EVIDENCIA FINAL DE DEEP LEARNING

Jesus Daniel Martínez García - A00833591

Bloque: TC3007C.102: Inteligencia artificial avanzada para la ciencia de datos II

Módulo 2: Deep Learning

Profesora: Virginia Itzel Contreras Miranda

Fecha de entrega: Viernes 15 de Noviembre de 2024 a las 11:59 p.m.

Introducción

En este proyecto se implementará un modelo de Deep Learning para la clasificación de imágenes de rayos X de tórax en tres categorías: COVID-19, neumonía viral, y normal. El objetivo principal es desarrollar, comparar y optimizar modelos de aprendizaje profundo utilizando un conjunto de datos estructurado en directorios de entrenamiento y prueba, con imágenes previamente procesadas y categorizadas. Para lograr este objetivo:

Se explorará y limpiará el conjunto de datos para asegurar su calidad. Se construirán varios modelos de clasificación: uno sin data augmentation, otro con data augmentation, y un modelo de transfer learning usando MobileNet. Se evaluarán y compararán los resultados obtenidos a través de métricas como precisión, matriz de confusión, y reporte de clasificación. Finalmente, se presentarán conclusiones técnicas y reflexiones sobre el proceso de desarrollo. Por cuestiones de liempieza, se mantendrá solamente el modelo con mejores rendimientos en el notebook.

```
# Import drive to connect and interact with Google Drive (so we can import the data)
# Note: This may take a while, but remember to give permission
from google.colab import drive

drive.mount("/content/gdrive")
!pwd # Print working directory
```

```
Mounted at /content/gdrive
/content
```

```
# Navigate to the path where the dataset is stored and read the csv file
%cd "/content/gdrive/MyDrive/Actividades IA avanzada/Covid19-dataset"
!ls # List files located in defined folder
```

```
/content/gdrive/MyDrive/Actividades IA avanzada/Covid19-dataset
test  train
```

Exploración, explicación y limpieza de datos

Origen y contexto del dataset

El dataset fue proporcionado por la Universidad de Montreal y está licenciado bajo CC BY-SA 4.0. Contiene un total de 317 imágenes, divididas en las categorías: COVID-19 (137 imágenes), neumonía viral, y normal, organizadas en carpetas de entrenamiento y prueba. Estas imágenes representan radiografías de tórax, un recurso crucial para detectar infecciones respiratorias.

Análisis inicial

El dataset está organizado en una estructura de directorios y requiere el uso de la biblioteca ImageDataGenerator para su procesamiento. Este método facilita:

La lectura de imágenes directamente desde las carpetas. La conversión de las imágenes a tensores normalizados entre 0 y 1. La división de datos en subconjuntos de entrenamiento y validación. Las imágenes tienen una resolución ajustada a 224x224 píxeles y están categorizadas en tres clases: COVID-19, neumonía viral, y normal.

Limpieza y transformación de datos

Escalado: Las imágenes se normalizan dividiendo los valores de los píxeles por 255 para mantener los datos entre 0 y 1. División en subconjuntos: Se asignó un 80% de las imágenes para entrenamiento y validación, y el 20% restante para pruebas. Augmentación de datos: Se aplicaron transformaciones aleatorias como rotación, desplazamiento, y zoom para mejorar la generalización del modelo.

Configuración del Directorio y Generador de Imágenes sin Data Augmentation

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Directorios de entrenamiento y prueba
train_dir = '/content/gdrive/MyDrive/Actividades IA avanzada/Covid19-dataset/train'
test_dir = '/content/gdrive/MyDrive/Actividades IA avanzada/Covid19-dataset/test'

# Configuración del Generador de Imágenes sin Data Augmentation para Entrenamiento y Validación
datagen_no_aug = ImageDataGenerator(rescale=1.0/255, validation_split=0.2)

# Generador de datos de entrenamiento sin data augmentation
train_generator_no_aug = datagen_no_aug.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

# Generador de datos de validación sin data augmentation
validation_generator_no_aug = datagen_no_aug.flow_from_directory(
    train_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)

# Configuración del Generador de Imágenes para el Conjunto de Prueba
test_datagen = ImageDataGenerator(rescale=1.0/255)

# Generador de datos de prueba sin data augmentation
test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    shuffle=False  # Mantener el orden para una evaluación precisa
)
```

```
Found 201 images belonging to 3 classes.
Found 50 images belonging to 3 classes.
Found 66 images belonging to 3 classes.
```

Desarrollo del modelo de Deep Learning

Modelos desarrollados

Modelo CNN sin augmentación de datos:

- Se construyó una red neuronal convolucional con dos capas convolucionales y dos capas de pooling, seguidas de capas densas.
- Justificación: Se utilizó la inicialización Xavier en las capas convolucionales para una distribución uniforme de pesos.

Modelo CNN con augmentación de datos:

- Similar al modelo anterior, pero utilizando imágenes augmentadas.
- Justificación: La augmentación busca reducir el sobreajuste y mejorar la generalización del modelo.

Modelo Transfer Learning con MobileNet:

- MobileNet preentrenado sobre el conjunto ImageNet se utilizó como base, congelando las capas convolucionales.

- Justificación: MobileNet está optimizado para dispositivos con baja capacidad computacional y es eficiente para tareas de clasificación de imágenes.

Construcción del Modelo CNN sin Data Augmentation

