 kB]
Get:32 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libteckit0 amd64
2.5.11+ds1-1 [421 kB]
Get:33 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexlua53
amd64 2021.20210626.59705-1ubuntu0.1 [120 kB]
Get:34 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libtexluajit2
amd64 2021.20210626.59705-1ubuntu0.1 [267 kB]
Get:35 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libzzip-0-13 amd64
0.13.72+dfsg.1-1.1 [27.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu2 [578 kB]
Get:37 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64
1:7.7+6build2 [94.6 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy/universe amd64 lmodern all
2.004.5-6.1 [9,471 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy/universe amd64 preview-latex-style
all 12.2-1ubuntu1 [185 kB]
Get:40 http://archive.ubuntu.com/ubuntu jammy/main amd64 t1utils amd64
1.41-4build2 [61.3 kB]
Get:41 http://archive.ubuntu.com/ubuntu jammy/universe amd64 teckit amd64
2.5.11+ds1-1 [699 kB]
Get:42 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tex-gyre all
20180621-3.1 [6,209 kB]
Get:43 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 texlive-
binaries amd64 2021.20210626.59705-1ubuntu0.1 [9,848 kB]
Get:44 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-base all
2021.20220204-1 [21.0 MB]
Get:45 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-fonts-
recommended all 2021.20220204-1 [4,972 kB]
Get:46 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-base
all 2021.20220204-1 [1,128 kB]
Get:47 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:48 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libpdfbox-java all
```

```
1:1.8.16-2 [5,199 kB]
Get:49 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-
recommended all 2021.20220204-1 [14.4 MB]
Get:50 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-pictures
all 2021.20220204-1 [8,720 kB]
Get:51 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-latex-extra
all 2021.20220204-1 [13.9 MB]
Get:52 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-plain-
generic all 2021.20220204-1 [27.5 MB]
Get:53 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tipa all 2:1.3-21
[2,967 kB]
Get:54 http://archive.ubuntu.com/ubuntu jammy/universe amd64 texlive-xetex all
2021.20220204-1 [12.4 MB]
Fetched 182 MB in 13s (14.1 MB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78,
<> line 54.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 120880 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1build1_all.deb
…
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2.1_all.deb …
Unpacking fonts-lato (2.0-2.1) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.11-1_all.deb …
Unpacking poppler-data (0.4.11-1) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.17_all.deb …
Unpacking tex-common (6.17) …
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack …/04-fonts-urw-base35_20200910-1_all.deb …
Unpacking fonts-urw-base35 (20200910-1) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/05-libgs9-common_9.55.0~dfsg1-0ubuntu5.5_all.deb …
Unpacking libgs9-common (9.55.0~dfsg1-0ubuntu5.5) …
Selecting previously unselected package libidn12:amd64.
Preparing to unpack …/06-libidn12_1.38-4ubuntu1_amd64.deb …
Unpacking libidn12:amd64 (1.38-4ubuntu1) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/07-libijs-0.35_0.35-15build2_amd64.deb …
```

