



Universidad Autónoma de San Luis Potosí

Facultad de ingeniería  
Inteligencia Artificial Aplicada

Práctica: 8

Nombre Práctica: Transfer Learning y Data Augmentation

Nombre del Alumno: Manuel Ramírez Galván



FACULTAD DE  
INGENIERÍA

Fecha: 26/03/2025

## Procedimiento

- 8.1.- Se entrenará una red neuronal para reconocer 4 señas distintas realizadas con las manos utilizando un conjunto de datos limitado.
- 8.2.- Recolectar al menos 80 imágenes de cada señal de la mano para el entrenamiento de la red y 20 para la validación.
- 8.3.- Abra el Notebook “transfer\_learning” y siga los pasos indicados para entrenar el modelo de red neuronal.
- 8.4.- Una vez entrenado el modelo pruebe el modelo con imágenes nuevas utilizando el método “predict” del modelo.

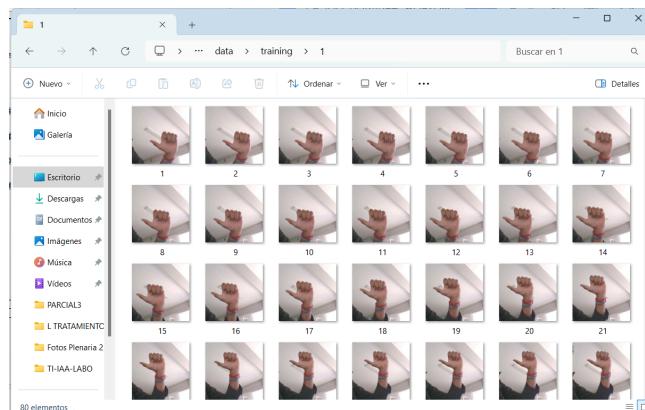


Imagen 1.- Imágenes para entrenamiento

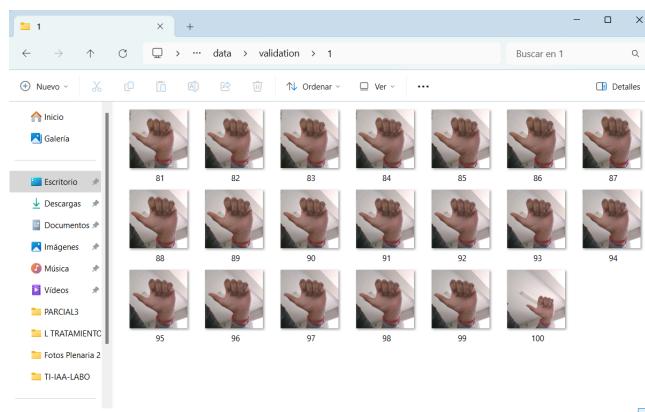


Imagen 2.- Imágenes para validación

Cada carpeta tendrá las imágenes correspondientes a esa sección.

```
from keras.utils import image_dataset_from_directory

train_data = image_dataset_from_directory(
    directory="data/training", # Ruta de la carpeta de entrenamiento
    labels="inferred", # Dejaremos que Keras infiera las etiquetas
    label_mode="categorical", # Utilizaremos etiquetas categóricas
    batch_size=32, # Tamaño del batch
    image_size=(224, 224), # Tamaño de las imágenes,
    color_mode="rgb" # Utilizaremos imágenes en escala de grises
)

#TODO: Crear el conjunto de validación
# Crear conjunto de validación
val_data = image_dataset_from_directory(
    directory="data/validation", # Ruta de la carpeta de validación
    labels="inferred", # Dejamos que Keras infiera las etiquetas
    label_mode="categorical", # Usamos etiquetas categóricas
    batch_size=32, # Tamaño del batch
    image_size=(224, 224), # Tamaño de las imágenes
    color_mode="rgb" # Imágenes en color (RGB)
)

```

✓ 0.2s

```
Found 320 files belonging to 4 classes.
Found 80 files belonging to 4 classes.
```

Imagen 3.- Utilizar imágenes de entrenamiento y validación

```
#Obtenemos las clases para saber el orden de las etiquetas
class_names = train_data.class_names
print(class_names)
```

✓ 0.0s

```
['1', '2', '3', '4']
```

Imagen 4.- Nombre de las etiquetas

```
from keras.layers import Input, Rescaling

#TODO Definir capa de entrada
input_layer = Input(shape=(224, 224, 3)) # Capa de entrada

#TODO Definir capa de normalización y conexión
x = Rescaling(1./255)(input_layer) # Normalizar las imágenes
```