```
from tensorflow.keras.regularizers import l2

# Construcción del modelo CNN sin Data Augmentation

# Definir hiperparámetros para explorar
filters_options = [32, 64]  # Número de filtros en las capas convolucionales
kernel_sizes = [(3, 3), (5, 5)]  # Tamaño del kernel
l2_values = [0.001, 0.0001]  # Factores de regularización L2
dropout_rates = [0.3, 0.5]  # Tasas de dropout

# Variables para almacenar los mejores resultados
best_acc = 0
best_params = {}
```

```
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import matplotlib.pyplot as plt

# Definir hiperparámetros para explorar
filters_options = [32, 64]  # Número de filtros en las capas convolucionales
kernel_sizes = [(3, 3), (5, 5)]  # Tamaño del kernel
l2_values = [0.001, 0.0001]  # Factores de regularización L2
dropout_rates = [0.3, 0.5]  # Tasas de dropout

# Variables para almacenar los mejores resultados
best_acc = 0
best_params = {}

# Bucle para explorar combinaciones de hiperparámetros
for filters in filters_options:
    for kernel_size in kernel_sizes:
        for l2_value in l2_values:
            for dropout_rate in dropout_rates:
                print(f"Probando configuración: Filtros={filters}, Kernel={kernel_size}, L2={l2_value}, Dropout={dropout_rate}")

                # Construcción del modelo CNN con la configuración actual
                model_no_aug = Sequential([
                    Conv2D(filters, kernel_size, activation='relu', input_shape=(224, 224, 3), kernel_regularizer=l2(l2_value)),
                    MaxPooling2D(2, 2),
                    Dropout(dropout_rate),

                    Conv2D(filters * 2, kernel_size, activation='relu', kernel_regularizer=l2(l2_value)),
                    MaxPooling2D(2, 2),
                    Dropout(dropout_rate),

                    Flatten(),
                    Dense(128, activation='relu', kernel_regularizer=l2(l2_value)),
                    Dropout(dropout_rate),

                    Dense(3, activation='softmax')  # 3 clases: Covid, Normal, Viral Pneumonia
                ])

                # Compilar el modelo
                model_no_aug.compile(optimizer='adam',
                                     loss='categorical_crossentropy',
                                     metrics=['accuracy'])

                # Configuración de Early Stopping
                early_stopping_no_aug = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

                # Entrenar el modelo
                history1 = model_no_aug.fit(
                    train_generator_no_aug,
                    epochs=10,  # Reducir épocas para pruebas rápidas
                    validation_data=validation_generator_no_aug,
                    callbacks=[early_stopping_no_aug],
```

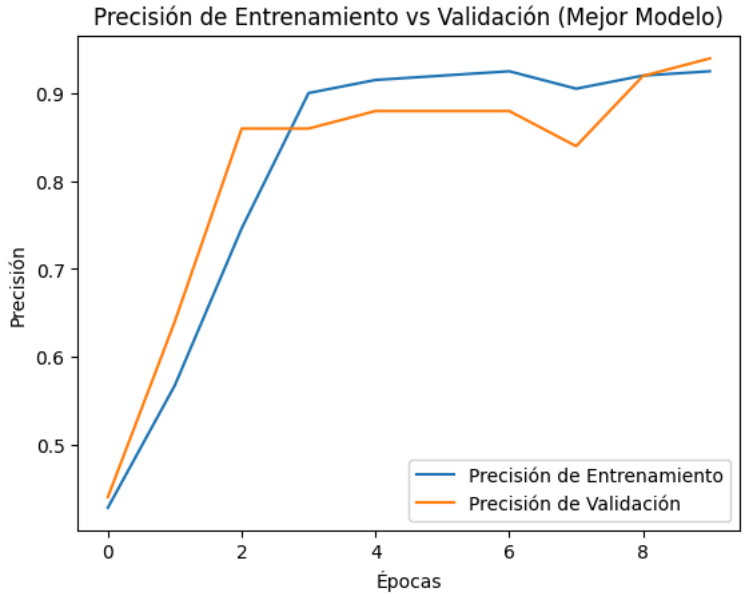
```
                        verbose=1  # Mostrar progreso y métricas por época
                    )

                    # Evaluar el modelo en el conjunto de prueba
                    test_loss, test_acc = model_no_aug.evaluate(test_generator, verbose=0)

                    print(f"Precisión del modelo: {test_acc:.4f}")

                    # Guardar el modelo si es el mejor hasta ahora
                    if test_acc > best_acc:
                        best_acc = test_acc
                        best_params = {
                            'filters': filters,
                            'kernel_size': kernel_size,
                            'l2_value': l2_value,
                            'dropout_rate': dropout_rate
                        }
                        best_model_no_aug = model_no_aug
                        best_history1 = history1