```
Unpacking libijs-0.35:amd64 (0.35-15build2) …
Selecting previously unselected package libjbig2dec0:amd64.
Preparing to unpack …/08-libjbig2dec0_0.19-3build2_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.19-3build2) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/09-libgs9_9.55.0~dfsg1-0ubuntu5.5_amd64.deb …
Unpacking libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.5) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/10-libkpathsea6_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libwoff1:amd64.
Preparing to unpack …/11-libwoff1_1.0.2-1build4_amd64.deb …
Unpacking libwoff1:amd64 (1.0.2-1build4) …
Selecting previously unselected package dvisvgm.
Preparing to unpack …/12-dvisvgm_2.13.1-1_amd64.deb …
Unpacking dvisvgm (2.13.1-1) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/13-fonts-lmodern_2.004.5-6.1_all.deb …
Unpacking fonts-lmodern (2.004.5-6.1) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/14-fonts-noto-mono_20201225-1build1_all.deb …
Unpacking fonts-noto-mono (20201225-1build1) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/15-fonts-texgyre_20180621-3.1_all.deb …
Unpacking fonts-texgyre (20180621-3.1) …
Selecting previously unselected package libapache-pom-java.
Preparing to unpack …/16-libapache-pom-java_18-1_all.deb …
Unpacking libapache-pom-java (18-1) …
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack …/17-libcommons-parent-java_43-1_all.deb …
Unpacking libcommons-parent-java (43-1) …
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack …/18-libcommons-logging-java_1.2-2_all.deb …
Unpacking libcommons-logging-java (1.2-2) …
Selecting previously unselected package libfontenc1:amd64.
Preparing to unpack …/19-libfontenc1_1%3a1.1.4-1build3_amd64.deb …
Unpacking libfontenc1:amd64 (1:1.1.4-1build3) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/20-libptexenc1_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/21-rubygems-integration_1.18_all.deb …
Unpacking rubygems-integration (1.18) …
Selecting previously unselected package ruby3.0.
Preparing to unpack …/22-ruby3.0_3.0.2-7ubuntu2.4_amd64.deb …
Unpacking ruby3.0 (3.0.2-7ubuntu2.4) …
```

```
Selecting previously unselected package ruby-rubygems.
Preparing to unpack …/23-ruby-rubygems_3.3.5-2_all.deb …
Unpacking ruby-rubygems (3.3.5-2) …
Selecting previously unselected package ruby.
Preparing to unpack …/24-ruby_1%3a3.0~exp1_amd64.deb …
Unpacking ruby (1:3.0~exp1) …
Selecting previously unselected package rake.
Preparing to unpack …/25-rake_13.0.6-2_all.deb …
Unpacking rake (13.0.6-2) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/26-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-webrick.
Preparing to unpack …/27-ruby-webrick_1.7.0-3_all.deb …
Unpacking ruby-webrick (1.7.0-3) …
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack …/28-ruby-xmlrpc_0.3.2-1ubuntu0.1_all.deb …
Unpacking ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Selecting previously unselected package libruby3.0:amd64.
Preparing to unpack …/29-libruby3.0_3.0.2-7ubuntu2.4_amd64.deb …
Unpacking libruby3.0:amd64 (3.0.2-7ubuntu2.4) …
Selecting previously unselected package libsynctex2:amd64.
Preparing to unpack …/30-libsynctex2_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack …/31-libteckit0_2.5.11+ds1-1_amd64.deb …
Unpacking libteckit0:amd64 (2.5.11+ds1-1) …
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack …/32-libtexlua53_2021.20210626.59705-1ubuntu0.1_amd64.deb
…
Unpacking libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack
…/33-libtexluajit2_2021.20210626.59705-1ubuntu0.1_amd64.deb …
Unpacking libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/34-libzzip-0-13_0.13.72+dfsg.1-1.1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Selecting previously unselected package xfonts-encodings.
Preparing to unpack …/35-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb …
Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) …
Selecting previously unselected package xfonts-utils.
Preparing to unpack …/36-xfonts-utils_1%3a7.7+6build2_amd64.deb …
Unpacking xfonts-utils (1:7.7+6build2) …
Selecting previously unselected package lmodern.
Preparing to unpack …/37-lmodern_2.004.5-6.1_all.deb …
Unpacking lmodern (2.004.5-6.1) …
```