✓ 0.0s

Imagen 5.- Definir Capa de Entrada y Normalizar

```
#TODO importar capas de Data Augmentation
from keras.layers import RandomFlip, RandomRotation, RandomZoom

#TODO Definir capas de Data Augmentation y concatenarlas

x = RandomFlip(mode="horizontal")(x) # Voltear las imágenes aleatoriamente
x = RandomRotation(factor=0.2)(x) # Rotar las imágenes aleatoriamente
x = RandomZoom(height_factor = 0.2, width_factor = 0.2)(x) # Aumentar el zoom aleatoriamente

✓ 0.0s
```

Imagen 6.- Agregar Capas de Preprocesamiento

```
from keras.applications import VGG16      Import "keras.applications" could not be resolved

#TODO Cargar modelo VGG16 sin las capas densas
base_model = VGG16(include_top=False, weights="imagenet", input_shape=(224, 224, 3))

#TODO Congelar las primeras 5 capas
for layer in base_model.layers[:5]:
    layer.trainable = False

✓ 0.3s
```

Imagen 7.- Importar Modelo y Congelar Capas

```
#TODO Conectar modelo previo a VG16

x = base_model(x) ValueError: Input 0 is incompatible with layer flatten_1: expected 4D array, got 1D array instead

from keras.layers import Flatten

# Conectar el modelo VGG16 a nuevas capas
x = Flatten()(x) # Aplanar la salida

✓ 0.0s
```

Imagen 8.- Conectar Modelo

```
#TODO Definir capa densa de salida

from keras.layers import Dense      Import "keras.layers"

x = Dense(units=128, activation="relu")(x) # Capa densa
output_layer = Dense(units=4, activation="softmax")(x)

#TODO Definir modelo

from keras.models import Model      Import "keras.models"

model = Model(inputs=input_layer, outputs=output_layer)
model.summary()

✓ 0.0s
```

Imagen 9.- Agregar Capas de Salida

```
● #TODO Compilar modelo
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
✓ 0.0s
```

Imagen 10.- Compilar Modelo

```
#TODO Entrenar modelo
model.fit(
    train_data, # Datos de entrenamiento
    epochs=10, # Número de épocas, ajusta según sea necesario
    validation_data=val_data, # Datos de validación
)
✓ 51m 58.9s
```

Imagen 11.-Entrenar Modelo

```
#TODO Guardar modelo
model.save("model.h5")
✓ 1.5s
```

Imagen 12.-Guardar Modelo

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

# Cargar el modelo previamente entrenado
model = load_model('model.h5')

# Inicializar la cámara
cap = cv2.VideoCapture(0)

while True:
    # Capturar frame por frame
    ret, frame = cap.read()

    if not ret:
        break

    # Convertir el frame a formato que el modelo pueda entender
    # Cambia el tamaño del frame a 224x224 como espera nuestro modelo
    frame_resized = cv2.resize(frame, (224, 224))
    # Convertir el frame de BGR a RGB (OpenCV captura en formato BGR)
    frame_rgb = cv2.cvtColor(frame_resized, cv2.COLOR_BGR2RGB)
    # Normalizar los pixeles
    frame_normalized = frame_rgb / 255.0
    # Expandir dimensiones para adaptarse al input del modelo
    frame_expanded = np.expand_dims(frame_normalized, axis=0)

    # Realizar predicción

    class_names = ['A', 'B', 'C', 'D']
    prediction = model.predict(frame_expanded)
    predicted = class_names[np.argmax(prediction)]

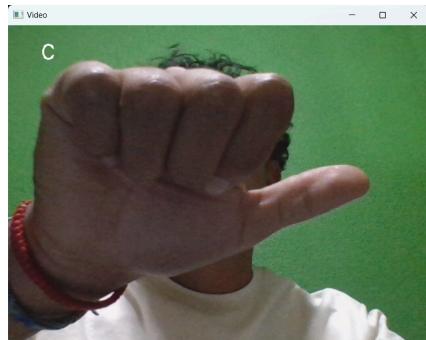
    # Mostrar la clase predicha en el video
    cv2.putText(frame, predicted, (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255),
2)
    # Mostrar el frame resultante
    cv2.imshow('Video', frame)