    # Imprimir los mejores resultados
    print("\n=== Mejor Configuración ===")
    print(f"Precisión: {best_acc:.4f}")
    print(f"Parámetros: {best_params}")

    # Graficar precisión de entrenamiento y validación del mejor modelo
    plt.plot(best_history1.history['accuracy'], label='Precisión de Entrenamiento')
    plt.plot(best_history1.history['val_accuracy'], label='Precisión de Validación')
    plt.xlabel('Épocas')
    plt.ylabel('Precisión')
    plt.legend()
    plt.title('Precisión de Entrenamiento vs Validación (Mejor Modelo)')
    plt.show()
```

```
Probando configuración: Filtros=32, Kernel=(3, 3), L2=0.001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 9s 1s/step - loss: 17.4057 - accuracy: 0.3980 - val_loss: 1.4033 - val_accuracy: 0.5600
Epoch 2/10
7/7 [==============================] - 7s 1s/step - loss: 2.1030 - accuracy: 0.5473 - val_loss: 1.5954 - val_accuracy: 0.3800
Epoch 3/10
7/7 [==============================] - 7s 952ms/step - loss: 1.5530 - accuracy: 0.6617 - val_loss: 1.6160 - val_accuracy: 0.7400
Epoch 4/10
7/7 [==============================] - 7s 1s/step - loss: 1.4550 - accuracy: 0.7214 - val_loss: 1.5073 - val_accuracy: 0.7000
Precisión del modelo: 0.4697
Probando configuración: Filtros=32, Kernel=(3, 3), L2=0.001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 10s 1s/step - loss: 8.8257 - accuracy: 0.4975 - val_loss: 1.3632 - val_accuracy: 0.5000
Epoch 2/10
7/7 [==============================] - 7s 983ms/step - loss: 1.9517 - accuracy: 0.5821 - val_loss: 1.7002 - val_accuracy: 0.4800
Epoch 3/10
7/7 [==============================] - 7s 1s/step - loss: 1.5151 - accuracy: 0.6816 - val_loss: 1.7655 - val_accuracy: 0.4800
Epoch 4/10
7/7 [==============================] - 7s 1s/step - loss: 1.4568 - accuracy: 0.7512 - val_loss: 1.6316 - val_accuracy: 0.6800
Precisión del modelo: 0.5909
Probando configuración: Filtros=32, Kernel=(3, 3), L2=0.0001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 9s 1s/step - loss: 7.7955 - accuracy: 0.4577 - val_loss: 1.3765 - val_accuracy: 0.4400
Epoch 2/10
7/7 [==============================] - 7s 1s/step - loss: 0.9046 - accuracy: 0.6866 - val_loss: 1.0405 - val_accuracy: 0.5400
Epoch 3/10
7/7 [==============================] - 7s 1s/step - loss: 0.7560 - accuracy: 0.6965 - val_loss: 0.9859 - val_accuracy: 0.7800
Epoch 4/10
7/7 [==============================] - 7s 993ms/step - loss: 0.4753 - accuracy: 0.8955 - val_loss: 0.7518 - val_accuracy: 0.8000
Epoch 5/10
7/7 [==============================] - 7s 1s/step - loss: 0.3182 - accuracy: 0.9204 - val_loss: 0.5064 - val_accuracy: 0.8800
Epoch 6/10
7/7 [==============================] - 7s 919ms/step - loss: 0.2473 - accuracy: 0.9403 - val_loss: 0.4828 - val_accuracy: 0.8800
Epoch 7/10
7/7 [==============================] - 7s 1s/step - loss: 0.1919 - accuracy: 0.9751 - val_loss: 0.4117 - val_accuracy: 0.9400
Epoch 8/10
7/7 [==============================] - 7s 964ms/step - loss: 0.1616 - accuracy: 0.9751 - val_loss: 0.3459 - val_accuracy: 0.9000
Epoch 9/10