```
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/38-preview-latex-style_12.2-1ubuntu1_all.deb …
Unpacking preview-latex-style (12.2-1ubuntu1) …
Selecting previously unselected package t1utils.
Preparing to unpack …/39-t1utils_1.41-4build2_amd64.deb …
Unpacking t1utils (1.41-4build2) …
Selecting previously unselected package teckit.
Preparing to unpack …/40-teckit_2.5.11+ds1-1_amd64.deb …
Unpacking teckit (2.5.11+ds1-1) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/41-tex-gyre_20180621-3.1_all.deb …
Unpacking tex-gyre (20180621-3.1) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/42-texlive-
binaries_2021.20210626.59705-1ubuntu0.1_amd64.deb …
Unpacking texlive-binaries (2021.20210626.59705-1ubuntu0.1) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/43-texlive-base_2021.20220204-1_all.deb …
Unpacking texlive-base (2021.20220204-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/44-texlive-fonts-recommended_2021.20220204-1_all.deb …
Unpacking texlive-fonts-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/45-texlive-latex-base_2021.20220204-1_all.deb …
Unpacking texlive-latex-base (2021.20220204-1) …
Selecting previously unselected package libfontbox-java.
Preparing to unpack …/46-libfontbox-java_1%3a1.8.16-2_all.deb …
Unpacking libfontbox-java (1:1.8.16-2) …
Selecting previously unselected package libpdfbox-java.
Preparing to unpack …/47-libpdfbox-java_1%3a1.8.16-2_all.deb …
Unpacking libpdfbox-java (1:1.8.16-2) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/48-texlive-latex-recommended_2021.20220204-1_all.deb …
Unpacking texlive-latex-recommended (2021.20220204-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/49-texlive-pictures_2021.20220204-1_all.deb …
Unpacking texlive-pictures (2021.20220204-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/50-texlive-latex-extra_2021.20220204-1_all.deb …
Unpacking texlive-latex-extra (2021.20220204-1) …
Selecting previously unselected package texlive-plain-generic.
Preparing to unpack …/51-texlive-plain-generic_2021.20220204-1_all.deb …
Unpacking texlive-plain-generic (2021.20220204-1) …
Selecting previously unselected package tipa.
Preparing to unpack …/52-tipa_2%3a1.3-21_all.deb …
Unpacking tipa (2:1.3-21) …
Selecting previously unselected package texlive-xetex.
Preparing to unpack …/53-texlive-xetex_2021.20220204-1_all.deb …
```

```
Unpacking texlive-xetex (2021.20220204-1) …
Setting up fonts-lato (2.0-2.1) …
Setting up fonts-noto-mono (20201225-1build1) …
Setting up libwoff1:amd64 (1.0.2-1build4) …
Setting up libtexlua53:amd64 (2021.20210626.59705-1ubuntu0.1) …
Setting up libijs-0.35:amd64 (0.35-15build2) …
Setting up libtexluajit2:amd64 (2021.20210626.59705-1ubuntu0.1) …
Setting up libfontbox-java (1:1.8.16-2) …
Setting up rubygems-integration (1.18) …
Setting up libzzip-0-13:amd64 (0.13.72+dfsg.1-1.1) …
Setting up fonts-urw-base35 (20200910-1) …
Setting up poppler-data (0.4.11-1) …
Setting up tex-common (6.17) …
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line
78.)
debconf: falling back to frontend: Readline
update-language: texlive-base not installed and configured, doing nothing!
Setting up libfontenc1:amd64 (1:1.1.4-1build3) …
Setting up libjbig2dec0:amd64 (0.19-3build2) …
Setting up libteckit0:amd64 (2.5.11+ds1-1) …
Setting up libapache-pom-java (18-1) …
Setting up ruby-net-telnet (0.1.1-2) …
Setting up xfonts-encodings (1:1.0.5-0ubuntu2) …
Setting up t1utils (1.41-4build2) …
Setting up libidn12:amd64 (1.38-4ubuntu1) …
Setting up fonts-texgyre (20180621-3.1) …
Setting up libkpathsea6:amd64 (2021.20210626.59705-1ubuntu0.1) …
Setting up ruby-webrick (1.7.0-3) …
Setting up fonts-lmodern (2.004.5-6.1) …
Setting up fonts-droid-fallback (1:6.0.1r16-1.1build1) …
Setting up ruby-xmlrpc (0.3.2-1ubuntu0.1) …
Setting up libsynctex2:amd64 (2021.20210626.59705-1ubuntu0.1) …
Setting up libgs9-common (9.55.0~dfsg1-0ubuntu5.5) …
Setting up teckit (2.5.11+ds1-1) …
Setting up libpdfbox-java (1:1.8.16-2) …
Setting up libgs9:amd64 (9.55.0~dfsg1-0ubuntu5.5) …
Setting up preview-latex-style (12.2-1ubuntu1) …
Setting up libcommons-parent-java (43-1) …
Setting up dvisvgm (2.13.1-1) …
Setting up libcommons-logging-java (1.2-2) …
Setting up xfonts-utils (1:7.7+6build2) …
Setting up libptexenc1:amd64 (2021.20210626.59705-1ubuntu0.1) …
Setting up texlive-binaries (2021.20210626.59705-1ubuntu0.1) …
update-alternatives: using /usr/bin/xdvi-xaw to provide /usr/bin/xdvi.bin
(xdvi.bin) in auto mode
update-alternatives: using /usr/bin/bibtex.original to provide /usr/bin/bibtex
```