    # Romper el bucle con la tecla 'q'
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    # Cuando todo está hecho, liberar la captura
cap.release()
cv2.destroyAllWindows()
```

Imagen 12.- Código para Probar el Modelo

## Resultados



*Imagen 13.- Identificar “A”*



*Imagen 14.- Identificar “B”*



*Imagen 15.- Identificar “C”*



*Imagen 16.- Identificar “D”*

## Comprepción

### 1. ¿En qué consiste Data Augmentation?

Consiste en aplicar transformaciones a los datos para crear versiones variadas de los mismos ejemplos, permitiendo entrenar modelos más robustos, precisos y resistentes a cambios en el entorno real.

### 2. Mencione 4 capas diferentes para implementar Data Augmentation en Keras y explique su funcionamiento

- RandomFlip, realiza un volteo aleatorio (horizontal y/o vertical) de la imagen.
- RandomRotation, rota la imagen aleatoriamente dentro de un rango especificado (en radianes o grados).
- RandomZoom, hace acercamientos o alejamientos aleatorios (zoom in o zoom out) sobre la imagen.
- RandomContrast, ajusta el contraste de la imagen aleatoriamente, haciéndola más clara u oscura en relación a su media.

### 3. ¿Qué es un modelo pre-entrenado?

Es una red neuronal que ya ha sido entrenada previamente en un conjunto de datos grande y general (como ImageNet, con millones de imágenes y miles de clases).

Ya ha aprendido a extraer características útiles (bordes, formas, texturas, etc.) y se puede reutilizar o adaptar para resolver otros problemas relacionados.

### 4. ¿Qué es Transfer Learning?

Es una técnica de aprendizaje automático donde un modelo previamente entrenado en una tarea se reutiliza parcial o totalmente para resolver otra tarea diferente pero relacionada, ahorrando tiempo, recursos y mejorando resultados.

En lugar de entrenar un modelo desde cero, se aprovechan los conocimientos ya aprendidos (como detectar bordes, formas, texturas) y se ajustan para un nuevo conjunto de datos o problema específico.

## **5. Mencione las ventajas de utilizar Transfer Learning y Fine-Tuning**

Ventajas de Transfer Learning

1. **Reduce el tiempo de entrenamiento**, No se necesita entrenar toda la red desde cero. Solo se entrena una parte del modelo (las últimas capas).
2. **Requiere menos recursos computacionales**, se aprovechan los pesos ya entrenados de modelos grandes, evitando largos entrenamientos con GPU.
3. **Mejora el rendimiento con pocos datos**, muy útil cuando se tiene un conjunto de datos pequeño o limitado.
4. **Aprovecha conocimiento generalizado**, el modelo ya ha aprendido características útiles como bordes, texturas y formas comunes.
5. **Fácil de implementar**, frameworks como TensorFlow y PyTorch permiten importar modelos preentrenados con pocas líneas de código.

Ventajas de Fine-Tuning

1. **Ajusta el modelo a una tarea específica**, al descongelar algunas capas del modelo preentrenado, se adapta mejor a los nuevos datos.
2. **Mejora la precisión**, al actualizar pesos en capas profundas, el modelo aprende detalles finos del nuevo conjunto de datos.
3. **Permite aprender características especializadas**, ideal cuando la nueva tarea es similar a la tarea original del modelo preentrenado.
4. **Combina lo mejor de ambos mundos**, reutiliza lo aprendido y se adapta al problema haciendo un modelo más robusto.

## **6. ¿En qué situaciones es mejor utilizar Transfer Learning y en cuáles Fine-Tuning?**

Transfer Learning cuando se necesitan resultados rápidos, se tienen pocos datos o recursos limitados.

Fine-Tuning es cuando se tienen más datos, tareas muy similares al modelo original y se busca mayor precisión.

## **Conclusiones**

El Transfer Learning funciona tomando un modelo ya enternado y usarlo para un nuevo modelo, el modelo cargado tiene información muy general del problema y solo se modifican las últimas capas con detalles más finos o con características más definidas para el reconocimiento.

Al usar el modelo en el código de prueba no funciona correctamente, ya que siempre marca la letra "C" independiente de la letra mostrada, y esto se ve en la precisión y en la perdida del modelo, donde son 24.19% y 138.62% respectivamente, lo que hace que el modelo no este funcionando correctamente y así el programa no detecte bien las señas.