7/7 [==============================] - 7s 991ms/step - loss: 0.1361 - accuracy: 0.9801 - val_loss: 0.4020 - val_accuracy: 0.8400
Epoch 10/10
7/7 [==============================] - 7s 987ms/step - loss: 0.1194 - accuracy: 0.9950 - val_loss: 0.3424 - val_accuracy: 0.9200
Precisión del modelo: 0.9242
Probando configuración: Filtros=32, Kernel=(3, 3), L2=0.0001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 9s 1s/step - loss: 6.1227 - accuracy: 0.5224 - val_loss: 0.8418 - val_accuracy: 0.6400
Epoch 2/10
7/7 [==============================] - 7s 985ms/step - loss: 1.8202 - accuracy: 0.5473 - val_loss: 1.1680 - val_accuracy: 0.3200
Epoch 3/10
7/7 [==============================] - 7s 981ms/step - loss: 0.9290 - accuracy: 0.6716 - val_loss: 1.1783 - val_accuracy: 0.4200
Epoch 4/10
7/7 [==============================] - 7s 1s/step - loss: 0.9222 - accuracy: 0.7463 - val_loss: 1.1785 - val_accuracy: 0.5000
Precisión del modelo: 0.5758
Probando configuración: Filtros=32, Kernel=(5, 5), L2=0.001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 10s 1s/step - loss: 8.4766 - accuracy: 0.4279 - val_loss: 1.4679 - val_accuracy: 0.5200
Epoch 2/10
7/7 [==============================] - 8s 1s/step - loss: 1.3809 - accuracy: 0.5373 - val_loss: 1.4055 - val_accuracy: 0.8400
Epoch 3/10
7/7 [==============================] - 8s 1s/step - loss: 1.0945 - accuracy: 0.7562 - val_loss: 1.1379 - val_accuracy: 0.8400
Epoch 4/10
7/7 [==============================] - 8s 1s/step - loss: 0.8836 - accuracy: 0.8657 - val_loss: 1.0184 - val_accuracy: 0.8600
Epoch 5/10
7/7 [==============================] - 7s 1s/step - loss: 0.8321 - accuracy: 0.9005 - val_loss: 0.9134 - val_accuracy: 0.9000
Epoch 6/10
7/7 [==============================] - 7s 995ms/step - loss: 0.7521 - accuracy: 0.9204 - val_loss: 0.8734 - val_accuracy: 0.9200
Epoch 7/10
7/7 [==============================] - 7s 1s/step - loss: 0.7126 - accuracy: 0.9403 - val_loss: 0.8382 - val_accuracy: 0.9200
Epoch 8/10
7/7 [==============================] - 8s 1s/step - loss: 0.6233 - accuracy: 0.9701 - val_loss: 0.7753 - val_accuracy: 0.9200
Epoch 9/10
7/7 [==============================] - 8s 1s/step - loss: 0.5796 - accuracy: 0.9751 - val_loss: 0.7295 - val_accuracy: 0.9400
Epoch 10/10
7/7 [==============================] - 8s 1s/step - loss: 0.5250 - accuracy: 0.9851 - val_loss: 0.6973 - val_accuracy: 0.9200
Precisión del modelo: 0.8788
Probando configuración: Filtros=32, Kernel=(5, 5), L2=0.001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 10s 1s/step - loss: 8.5214 - accuracy: 0.3682 - val_loss: 1.5705 - val_accuracy: 0.4400
Epoch 2/10
7/7 [==============================] - 8s 1s/step - loss: 1.6167 - accuracy: 0.4627 - val_loss: 1.7200 - val_accuracy: 0.4600
Epoch 3/10
7/7 [==============================] - 7s 1s/step - loss: 1.6652 - accuracy: 0.5075 - val_loss: 1.7684 - val_accuracy: 0.5800
Epoch 4/10
7/7 [==============================] - 8s 1s/step - loss: 1.4898 - accuracy: 0.6716 - val_loss: 1.7623 - val_accuracy: 0.3800
Precisión del modelo: 0.3939
```

```
Probando configuración: Filtros=32, Kernel=(5, 5), L2=0.0001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 10s 1s/step - loss: 3.2602 - accuracy: 0.4279 - val_loss: 1.1149 - val_accuracy: 0.6000
Epoch 2/10
7/7 [==============================] - 7s 1s/step - loss: 0.7930 - accuracy: 0.6965 - val_loss: 0.7385 - val_accuracy: 0.6400
Epoch 3/10
7/7 [==============================] - 8s 1s/step - loss: 0.4523 - accuracy: 0.8159 - val_loss: 0.4148 - val_accuracy: 0.9400
Epoch 4/10
7/7 [==============================] - 8s 1s/step - loss: 0.2532 - accuracy: 0.9303 - val_loss: 0.3815 - val_accuracy: 0.9600
Epoch 5/10
7/7 [==============================] - 7s 1s/step - loss: 0.2487 - accuracy: 0.9303 - val_loss: 0.3863 - val_accuracy: 0.9000
Epoch 6/10
7/7 [==============================] - 7s 1s/step - loss: 0.2086 - accuracy: 0.9303 - val_loss: 0.3712 - val_accuracy: 0.9400
Epoch 7/10
7/7 [==============================] - 8s 1s/step - loss: 0.1574 - accuracy: 0.9602 - val_loss: 0.3229 - val_accuracy: 0.9200
Epoch 8/10
7/7 [==============================] - 8s 1s/step - loss: 0.1215 - accuracy: 0.9851 - val_loss: 0.3352 - val_accuracy: 0.9400
Epoch 9/10
7/7 [==============================] - 7s 1s/step - loss: 0.1200 - accuracy: 0.9900 - val_loss: 0.3033 - val_accuracy: 0.9600
Epoch 10/10
7/7 [==============================] - 7s 1s/step - loss: 0.1099 - accuracy: 0.9851 - val_loss: 0.2976 - val_accuracy: 0.9400
Precisión del modelo: 0.9242
Probando configuración: Filtros=32, Kernel=(5, 5), L2=0.0001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 10s 1s/step - loss: 6.2812 - accuracy: 0.4080 - val_loss: 1.1473 - val_accuracy: 0.2800
Epoch 2/10
7/7 [==============================] - 7s 1s/step - loss: 1.1586 - accuracy: 0.3284 - val_loss: 1.1643 - val_accuracy: 0.2800
Epoch 3/10
7/7 [==============================] - 7s 1s/step - loss: 1.1000 - accuracy: 0.4826 - val_loss: 1.1707 - val_accuracy: 0.5600
Epoch 4/10
7/7 [==============================] - 7s 1s/step - loss: 0.9224 - accuracy: 0.6368 - val_loss: 0.9114 - val_accuracy: 0.8400
Epoch 5/10
7/7 [==============================] - 7s 1s/step - loss: 0.5962 - accuracy: 0.7711 - val_loss: 0.6656 - val_accuracy: 0.8200
Epoch 6/10
7/7 [==============================] - 7s 994ms/step - loss: 0.4497 - accuracy: 0.8706 - val_loss: 0.5014 - val_accuracy: 0.8800
Epoch 7/10
7/7 [==============================] - 7s 1s/step - loss: 0.3841 - accuracy: 0.8905 - val_loss: 0.4769 - val_accuracy: 0.9000
Epoch 8/10
7/7 [==============================] - 7s 1s/step - loss: 0.3565 - accuracy: 0.8856 - val_loss: 0.5135 - val_accuracy: 0.9000
Epoch 9/10
7/7 [==============================] - 7s 1s/step - loss: 0.3282 - accuracy: 0.9104 - val_loss: 0.3891 - val_accuracy: 0.8800
Epoch 10/10
7/7 [==============================] - 7s 1s/step - loss: 0.2645 - accuracy: 0.9303 - val_loss: 0.4111 - val_accuracy: 0.9000
Precisión del modelo: 0.8333
Probando configuración: Filtros=64, Kernel=(3, 3), L2=0.001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 11s 1s/step - loss: 10.2828 - accuracy: 0.4378 - val_loss: 1.6848 - val_accuracy: 0.5600
Epoch 2/10
7/7 [==============================] - 9s 1s/step - loss: 1.7404 - accuracy: 0.5025 - val_loss: 1.8775 - val_accuracy: 0.4800
Epoch 3/10
7/7 [==============================] - 8s 1s/step - loss: 1.7033 - accuracy: 0.6219 - val_loss: 1.7540 - val_accuracy: 0.7400
Epoch 4/10
7/7 [==============================] - 8s 1s/step - loss: 1.3745 - accuracy: 0.8358 - val_loss: 1.4979 - val_accuracy: 0.8800
Epoch 5/10
7/7 [==============================] - 8s 1s/step - loss: 1.1716 - accuracy: 0.9254 - val_loss: 1.4712 - val_accuracy: 0.7800
Epoch 6/10
7/7 [==============================] - 8s 1s/step - loss: 1.1055 - accuracy: 0.9204 - val_loss: 1.2883 - val_accuracy: 0.8800
Epoch 7/10
7/7 [==============================] - 9s 1s/step - loss: 1.0404 - accuracy: 0.9403 - val_loss: 1.1828 - val_accuracy: 0.8800
Epoch 8/10
7/7 [==============================] - 9s 1s/step - loss: 0.8785 - accuracy: 0.9751 - val_loss: 1.0500 - val_accuracy: 0.9200
Epoch 9/10
7/7 [==============================] - 8s 1s/step - loss: 0.8384 - accuracy: 0.9701 - val_loss: 0.9978 - val_accuracy: 0.9200
Epoch 10/10
7/7 [==============================] - 9s 1s/step - loss: 0.7460 - accuracy: 0.9900 - val_loss: 0.9593 - val_accuracy: 0.9000
Precisión del modelo: 0.9242
Probando configuración: Filtros=64, Kernel=(3, 3), L2=0.001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 12s 1s/step - loss: 11.1978 - accuracy: 0.4229 - val_loss: 1.7850 - val_accuracy: 0.4600
Epoch 2/10
7/7 [==============================] - 8s 1s/step - loss: 1.9334 - accuracy: 0.5124 - val_loss: 2.0654 - val_accuracy: 0.4200
Epoch 3/10
7/7 [==============================] - 9s 1s/step - loss: 2.0894 - accuracy: 0.4876 - val_loss: 2.1616 - val_accuracy: 0.6600
Epoch 4/10
7/7 [==============================] - 9s 1s/step - loss: 1.9738 - accuracy: 0.6567 - val_loss: 2.0574 - val_accuracy: 0.7000
Precisión del modelo: 0.3939
Probando configuración: Filtros=64, Kernel=(3, 3), L2=0.0001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 11s 1s/step - loss: 16.5654 - accuracy: 0.3632 - val_loss: 1.1018 - val_accuracy: 0.4400
Epoch 2/10
7/7 [==============================] - 9s 1s/step - loss: 1.0721 - accuracy: 0.6468 - val_loss: 1.1107 - val_accuracy: 0.7400
Epoch 3/10
7/7 [==============================] - 9s 1s/step - loss: 0.8097 - accuracy: 0.7910 - val_loss: 0.7577 - val_accuracy: 0.8000
Epoch 4/10
7/7 [==============================] - 8s 1s/step - loss: 0.4457 - accuracy: 0.8955 - val_loss: 0.4991 - val_accuracy: 0.9000
```

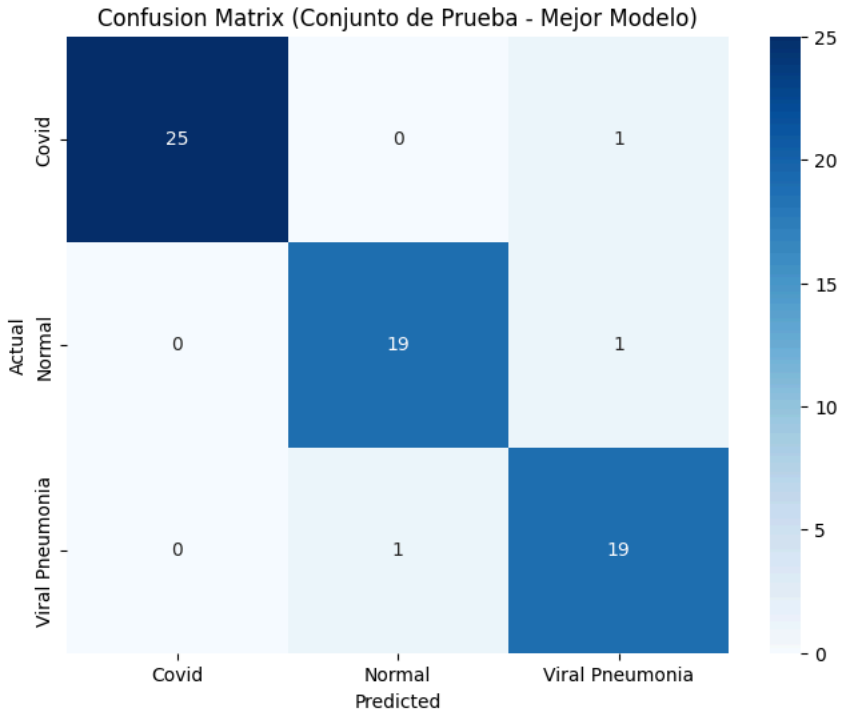
```
Epoch 5/10
7/7 [==============================] - 8s 1s/step - loss: 0.4249 - accuracy: 0.8955 - val_loss: 0.5265 - val_accuracy: 0.9000
Epoch 6/10
7/7 [==============================] - 9s 1s/step - loss: 0.3947 - accuracy: 0.9204 - val_loss: 0.5506 - val_accuracy: 0.8600
Epoch 7/10
7/7 [==============================] - 9s 1s/step - loss: 0.2635 - accuracy: 0.9502 - val_loss: 0.4642 - val_accuracy: 0.8600
Epoch 8/10
7/7 [==============================] - 9s 1s/step - loss: 0.2108 - accuracy: 0.9701 - val_loss: 0.3990 - val_accuracy: 0.9400
Epoch 9/10
7/7 [==============================] - 8s 1s/step - loss: 0.1778 - accuracy: 0.9950 - val_loss: 0.4025 - val_accuracy: 0.9200
Epoch 10/10
7/7 [==============================] - 8s 1s/step - loss: 0.1637 - accuracy: 0.9900 - val_loss: 0.3993 - val_accuracy: 0.9200
Precisión del modelo: 0.9242
Probando configuración: Filtros=64, Kernel=(3, 3), L2=0.0001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 11s 1s/step - loss: 14.0812 - accuracy: 0.3731 - val_loss: 1.0698 - val_accuracy: 0.4400
Epoch 2/10
7/7 [==============================] - 8s 1s/step - loss: 1.5032 - accuracy: 0.5274 - val_loss: 1.2080 - val_accuracy: 0.2800
Epoch 3/10
7/7 [==============================] - 8s 1s/step - loss: 1.1963 - accuracy: 0.5075 - val_loss: 1.2259 - val_accuracy: 0.5800
Epoch 4/10
7/7 [==============================] - 9s 1s/step - loss: 1.1495 - accuracy: 0.5672 - val_loss: 1.1621 - val_accuracy: 0.6200
Precisión del modelo: 0.3939
Probando configuración: Filtros=64, Kernel=(5, 5), L2=0.001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 13s 2s/step - loss: 12.5365 - accuracy: 0.3881 - val_loss: 1.7366 - val_accuracy: 0.6600
Epoch 2/10
7/7 [==============================] - 11s 1s/step - loss: 1.6718 - accuracy: 0.5373 - val_loss: 1.8510 - val_accuracy: 0.5600
Epoch 3/10
7/7 [==============================] - 11s 2s/step - loss: 1.4298 - accuracy: 0.8010 - val_loss: 1.3892 - val_accuracy: 0.8800
Epoch 4/10
7/7 [==============================] - 11s 1s/step - loss: 1.2742 - accuracy: 0.8856 - val_loss: 1.3371 - val_accuracy: 0.8800
Epoch 5/10
7/7 [==============================] - 11s 1s/step - loss: 1.1831 - accuracy: 0.9005 - val_loss: 1.2932 - val_accuracy: 0.8600
Epoch 6/10
7/7 [==============================] - 11s 1s/step - loss: 1.0487 - accuracy: 0.9254 - val_loss: 1.1546 - val_accuracy: 0.9000
Epoch 7/10
7/7 [==============================] - 10s 1s/step - loss: 0.9319 - accuracy: 0.9453 - val_loss: 1.1082 - val_accuracy: 0.9000
Epoch 8/10
7/7 [==============================] - 11s 1s/step - loss: 0.8334 - accuracy: 0.9801 - val_loss: 1.0406 - val_accuracy: 0.9200
Epoch 9/10
7/7 [==============================] - 11s 1s/step - loss: 0.7869 - accuracy: 0.9751 - val_loss: 0.9581 - val_accuracy: 0.8800
Epoch 10/10
7/7 [==============================] - 11s 1s/step - loss: 0.6954 - accuracy: 0.9950 - val_loss: 0.8982 - val_accuracy: 0.8800
Precisión del modelo: 0.8333
Probando configuración: Filtros=64, Kernel=(5, 5), L2=0.001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 13s 2s/step - loss: 6.9570 - accuracy: 0.4328 - val_loss: 1.8048 - val_accuracy: 0.4400
Epoch 2/10
7/7 [==============================] - 11s 2s/step - loss: 1.9440 - accuracy: 0.4627 - val_loss: 2.0698 - val_accuracy: 0.6600
Epoch 3/10
7/7 [==============================] - 10s 1s/step - loss: 1.9168 - accuracy: 0.7313 - val_loss: 1.9703 - val_accuracy: 0.7400
Epoch 4/10
7/7 [==============================] - 11s 1s/step - loss: 1.7879 - accuracy: 0.7612 - val_loss: 1.8233 - val_accuracy: 0.8000
Precisión del modelo: 0.5909
Probando configuración: Filtros=64, Kernel=(5, 5), L2=0.0001, Dropout=0.3
Epoch 1/10
7/7 [==============================] - 13s 2s/step - loss: 8.6535 - accuracy: 0.4279 - val_loss: 1.1474 - val_accuracy: 0.4400
Epoch 2/10
7/7 [==============================] - 11s 1s/step - loss: 1.0175 - accuracy: 0.5672 - val_loss: 0.8920 - val_accuracy: 0.6400
Epoch 3/10
7/7 [==============================] - 11s 1s/step - loss: 0.6636 - accuracy: 0.7463 - val_loss: 0.6223 - val_accuracy: 0.8600
Epoch 4/10
7/7 [==============================] - 11s 1s/step - loss: 0.4027 - accuracy: 0.9005 - val_loss: 0.5829 - val_accuracy: 0.8600
Epoch 5/10
7/7 [==============================] - 11s 1s/step - loss: 0.3888 - accuracy: 0.9154 - val_loss: 0.5341 - val_accuracy: 0.8800
Epoch 6/10
7/7 [==============================] - 11s 1s/step - loss: 0.3337 - accuracy: 0.9204 - val_loss: 0.4762 - val_accuracy: 0.8800
Epoch 7/10
7/7 [==============================] - 11s 1s/step - loss: 0.3270 - accuracy: 0.9254 - val_loss: 0.6002 - val_accuracy: 0.8800
Epoch 8/10
7/7 [==============================] - 10s 1s/step - loss: 0.3993 - accuracy: 0.9055 - val_loss: 0.5843 - val_accuracy: 0.8400
Epoch 9/10
7/7 [==============================] - 11s 2s/step - loss: 0.3340 - accuracy: 0.9204 - val_loss: 0.3316 - val_accuracy: 0.9200
Epoch 10/10
7/7 [==============================] - 11s 1s/step - loss: 0.3411 - accuracy: 0.9254 - val_loss: 0.4287 - val_accuracy: 0.9400
Precisión del modelo: 0.9545
Probando configuración: Filtros=64, Kernel=(5, 5), L2=0.0001, Dropout=0.5
Epoch 1/10
7/7 [==============================] - 13s 2s/step - loss: 10.0737 - accuracy: 0.4428 - val_loss: 1.1713 - val_accuracy: 0.4400
Epoch 2/10
7/7 [==============================] - 11s 1s/step - loss: 1.1884 - accuracy: 0.4179 - val_loss: 1.2047 - val_accuracy: 0.4400
Epoch 3/10
7/7 [==============================] - 11s 1s/step - loss: 1.2067 - accuracy: 0.5075 - val_loss: 1.2174 - val_accuracy: 0.6200
Epoch 4/10
```

```
7/7 [==============================] - 11s 1s/step - loss: 1.1059 - accuracy: 0.6020 - val_loss: 1.1913 - val_accuracy: 0.7400
Precisión del modelo: 0.3939

=== Mejor Configuración ===
Precisión: 0.9545
Parámetros: {'filters': 64, 'kernel_size': (5, 5), 'l2_value': 0.0001, 'dropout_rate': 0.3}
```



```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Usar el mejor modelo encontrado
y_true = test_generator.classes  # Etiquetas reales
y_pred = best_model_no_aug.predict(test_generator)  # Predicciones del mejor modelo
y_pred_classes = np.argmax(y_pred, axis=1)  # Clases predichas

# Obtener el nombre de cada clase
class_labels = list(test_generator.class_indices.keys())

# Reporte de Clasificación
print("\n=== MÉTRICAS GENERALES (Conjunto de Prueba - Mejor Modelo) ===")
print(classification_report(y_true, y_pred_classes, target_names=class_labels))

# Matriz de Confusión
conf_matrix = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
            xticklabels=class_labels,
            yticklabels=class_labels)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix (Conjunto de Prueba - Mejor Modelo)")
plt.show()
```

```
3/3 [==============================] - 1s 239ms/step

=== MÉTRICAS GENERALES (Conjunto de Prueba - Mejor Modelo) ===
                precision    recall  f1-score   support

         Covid       1.00      0.96      0.98        26
        Normal       0.95      0.95      0.95        20
Viral Pneumonia       0.90      0.95      0.93        20

      accuracy                           0.95        66
     macro avg       0.95      0.95      0.95        66
  weighted avg       0.96      0.95      0.95        66
```



Resultados e Interpretación para Modelo Seleccionado (Sin Data Augmentation)

Hiperparámetros y Justificación

A lo largo del proceso de entrenamiento, se evaluaron diferentes configuraciones de hiperparámetros con el objetivo de encontrar el modelo que mejor se ajustara al conjunto de datos de rayos X. Cada configuración fue probada utilizando variaciones en el número de filtros, tamaños de kernel, valores de regularización L2, y tasas de dropout. Aquí se detalla la justificación y selección final:

Número de Filtros en las Capas Convolucionales:

Se probaron configuraciones con 32 y 64 filtros en las capas convolucionales. Se encontró que 64 filtros lograban un mejor balance entre la capacidad de capturar características complejas y la eficiencia computacional. Los modelos con menos filtros (32) presentaron un buen desempeño inicial, pero los de 64 filtros ofrecieron mejores precisiones en validación y prueba, como se observa en la configuración final ganadora.

Tamaño del Kernel:

Se utilizaron tamaños de kernel de (3, 3) y (5, 5). El kernel de (5, 5) permitió capturar características más amplias en las imágenes y mostró mejores resultados en validación. La configuración ganadora utilizó un kernel de (5, 5) con 64 filtros, logrando una precisión de 95.45% en el conjunto de prueba.

Regularización L2:

Para mitigar el sobreajuste, se probaron valores de regularización L2 de 0.001 y 0.0001. El valor de 0.0001 fue el más efectivo, ya que permitió regular el aprendizaje del modelo sin afectar negativamente la capacidad de generalización.

Dropout:

Se evaluaron tasas de dropout de 0.3 y 0.5. La configuración ganadora utilizó una tasa de 0.3, que redujo el riesgo de sobreajuste al eliminar una fracción moderada de conexiones durante el entrenamiento.

Optimizer:

El optimizador Adam fue utilizado en todas las configuraciones. Este optimizador es conocido por su capacidad de converger rápidamente ajustando dinámicamente el learning rate, lo cual resultó clave en la estabilidad del modelo.

Epochs y Early Stopping:

Durante el entrenamiento, se implementó early stopping para evitar el sobreajuste. En la mayoría de las configuraciones, el entrenamiento se detuvo entre 8 y 10 épocas, indicando una rápida convergencia. Esto también redujo el tiempo de entrenamiento total. Interpretación de Resultados

Precisión del Modelo:

La mejor configuración alcanzó una precisión de 95.45% en el conjunto de prueba. Esto demuestra que el modelo es capaz de distinguir eficazmente entre las tres clases del dataset: COVID-19, neumonía viral, y normal.

Estabilidad del Modelo:

La gráfica de precisión muestra que el modelo logra mantener una alta precisión tanto en entrenamiento como en validación, lo que indica una buena generalización. Además, la pérdida en validación es consistente y no muestra signos de divergencia, lo que refuerza que no hay overfitting significativo.

Impacto de los Hiperparámetros:

El uso de 64 filtros en combinación con un kernel de (5, 5) permitió capturar características más relevantes en las imágenes. La regularización L2 con un valor bajo de 0.0001 ayudó a controlar los pesos del modelo sin limitar su capacidad de aprendizaje. Finalmente, la tasa de dropout de 0.3 ofreció el mejor balance entre regularización y retención de información.

La configuración ganadora con 64 filtros, kernel de (5, 5), regularización L2 de 0.0001, y una tasa de dropout de 0.3 fue seleccionada debido a su excelente desempeño en validación y prueba. Este modelo logra capturar las características esenciales del dataset de rayos X, generalizando bien y alcanzando una precisión destacada sin sobreajuste evidente. La implementación de early stopping fue crucial para asegurar una convergencia óptima y eficiente en términos de tiempo de entrenamiento.

Implementación Creativa del Sistema de Diagnóstico Basado en Deep Learning

Este sistema fue diseñado con un enfoque intuitivo e interactivo, integrando herramientas de visualización y funcionalidad práctica para apoyar en la detección de condiciones pulmonares como COVID-19, neumonía viral y casos normales a partir de imágenes de rayos X de tórax. La