```
(bibtex) in auto mode
Setting up lmodern (2.004.5-6.1) …
Setting up texlive-base (2021.20220204-1) …
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
/usr/bin/ucfr
mktexlsr: Updating /var/lib/texmf/ls-R-TEXLIVEDIST…
mktexlsr: Updating /var/lib/texmf/ls-R-TEXMFMAIN…
mktexlsr: Updating /var/lib/texmf/ls-R…
mktexlsr: Done.
tl-paper: setting paper size for dvips to a4:
/var/lib/texmf/dvips/config/config-paper.ps
tl-paper: setting paper size for dvipdfmx to a4:
/var/lib/texmf/dvipdfmx/dvipdfmx-paper.cfg
tl-paper: setting paper size for xdvi to a4: /var/lib/texmf/xdvi/XDvi-paper
tl-paper: setting paper size for pdftex to a4: /var/lib/texmf/tex/generic/tex-
ini-files/pdftexconfig.tex
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line
78.)
debconf: falling back to frontend: Readline
Setting up tex-gyre (20180621-3.1) …
Setting up texlive-plain-generic (2021.20220204-1) …
Setting up texlive-latex-base (2021.20220204-1) …
Setting up texlive-latex-recommended (2021.20220204-1) …
Setting up texlive-pictures (2021.20220204-1) …
Setting up texlive-fonts-recommended (2021.20220204-1) …
Setting up tipa (2:1.3-21) …
Setting up texlive-latex-extra (2021.20220204-1) …
Setting up texlive-xetex (2021.20220204-1) …
Setting up rake (13.0.6-2) …
Setting up libruby3.0:amd64 (3.0.2-7ubuntu2.4) …
Setting up ruby3.0 (3.0.2-7ubuntu2.4) …
Setting up ruby (1:3.0~exp1) …
Setting up ruby-rubygems (3.3.5-2) …
Processing triggers for man-db (2.10.2-1) …
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) …
Processing triggers for libc-bin (2.35-0ubuntu3.4) …
/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link
```

```
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic
link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

Processing triggers for tex-common (6.17) …
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line
78.)
debconf: falling back to frontend: Readline
Running updmap-sys. This may take some time… done.
Running mktexlsr /var/lib/texmf … done.
Building format(s) --all.
        This may take some time… done.
```

[ ]:

```
from google.colab import drive
drive.mount('/content/drive')

!jupyter nbconvert --to pdf '/content/drive/MyDrive/
 ↪07MIAR_Proyecto_Programacion/07MIAR_Proyecto_Programacion.ipynb'
```

```
[NbConvertApp] CRITICAL | Bad config encountered during initialization: Error
loading argument NbConvertApp.export_format=['pdf', 'pdf'], export_format only
accepts one value, got 2: ['pdf', 'pdf']
```

[ ]: