Practical session 13

Biomedical Data Science

Lucas Fayolle & Jose Valero

Libraries

In [18]:

```
import numpy as np
import random
from scipy import ndimage
import pandas as pd
import matplotlib.pyplot as plt
import os
import json

import gc
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

from tensorflow.keras.models import load_model
```

Data

Data loading

The first step is to load the preprocessed data stored in a file named `processed_data.npz`. This file contains the training and validation datasets, including both the features (`x_train`, `x_val`) and their corresponding labels (`y_train`, `y_val`). These datasets have been split into training and validation sets in a 70-30 ratio to ensure a robust evaluation of the model's performance.

The `processed_data.npz` file was created from a set of CT scans, which were processed using the following steps:

1. **Reading the Scans**: Each CT scan was loaded from NIfTI files using the `nibabel` library.
2. **Normalization**: The pixel intensity values were clipped to the range `[-1000, 400]` and then normalized to the range `[0, 1]`.
3. **Resizing**: Each scan was resized to have a fixed resolution of `128x128x64` to standardize the data for model input.
4. **Labeling**:
    - Scans with normal lung tissue (without signs of viral pneumonia) were labeled as `0`.
    - Scans showing severe ground-glass opacifications (indicative of viral pneumonia) were labeled as `1`.
5. **Data Splitting**: The dataset was split into training and validation subsets, ensuring balanced representation of both classes.
6. **Saving**: The processed arrays were saved in a compressed `.npz` file for efficient storage and easy reuse.

This preprocessing ensures the data is clean, standardized, and ready for model training. Below is the code for loading the preprocessed data.

In [6]:

```
data = np.load("processed_data.npz")
x_train = data["x_train"]
y_train = data["y_train"]
x_val = data["x_val"]
y_val = data["y_val"]
```

As a result of the preprocessing and data preparation steps, we have the following number of samples:

In [7]:

```
print(
    "Number of samples in train and validation are %d and %d."
    % (x_train.shape[0], x_val.shape[0])
)
```

```
Number of samples in train and validation are 140 and 60.
```

Data Augmentation

To improve the generalization ability of the model and make it more robust to variations in the data, we apply **data augmentation** during the training phase. Data augmentation artificially expands the training dataset by introducing transformed versions of the original data, which helps the model generalize better and reduces the risk of overfitting.

Augmentation Techniques Applied:

1. **Rotation**:

   - The CT scans are randomly rotated by a small angle chosen from a predefined range (`-20°` to `20°`).
   - This simulates variations in the orientation of the scans, which might naturally occur due to patient positioning during imaging.
   - After rotation, pixel values are clipped to ensure they remain within the normalized range `[0, 1]` .

2. **Channel Expansion**:

   - Since the data consists of 3D volumes represented as rank-3 tensors (`samples, height, width, depth`), we add a channel dimension of size `1` at axis `4` to enable 3D convolution operations.
   - This transformation changes the shape of the data to `(samples, height, width, depth, 1)` .

Preprocessing Functions:

- **Training Preprocessing**: During training, each volume is augmented with random rotations and then reshaped to include the channel dimension.
- **Validation Preprocessing**: For validation data, no augmentation is applied to ensure consistent evaluation. Only the channel dimension is added.

By applying these preprocessing steps, we ensure that the training data is diverse, making the model more robust to unseen variations, while the validation data remains unaltered for accurate performance assessment.

In [8]:

```python
@tf.function
def rotate(volume):
    """Rotate the volume by a few degrees"""
    def scipy_rotate(volume):
        angles = [-20, -10, -5, 5, 10, 20]
        angle = random.choice(angles)
        volume = ndimage.rotate(volume, angle, reshape=False)
        volume[volume < 0] = 0
        volume[volume > 1] = 1
        return volume

    augmented_volume = tf.numpy_function(scipy_rotate, [volume], tf.float32)
    augmented_volume.set_shape(volume.shape)
    return augmented_volume


def train_preprocessing(volume, label):
    """Process training data by rotating and adding a channel."""
    volume = rotate(volume)
    volume = tf.expand_dims(volume, axis=3)
    return volume, label


def validation_preprocessing(volume, label):
    """Process validation data by only adding a channel."""
    volume = tf.expand_dims(volume, axis=3)
    return volume, label
```

We now apply the transformations explained earlier and prepare the data for training and validation. We do this in the following steps:

- **Shuffling**: The training dataset is shuffled to prevent the model from learning patterns based on the order of the data.
- **Mapping Preprocessing Functions**: For training, we apply `train_preprocessing` , which includes random rotations and adding a channel dimension. For validation, we apply `validation_preprocessing` , which only adds the channel dimension to ensure consistent evaluations.
- **Batching**: The data is divided into batches of size `2` to optimize memory usage and allow for more frequent updates to the model.
- **Prefetching**: We use `prefetch` to load the next batch while the current one is being processed, improving training efficiency.

In [9]:

```python
train_loader = tf.data.Dataset.from_tensor_slices((x_train, y_train))
validation_loader = tf.data.Dataset.from_tensor_slices((x_val, y_val))

batch_size = 2
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
validation_dataset = (
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
```

```
I0000 00:00:1735057764.895399   35176 gpu_device.cc:2022] Created device /job:localhost/replica:0/task:0/device:
GPU:0 with 5563 MB memory:  -> device: 0, name: NVIDIA GeForce RTX 4060, pci bus id: 0000:01:00.0, compute capab
ility: 8.9
```

Due to the small batch size of 2, the datasets are divided into multiple smaller batches to optimize memory usage and facilitate efficient training. Below are the number of batches created for each dataset:

In [10]:

```python
len(train_dataset)
```

Out[10]:

```
70
```

In [11]:

```python
len(validation_dataset)
```

Out[11]:

```
30
```

Training

Functions

We now introduce several key functions that serve as building blocks for training and designing a 3D CNN model for our medical imaging task. These functions collectively handle tasks such as training, learning rate scheduling, model architecture design, and implementation of advanced 3D convolutional blocks. Below is a summary of their purpose.

Explanation of `train_model`

This function is responsible for orchestrating the training process of a given model. It includes key steps such as setting up a learning rate schedule, compiling the model, and defining callbacks to monitor performance and manage the learning process. Here's a breakdown of each part:

1. **Initialization of Learning Rate Parameters**:
   - `initial_lr = 1e-6` : The starting learning rate, set to a very small value to ensure stable initial training.
   - `target_lr = 1e-5` : The target learning rate after the warm-up phase.
   - `warmup_epochs = 5` : Specifies the number of epochs during which the learning rate will gradually increase from `initial_lr` to `target_lr` . Warm-up phases help the model adjust to training dynamics smoothly, especially in deep networks.
2. **Learning Rate Scheduler**:
   - The nested `scheduler` function adjusts the learning rate based on the current epoch.
   - For the first `warmup_epochs` , the learning rate linearly increases from `initial_lr` to `target_lr` . After the warm-up, it remains constant at `target_lr` .
   - This strategy is used to stabilize training early on and then maintain a consistent learning rate for the remainder of the process.
3. **Model Compilation**: The model is compiled with the following:
   - **Loss Function**: `binary_crossentropy` , used for binary classification tasks, as the one we have to solve.
   - **Optimizer**: `Adam` , which is widely used for its adaptive learning rate and efficient convergence.
   - **Metrics**: `"acc"` , accuracy is tracked as a performance metric.
4. **Callbacks**:
   - `ModelCheckpoint` : Saves the model weights to a file (`{model.name}.keras`) whenever the validation accuracy improves, ensuring that the best version of the model is preserved.
   - `EarlyStopping` : Monitors validation accuracy and stops training if it does not improve for 15 consecutive epochs, preventing overfitting and saving computational resources.
   - `LearningRateScheduler` : Adjusts the learning rate dynamically at each epoch using the defined `scheduler` function.

In [9]:

```python
def train_model(model, train_dataset, validation_dataset, epochs=20):
    initial_lr = 1e-6
    target_lr = 1e-5
    warmup_epochs = 5

    def scheduler(epoch, lr):
        if epoch < warmup_epochs:
            new_lr = initial_lr + (epoch * (target_lr - initial_lr) / warmup_epochs)
            return new_lr
        else:
            return target_lr

    model.compile(
        loss="binary_crossentropy",
        optimizer=keras.optimizers.Adam(learning_rate=initial_lr),
        metrics=["acc"]
    )

    checkpoint_cb = keras.callbacks.ModelCheckpoint(
        f"{model.name}.keras", save_best_only=True
    )
    early_stopping_cb = keras.callbacks.EarlyStopping(monitor="val_acc", patience=15)

    lr_scheduler_cb = keras.callbacks.LearningRateScheduler(scheduler, verbose=0)

    history = model.fit(
        train_dataset,
        validation_data=validation_dataset,
        epochs=epochs,
        shuffle=True,
        verbose=2,
        callbacks=[checkpoint_cb, early_stopping_cb, lr_scheduler_cb],
    )

    return history
```

Flexible Model Creation

The `build_3d_cnn` function is designed to provide a flexible framework for constructing various 3D CNN architectures. It incorporates several parameters to allow modifications, permitting testing of all configurations suggested in the lab boletin. The function can generate models with varying complexity, from standard convolutional layers to advanced architectures incorporating residual and inception blocks.

Key Features:

- **Flexibility**:
  - The function allows toggling between using standard convolutional layers, residual blocks, or residual inception blocks for feature extraction at each level.
  - Parameters like `extra_dense`, `two_conv_per_level`, and `pooling` enable easy experimentation with network depth and pooling strategies.
- **Customization**:
  - You can add an extra dense layer before the output by setting `extra_dense=True`.
  - A second convolutional layer can be added at each level by setting `two_conv_per_level=True`.
  - The pooling mechanism (MaxPooling or AveragePooling) is adjustable via the `pooling` parameter.
- **Advanced Architectures**: By enabling `use_residual` or `use_inception`, you can replace standard convolutional layers with more complex residual or inception-based structures, respectively, to experiment with advanced model designs.

- **Other Parameters**:
  - Adjustable activation functions (`activation`).
  - Easily configurable number of filters for each convolutional layer via the `initial_filters` list.

In [12]:

```
def build_3d_cnn(
    width=128,
    height=128,
    depth=64,
    initial_filters=[64, 64, 128, 256],
    extra_dense=False,
    two_conv_per_level=False,
    activation="relu",
    pooling="max",
    use_residual=False,
    use_inception=False,
    name="3dcnn"
):
    inputs = keras.Input((width, height, depth, 1))
    x = inputs

    for f in initial_filters:
        if use_residual:
            x = residual_block_3d(
                x,
                f,
                activation=activation,
                kernel_regularizer=keras.regularizers.l2(1e-5)
            )
        elif use_inception:
            x = residual_inception_block_3d(
                x,
                f,
                activation=activation,
                kernel_regularizer=keras.regularizers.l2(1e-5)
            )
        else:
            x = layers.Conv3D(
                filters=f,
                kernel_size=3,
                padding="same",
                activation=activation,
                kernel_regularizer=keras.regularizers.l2(1e-5)
            )(x)
            if two_conv_per_level:
                x = layers.Conv3D(
                    filters=f,
                    kernel_size=3,
                    padding="same",
                    activation=activation,
                    kernel_regularizer=keras.regularizers.l2(1e-5)
                )(x)

        if pooling == "avg":
            x = layers.AveragePooling3D(pool_size=2)(x)
        else:
            x = layers.MaxPool3D(pool_size=2)(x)

        x = layers.BatchNormalization()(x)

    x = layers.GlobalAveragePooling3D()(x)
    x = layers.Dense(units=512, activation=activation)(x)
    # x = layers.Dropout(0.5)(x)

    if extra_dense:
        x = layers.Dense(units=100, activation="relu")(x)

    outputs = layers.Dense(units=1, activation="sigmoid")(x)
    model = keras.Model(inputs, outputs, name=name)
    return model
```

Two of the modifications that require our attention are `residual_block_3d` and `residual_inception_block_3d`, as they introduce advanced mechanisms for feature extraction. These will be explained in more detail below:

`residual_block_3d`

This function implements a 3D residual block, which is a critical component for deep networks. Residual blocks allow the network to learn residual mappings instead of direct mappings, facilitating gradient flow and preventing vanishing gradient problems in deeper architectures. Key components include:

1. **Main Path**: Two 3D convolutional layers with a kernel size of 3x3x3, each followed by batch normalization and activation. These layers extract features while normalizing and activating them.
2. **Shortcut Path**: Ensures the input is directly added to the output of the main path. If the input and output dimensions differ, a 1x1x1 convolution is applied to the shortcut to match dimensions.
3. **Addition and Activation**: The shortcut and main path outputs are combined using element-wise addition, followed by an activation function to ensure non-linearity.

In [ ]:

```
def residual_block_3d(
    x,
    filters,
    activation="relu",
    kernel_regularizer=keras.regularizers.l2(1e-5)
):
    shortcut = x

    y = layers.Conv3D(
        filters,
        kernel_size=3,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(x)
    y = layers.BatchNormalization()(y)
    y = layers.Activation(activation)(y)

    y = layers.Conv3D(
        filters,
        kernel_size=3,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(y)
    y = layers.BatchNormalization()(y)
    if x.shape[-1] != filters:
        shortcut = layers.Conv3D(
            filters,
            kernel_size=1,
            padding="same",
            activation=None,
            kernel_regularizer=kernel_regularizer
        )(shortcut)
        shortcut = layers.BatchNormalization()(shortcut)

    y = layers.Add()([shortcut, y])
    y = layers.Activation(activation)(y)

    return y
```

`residual_inception_block_3d`

This function combines the ideas of residual blocks and inception modules. Inception modules are designed for multi-scale feature extraction, while residual connections allow efficient gradient propagation. Key components include:

1. **Path1 (1x1 Convolution)**: Extracts features with minimal spatial reduction, focusing on localized information.
2. **Path2 (1x1 followed by 3x3 Convolution)**: Captures medium-scale features by adding a 3x3 convolution after the 1x1 layer.
3. **Path3 (1x1 followed by 5x5 Convolution)**: Focuses on larger spatial features by incorporating a 5x5 convolution after the 1x1 layer.
4. **Path4 (Pooling)**: Applies a MaxPooling operation followed by a 1x1 convolution to extract abstracted features from larger receptive fields.
5. **Merging**: Combines the outputs of all paths using concatenation, ensuring multi-scale feature representation.
6. **Shortcut Connection**: Similar to `residual_block_3d`, the shortcut path ensures that the input is directly added to the merged paths after a 1x1 convolution (if necessary).
7. **Output**: The combined result is activated using the specified activation function.

In [ ]:

```
def residual_inception_block_3d(
    x,
    filters,
    activation="relu",
    kernel_regularizer=keras.regularizers.l2(1e-5)
):
    """Inception-style 3D residual block."""
    shortcut = x

    path1 = layers.Conv3D(
        filters // 4,
        kernel_size=1,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(x)
    path1 = layers.BatchNormalization()(path1)
    path1 = layers.Activation(activation)(path1)

    path2 = layers.Conv3D(
        filters // 4,
        kernel_size=1,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(x)
    path2 = layers.BatchNormalization()(path2)
    path2 = layers.Activation(activation)(path2)
    path2 = layers.Conv3D(
        filters // 4,
        kernel_size=3,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(path2)
    path2 = layers.BatchNormalization()(path2)
    path2 = layers.Activation(activation)(path2)

    path3 = layers.Conv3D(
        filters // 4,
        kernel_size=1,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(x)
    path3 = layers.BatchNormalization()(path3)
    path3 = layers.Activation(activation)(path3)
    path3 = layers.Conv3D(
        filters // 4,
        kernel_size=5,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(path3)
    path3 = layers.BatchNormalization()(path3)
    path3 = layers.Activation(activation)(path3)

    path4 = layers.MaxPool3D(pool_size=3, strides=1, padding="same")(x)
    path4 = layers.Conv3D(
        filters // 4,
        kernel_size=1,
        padding="same",
        activation=None,
        kernel_regularizer=kernel_regularizer
    )(path4)
    path4 = layers.BatchNormalization()(path4)
    path4 = layers.Activation(activation)(path4)

    merged = layers.Concatenate()([path1, path2, path3, path4])

    if merged.shape[-1] != filters:
        merged = layers.Conv3D(
            filters,
            kernel_size=1,
            padding="same",
            activation=None,
            kernel_regularizer=kernel_regularizer
        )(merged)
        merged = layers.BatchNormalization()(merged)

    if shortcut.shape[-1] != filters:
        shortcut = layers.Conv3D(
            filters,
            kernel_size=1,
            padding="same",
            activation=None,
            kernel_regularizer=kernel_regularizer
        )(shortcut)
        shortcut = layers.BatchNormalization()(shortcut)

    out = layers.Add()([shortcut, merged])
    out = layers.Activation(activation)(out)

    return out
```

Configurations

Once the function for creating the model is complete, we move on to the experimentation phase, for which we create a function to manage individual experiments. This function, `run_single_experiment`, automates the training process, tracks results, and stores outputs for later analysis. Here's what it does:

1. **Model Building**:
   - Builds a 3D CNN model using the parameters defined in the configuration (`cfg["params"]`).
   - Assigns a unique name to the model based on the experiment's configuration.
2. **Training**:
   - Calls the `train_model` function to train the model on the provided datasets (`train_dataset` and `validation_dataset`).
   - Extracts the final training and validation accuracy from the training history.
3. **Results Tracking**:
   - Compiles the experiment's results (configuration name, description, and final accuracies).
   - Saves these results to a CSV file (`results_path`), appending them if the file already exists.
4. **History and Model Saving**:
   - Saves the training history as a JSON file for detailed analysis.
   - Saves the trained model weights as an `.h5` file for future use or deployment.

In [13]:

```
def run_single_experiment(config_name, cfg, train_dataset, validation_dataset, epochs=20, results_path="results
    print(f"\n=== Entrenando modelo '{config_name}' ===")
    params = cfg["params"]

    os.makedirs(history_dir, exist_ok=True)
    os.makedirs(models_dir, exist_ok=True)
```

```
model = build_3d_cnn(
    width=128,
    height=128,
    depth=64,
    **params,
    name=f"3dcnn_{config_name}"
)
model.summary()

history = train_model(model, train_dataset, validation_dataset, epochs=epochs)

final_train_acc = history.history["acc"][-1]
final_val_acc = history.history["val_acc"][-1]

result = {
    "config_name": config_name,
    "description": cfg["description"],
    "final_train_acc": final_train_acc,
    "final_val_acc": final_val_acc,
}

if not os.path.exists(results_path):
    pd.DataFrame([result]).to_csv(results_path, index=False)
else:
    df = pd.read_csv(results_path)
    df = pd.concat([df, pd.DataFrame([result])], ignore_index=True)
    df.to_csv(results_path, index=False)

history_path = os.path.join(history_dir, f"{config_name}_history.json")
with open(history_path, "w") as f:
    json.dump(history.history, f)

model_path = os.path.join(models_dir, f"{config_name}_model.h5")
model.save(model_path)

del model
tf.keras.backend.clear_session()
gc.collect()

print(f"Resultados guardados en '{results_path}'")
print(f"Historial guardado en '{history_path}'")
print(f"Modelo guardado en '{model_path}'")
```

The tested configurations for the 3D CNN model include various architectural modifications to evaluate their impact on performance. These align with the examples provided in the lab bulletin:

1. `original` :
   - **Description**: This is the baseline model without any modifications.
   - **Parameters**:
     - No extra dense layer before the output.
     - A single Conv3D layer per level.
     - ReLU as the activation function.
     - MaxPooling3D for down-sampling.
     - No use of residual or inception blocks.
2. `extra_dense` :
   - **Description**: Adds a dense layer with 100 units and a ReLU activation function before the final output layer.
   - **Parameters**:
     - Includes an extra dense layer.
     - Other parameters remain unchanged from the baseline.
3. `two_conv_per_level` :
   - **Description**: Adds a second Conv3D layer at each level.
   - **Parameters**:
     - Two Conv3D layers per level.
     - Other parameters remain unchanged from the baseline.
4. `relu_to_sigmoid` :
   - **Description**: Replaces ReLU activation functions with Sigmoid activations throughout the network.
   - **Parameters**:
     - Uses Sigmoid activation instead of ReLU.
     - Other parameters remain unchanged from the baseline.
5. `max_to_avg_pooling` :
   - **Description**: Replaces MaxPooling3D layers with AveragePooling3D layers for down-sampling.

- **Parameters**:
    - Uses AveragePooling3D instead of MaxPooling3D.
    - Other parameters remain unchanged from the baseline.

6. `residual_blocks` :

    - **Description**: Replaces standard convolutional layers with 3D residual blocks.
    - **Parameters**:
        - Introduces residual connections to enable gradient flow.
        - Uses an adjusted list of initial filters: [32, 64, 128] (due to computational limitations).
        - Other parameters remain unchanged from the baseline.

7. `inception_blocks` :

    - **Description**: Replaces standard convolutional layers with 3D residual inception blocks.
    - **Parameters**:
        - Incorporates inception-style architecture for multi-scale feature extraction.
        - Uses an adjusted list of initial filters: [32, 64, 128] (due to computational limitations).
        - Other parameters remain unchanged from the baseline.

In [14]:

```python
models_config = {
    "original": {
        "description": "Modelo base sin modificaciones",
        "params": {
            "extra_dense": False,
            "two_conv_per_level": False,
            "activation": "relu",
            "pooling": "max",
            "use_residual": False,
            "use_inception": False,
        },
    },
    "extra_dense": {
        "description": "Añade capa Dense de 100 unidades antes de la salida",
        "params": {
            "extra_dense": True,
            "two_conv_per_level": False,
            "activation": "relu",
            "pooling": "max",
            "use_residual": False,
            "use_inception": False,
        },
    },
    "two_conv_per_level": {
        "description": "Agrega una segunda capa Conv3D en cada bloque",
        "params": {
            "extra_dense": False,
            "two_conv_per_level": True,
            "activation": "relu",
            "pooling": "max",
            "use_residual": False,
            "use_inception": False,
        },
    },
    "relu_to_sigmoid": {
        "description": "Cambia la activación ReLU por sigmoid en todas las capas",
        "params": {
            "extra_dense": False,
            "two_conv_per_level": False,
            "activation": "sigmoid",
            "pooling": "max",
            "use_residual": False,
            "use_inception": False,
        },
    },
    "max_to_avg_pooling": {
        "description": "Usa AveragePooling3D en lugar de MaxPooling3D",
        "params": {
            "extra_dense": False,
            "two_conv_per_level": False,
            "activation": "relu",
            "pooling": "avg",
            "use_residual": False,
            "use_inception": False,
        },
    },
    "residual_blocks": {
        "description": "Reemplaza capas conv por bloques residuales",
        "params": {
            "initial_filters" : [32, 64, 128],
            "extra_dense": False,
            "two_conv_per_level": False,
            "activation": "relu",
            "pooling": "max",
            "use_residual": True,
            "use_inception": False,
        },
    },
    "inception_blocks": {
        "description": "Reemplaza capas conv por bloques inception 3D",
        "params": {
            "initial_filters" : [32, 64, 128],
            "extra_dense": False,
            "two_conv_per_level": False,
            "activation": "relu",
            "pooling": "max",
            "use_residual": False,
            "use_inception": True,
        },
    },
}
```

All these models are then trained:

In [16]:

```
# config_name = "original"
# cfg = models_config[config_name]

# run_single_experiment(
#     config_name=config_name,
#     cfg=cfg,
#     train_dataset=train_dataset,
#     validation_dataset=validation_dataset,
#     epochs=50,
#     results_path="results.csv",
#     history_dir="histories",
#     models_dir="models"
# )
```

```
 === Entrenando modelo 'original' ===
Model: "3dcnn_original"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)        │ (None, 128, 128, 64,   │             0 │
│                                 │ 1)                     │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d (Conv3D)                 │ (None, 128, 128, 64,   │         1,792 │
│                                 │ 64)                    │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d (MaxPooling3D)    │ (None, 64, 64, 32, 64) │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization             │ (None, 64, 64, 32, 64) │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_1 (Conv3D)               │ (None, 64, 64, 32, 64) │       110,656 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_1 (MaxPooling3D)  │ (None, 32, 32, 16, 64) │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_1           │ (None, 32, 32, 16, 64) │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_2 (Conv3D)               │ (None, 32, 32, 16,     │       221,312 │
│                                 │ 128)                   │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_2 (MaxPooling3D)  │ (None, 16, 16, 8, 128) │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_2           │ (None, 16, 16, 8, 128) │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_3 (Conv3D)               │ (None, 16, 16, 8, 256) │       884,992 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_3 (MaxPooling3D)  │ (None, 8, 8, 4, 256)   │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_3           │ (None, 8, 8, 4, 256)   │         1,024 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ global_average_pooling3d        │ (None, 256)            │             0 │
│ (GlobalAveragePooling3D)        │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 512)            │       131,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 1)              │           513 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
```

```
Total params: 1,352,897 (5.16 MB)
Trainable params: 1,351,873 (5.16 MB)
Non-trainable params: 1,024 (4.00 KB)
```

```
Epoch 1/50
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1735052972.617718   14519 service.cc:148] XLA service 0x7fb6080080b0 initialized for platform CUDA (
this does not guarantee that XLA will be used). Devices:
I0000 00:00:1735052972.617756   14519 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 4060, Comp
ute Capability 8.9
2024-12-24 16:09:32.661336: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR cr
ash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1735052972.846097   14519 cuda_dnn.cc:529] Loaded cuDNN version 90300
2024-12-24 16:09:38.953603: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warnin
g : Registers are spilled to local memory in function 'input_reduce_reduce_window_fusion_3', 32 bytes spill stor
es, 32 bytes spill loads

I0000 00:00:1735052978.963309   14519 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at
most once for the lifetime of the process.
```

```
70/70 - 18s - 262ms/step - acc: 0.6071 - loss: 0.6852 - val_acc: 0.5000 - val_loss: 0.6966 - learning_rate: 1.00
00e-06
Epoch 2/50
70/70 - 10s - 141ms/step - acc: 0.5786 - loss: 0.6884 - val_acc: 0.5000 - val_loss: 0.7119 - learning_rate: 2.80
00e-06
Epoch 3/50
70/70 - 10s - 141ms/step - acc: 0.6071 - loss: 0.6755 - val_acc: 0.5000 - val_loss: 0.8198 - learning_rate: 4.60
00e-06
Epoch 4/50
70/70 - 10s - 140ms/step - acc: 0.5571 - loss: 0.6843 - val_acc: 0.5000 - val_loss: 1.1314 - learning_rate: 6.40
00e-06
Epoch 5/50
70/70 - 9s - 123ms/step - acc: 0.5357 - loss: 0.6933 - val_acc: 0.5000 - val_loss: 0.9945 - learning_rate: 8.200
0e-06
Epoch 6/50
70/70 - 10s - 141ms/step - acc: 0.6214 - loss: 0.6745 - val_acc: 0.5000 - val_loss: 1.3803 - learning_rate: 1.00
00e-05
Epoch 7/50
70/70 - 10s - 139ms/step - acc: 0.5929 - loss: 0.6698 - val_acc: 0.5000 - val_loss: 1.2347 - learning_rate: 1.00
00e-05
Epoch 8/50
70/70 - 9s - 126ms/step - acc: 0.5429 - loss: 0.6841 - val_acc: 0.5000 - val_loss: 0.8755 - learning_rate: 1.000
0e-05
Epoch 9/50
70/70 - 10s - 139ms/step - acc: 0.5643 - loss: 0.6780 - val_acc: 0.5000 - val_loss: 0.7405 - learning_rate: 1.00
00e-05
Epoch 10/50
70/70 - 10s - 140ms/step - acc: 0.6143 - loss: 0.6481 - val_acc: 0.5833 - val_loss: 0.7015 - learning_rate: 1.00
00e-05
Epoch 11/50
70/70 - 9s - 127ms/step - acc: 0.6214 - loss: 0.6552 - val_acc: 0.7000 - val_loss: 0.5934 - learning_rate: 1.000
0e-05
Epoch 12/50
70/70 - 10s - 140ms/step - acc: 0.6071 - loss: 0.6582 - val_acc: 0.7000 - val_loss: 0.5961 - learning_rate: 1.00
00e-05
Epoch 13/50
70/70 - 10s - 140ms/step - acc: 0.6643 - loss: 0.6355 - val_acc: 0.7167 - val_loss: 0.5891 - learning_rate: 1.00
00e-05
Epoch 14/50
70/70 - 9s - 127ms/step - acc: 0.6000 - loss: 0.6652 - val_acc: 0.6833 - val_loss: 0.5748 - learning_rate: 1.000
0e-05
Epoch 15/50
70/70 - 10s - 140ms/step - acc: 0.6643 - loss: 0.6309 - val_acc: 0.6833 - val_loss: 0.6104 - learning_rate: 1.00
00e-05
Epoch 16/50
70/70 - 10s - 139ms/step - acc: 0.6571 - loss: 0.6596 - val_acc: 0.6667 - val_loss: 0.5959 - learning_rate: 1.00
00e-05
Epoch 17/50
70/70 - 10s - 139ms/step - acc: 0.5786 - loss: 0.6721 - val_acc: 0.6667 - val_loss: 0.6058 - learning_rate: 1.00
00e-05
Epoch 18/50
70/70 - 9s - 127ms/step - acc: 0.6571 - loss: 0.6231 - val_acc: 0.6833 - val_loss: 0.5490 - learning_rate: 1.000
0e-05
Epoch 19/50
70/70 - 11s - 159ms/step - acc: 0.6714 - loss: 0.6419 - val_acc: 0.7167 - val_loss: 0.5551 - learning_rate: 1.00
00e-05
Epoch 20/50
70/70 - 10s - 140ms/step - acc: 0.6000 - loss: 0.6406 - val_acc: 0.7000 - val_loss: 0.5527 - learning_rate: 1.00
00e-05
Epoch 21/50
70/70 - 9s - 127ms/step - acc: 0.6286 - loss: 0.6514 - val_acc: 0.7500 - val_loss: 0.5391 - learning_rate: 1.000
0e-05
Epoch 22/50
```

```
70/70 - 10s - 140ms/step - acc: 0.6214 - loss: 0.6598 - val_acc: 0.7500 - val_loss: 0.5672 - learning_rate: 1.00
00e-05
Epoch 23/50
70/70 - 10s - 141ms/step - acc: 0.5857 - loss: 0.6453 - val_acc: 0.7500 - val_loss: 0.5668 - learning_rate: 1.00
00e-05
Epoch 24/50
70/70 - 9s - 125ms/step - acc: 0.6143 - loss: 0.6565 - val_acc: 0.6500 - val_loss: 0.5887 - learning_rate: 1.000
0e-05
Epoch 25/50
70/70 - 10s - 139ms/step - acc: 0.7214 - loss: 0.6036 - val_acc: 0.7000 - val_loss: 0.6161 - learning_rate: 1.00
00e-05
Epoch 26/50
70/70 - 10s - 141ms/step - acc: 0.6571 - loss: 0.6056 - val_acc: 0.7667 - val_loss: 0.5151 - learning_rate: 1.00
00e-05
Epoch 27/50
70/70 - 9s - 126ms/step - acc: 0.5500 - loss: 0.6580 - val_acc: 0.7333 - val_loss: 0.5355 - learning_rate: 1.000
0e-05
Epoch 28/50
70/70 - 10s - 139ms/step - acc: 0.7000 - loss: 0.6419 - val_acc: 0.7500 - val_loss: 0.5487 - learning_rate: 1.00
00e-05
Epoch 29/50
70/70 - 10s - 139ms/step - acc: 0.6571 - loss: 0.6252 - val_acc: 0.7167 - val_loss: 0.5265 - learning_rate: 1.00
00e-05
Epoch 30/50
70/70 - 10s - 139ms/step - acc: 0.6429 - loss: 0.6265 - val_acc: 0.7333 - val_loss: 0.5152 - learning_rate: 1.00
00e-05
Epoch 31/50
70/70 - 9s - 125ms/step - acc: 0.6357 - loss: 0.6278 - val_acc: 0.7500 - val_loss: 0.5460 - learning_rate: 1.000
0e-05
Epoch 32/50
70/70 - 10s - 141ms/step - acc: 0.6571 - loss: 0.6118 - val_acc: 0.7500 - val_loss: 0.5022 - learning_rate: 1.00
00e-05
Epoch 33/50
70/70 - 10s - 139ms/step - acc: 0.6429 - loss: 0.6321 - val_acc: 0.7167 - val_loss: 0.5128 - learning_rate: 1.00
00e-05
Epoch 34/50
70/70 - 9s - 126ms/step - acc: 0.6643 - loss: 0.6358 - val_acc: 0.7000 - val_loss: 0.5730 - learning_rate: 1.000
0e-05
Epoch 35/50
70/70 - 10s - 140ms/step - acc: 0.6929 - loss: 0.6235 - val_acc: 0.6500 - val_loss: 0.5776 - learning_rate: 1.00
00e-05
Epoch 36/50
70/70 - 10s - 140ms/step - acc: 0.6571 - loss: 0.6236 - val_acc: 0.7667 - val_loss: 0.5373 - learning_rate: 1.00
00e-05
Epoch 37/50
70/70 - 9s - 127ms/step - acc: 0.6643 - loss: 0.6043 - val_acc: 0.5833 - val_loss: 0.6780 - learning_rate: 1.000
0e-05
Epoch 38/50
70/70 - 10s - 139ms/step - acc: 0.6786 - loss: 0.6314 - val_acc: 0.7500 - val_loss: 0.5255 - learning_rate: 1.00
00e-05
Epoch 39/50
70/70 - 10s - 140ms/step - acc: 0.6286 - loss: 0.6361 - val_acc: 0.6000 - val_loss: 0.6071 - learning_rate: 1.00
00e-05
Epoch 40/50
70/70 - 10s - 139ms/step - acc: 0.6714 - loss: 0.5930 - val_acc: 0.7000 - val_loss: 0.5521 - learning_rate: 1.00
00e-05
Epoch 41/50
70/70 - 9s - 124ms/step - acc: 0.6929 - loss: 0.5959 - val_acc: 0.7500 - val_loss: 0.5088 - learning_rate: 1.000
0e-05
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
Resultados guardados en 'results.csv'
Historial guardado en 'histories/original_history.json'
Modelo guardado en 'models/original_model.h5'
```

In [16]:

```
# config_name = "extra_dense"
# cfg = models_config[config_name]

# run_single_experiment(
#     config_name=config_name,
#     cfg=cfg,
#     train_dataset=train_dataset,
#     validation_dataset=validation_dataset,
#     epochs=50,
#     results_path="results.csv",
#     history_dir="histories",
#     models_dir="models"
# )
```

```
=== Entrenando modelo 'extra_dense' ===
```

Model: "3dcnn_extra_dense"

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)         │ (None, 128, 128, 64,    │             0 │
│                                  │ 1)                      │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ conv3d (Conv3D)                  │ (None, 128, 128, 64,    │         1,792 │
│                                  │ 64)                     │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ max_pooling3d (MaxPooling3D)     │ (None, 64, 64, 32, 64)  │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ batch_normalization              │ (None, 64, 64, 32, 64)  │           256 │
│ (BatchNormalization)             │                         │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ conv3d_1 (Conv3D)                │ (None, 64, 64, 32, 64)  │       110,656 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ max_pooling3d_1 (MaxPooling3D)   │ (None, 32, 32, 16, 64)  │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ batch_normalization_1            │ (None, 32, 32, 16, 64)  │           256 │
│ (BatchNormalization)             │                         │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ conv3d_2 (Conv3D)                │ (None, 32, 32, 16,      │       221,312 │
│                                  │ 128)                    │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ max_pooling3d_2 (MaxPooling3D)   │ (None, 16, 16, 8, 128)  │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ batch_normalization_2            │ (None, 16, 16, 8, 128)  │           512 │
│ (BatchNormalization)             │                         │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ conv3d_3 (Conv3D)                │ (None, 16, 16, 8, 256)  │       884,992 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ max_pooling3d_3 (MaxPooling3D)   │ (None, 8, 8, 4, 256)    │             0 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ batch_normalization_3            │ (None, 8, 8, 4, 256)    │         1,024 │
│ (BatchNormalization)             │                         │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ global_average_pooling3d         │ (None, 256)             │             0 │
│ (GlobalAveragePooling3D)         │                         │               │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dense (Dense)                    │ (None, 512)             │       131,584 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 100)             │        51,300 │
├──────────────────────────────────┼─────────────────────────┼───────────────┤
│ dense_2 (Dense)                  │ (None, 1)               │           101 │
└──────────────────────────────────┴─────────────────────────┴───────────────┘
 Total params: 1,403,785 (5.36 MB)
 Trainable params: 1,402,761 (5.35 MB)
 Non-trainable params: 1,024 (4.00 KB)
```

```
Epoch 1/50
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1735053769.679174   17850 service.cc:148] XLA service 0x7f05dc026a70 initialized for platform CUDA (
this does not guarantee that XLA will be used). Devices:
I0000 00:00:1735053769.679210   17850 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 4060, Comp
ute Capability 8.9
2024-12-24 16:22:49.724054: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR cr
ash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1735053769.914378   17850 cuda_dnn.cc:529] Loaded cuDNN version 90300
2024-12-24 16:22:55.354870: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warnin
g : Registers are spilled to local memory in function 'input_reduce_reduce_window_fusion_3', 32 bytes spill stor
es, 32 bytes spill loads

I0000 00:00:1735053775.364397   17850 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at
most once for the lifetime of the process.
70/70 - 18s - 263ms/step - acc: 0.5214 - loss: 0.6959 - val_acc: 0.5000 - val_loss: 0.6969 - learning_rate: 1.00
00e-06
Epoch 2/50
70/70 - 10s - 137ms/step - acc: 0.5643 - loss: 0.6921 - val_acc: 0.5000 - val_loss: 0.7090 - learning_rate: 2.80
00e-06
Epoch 3/50
70/70 - 8s - 118ms/step - acc: 0.5929 - loss: 0.6910 - val_acc: 0.5000 - val_loss: 0.7804 - learning_rate: 4.600
0e-06
Epoch 4/50
70/70 - 9s - 133ms/step - acc: 0.6500 - loss: 0.6801 - val_acc: 0.5000 - val_loss: 0.9667 - learning_rate: 6.400
0e-06
Epoch 5/50
70/70 - 9s - 135ms/step - acc: 0.6357 - loss: 0.6762 - val_acc: 0.5000 - val_loss: 1.1269 - learning_rate: 8.200
0e-06
Epoch 6/50
70/70 - 9s - 134ms/step - acc: 0.6571 - loss: 0.6612 - val_acc: 0.5000 - val_loss: 1.3748 - learning_rate: 1.000
0e-05
Epoch 7/50
```

```
70/70 - 8s - 120ms/step - acc: 0.5714 - loss: 0.6670 - val_acc: 0.5000 - val_loss: 1.2806 - learning_rate: 1.000
0e-05
Epoch 8/50
70/70 - 10s - 136ms/step - acc: 0.6357 - loss: 0.6645 - val_acc: 0.5000 - val_loss: 1.2197 - learning_rate: 1.00
00e-05
Epoch 9/50
70/70 - 9s - 135ms/step - acc: 0.6214 - loss: 0.6462 - val_acc: 0.5000 - val_loss: 0.8959 - learning_rate: 1.000
0e-05
Epoch 10/50
70/70 - 9s - 122ms/step - acc: 0.6500 - loss: 0.6599 - val_acc: 0.5000 - val_loss: 0.7021 - learning_rate: 1.000
0e-05
Epoch 11/50
70/70 - 10s - 136ms/step - acc: 0.5429 - loss: 0.6816 - val_acc: 0.6833 - val_loss: 0.6059 - learning_rate: 1.00
00e-05
Epoch 12/50
70/70 - 10s - 137ms/step - acc: 0.6000 - loss: 0.6684 - val_acc: 0.7500 - val_loss: 0.5881 - learning_rate: 1.00
00e-05
Epoch 13/50
70/70 - 9s - 124ms/step - acc: 0.5929 - loss: 0.6554 - val_acc: 0.7000 - val_loss: 0.5775 - learning_rate: 1.000
0e-05
Epoch 14/50
70/70 - 10s - 141ms/step - acc: 0.6357 - loss: 0.6611 - val_acc: 0.7000 - val_loss: 0.5802 - learning_rate: 1.00
00e-05
Epoch 15/50
70/70 - 10s - 141ms/step - acc: 0.6000 - loss: 0.6600 - val_acc: 0.6833 - val_loss: 0.5593 - learning_rate: 1.00
00e-05
Epoch 16/50
70/70 - 9s - 135ms/step - acc: 0.6643 - loss: 0.6573 - val_acc: 0.7500 - val_loss: 0.5725 - learning_rate: 1.000
0e-05
Epoch 17/50
70/70 - 9s - 122ms/step - acc: 0.6786 - loss: 0.6310 - val_acc: 0.7000 - val_loss: 0.5518 - learning_rate: 1.000
0e-05
Epoch 18/50
70/70 - 10s - 137ms/step - acc: 0.6286 - loss: 0.6581 - val_acc: 0.7167 - val_loss: 0.5489 - learning_rate: 1.00
00e-05
Epoch 19/50
70/70 - 9s - 135ms/step - acc: 0.5929 - loss: 0.6545 - val_acc: 0.6833 - val_loss: 0.5644 - learning_rate: 1.000
0e-05
Epoch 20/50
70/70 - 9s - 122ms/step - acc: 0.6357 - loss: 0.6552 - val_acc: 0.7500 - val_loss: 0.5353 - learning_rate: 1.000
0e-05
Epoch 21/50
70/70 - 9s - 135ms/step - acc: 0.6286 - loss: 0.6597 - val_acc: 0.7333 - val_loss: 0.5600 - learning_rate: 1.000
0e-05
Epoch 22/50
70/70 - 10s - 143ms/step - acc: 0.6143 - loss: 0.6472 - val_acc: 0.7333 - val_loss: 0.5679 - learning_rate: 1.00
00e-05
Epoch 23/50
70/70 - 9s - 130ms/step - acc: 0.6786 - loss: 0.6246 - val_acc: 0.7333 - val_loss: 0.5960 - learning_rate: 1.000
0e-05
Epoch 24/50
70/70 - 9s - 134ms/step - acc: 0.6786 - loss: 0.6430 - val_acc: 0.7333 - val_loss: 0.5768 - learning_rate: 1.000
0e-05
Epoch 25/50
70/70 - 10s - 137ms/step - acc: 0.6000 - loss: 0.6653 - val_acc: 0.7333 - val_loss: 0.5263 - learning_rate: 1.00
00e-05
Epoch 26/50
70/70 - 11s - 154ms/step - acc: 0.6786 - loss: 0.6435 - val_acc: 0.7833 - val_loss: 0.5450 - learning_rate: 1.00
00e-05
Epoch 27/50
70/70 - 8s - 116ms/step - acc: 0.6500 - loss: 0.6278 - val_acc: 0.5333 - val_loss: 0.7481 - learning_rate: 1.000
0e-05
Epoch 28/50
70/70 - 10s - 137ms/step - acc: 0.6857 - loss: 0.5989 - val_acc: 0.7333 - val_loss: 0.5505 - learning_rate: 1.00
00e-05
Epoch 29/50
70/70 - 10s - 137ms/step - acc: 0.6500 - loss: 0.6340 - val_acc: 0.7333 - val_loss: 0.5174 - learning_rate: 1.00
00e-05
Epoch 30/50
70/70 - 8s - 121ms/step - acc: 0.7071 - loss: 0.6016 - val_acc: 0.7167 - val_loss: 0.5451 - learning_rate: 1.000
0e-05
Epoch 31/50
70/70 - 9s - 135ms/step - acc: 0.5786 - loss: 0.6523 - val_acc: 0.7167 - val_loss: 0.5622 - learning_rate: 1.000
0e-05
Epoch 32/50
70/70 - 10s - 144ms/step - acc: 0.5786 - loss: 0.6642 - val_acc: 0.6833 - val_loss: 0.6202 - learning_rate: 1.00
00e-05
Epoch 33/50
70/70 - 8s - 121ms/step - acc: 0.6643 - loss: 0.6375 - val_acc: 0.7500 - val_loss: 0.5433 - learning_rate: 1.000
0e-05
Epoch 34/50
70/70 - 11s - 158ms/step - acc: 0.7000 - loss: 0.6100 - val_acc: 0.6500 - val_loss: 0.6009 - learning_rate: 1.00
00e-05
```

```
Epoch 35/50
70/70 - 9s - 135ms/step - acc: 0.6786 - loss: 0.5909 - val_acc: 0.6333 - val_loss: 0.5921 - learning_rate: 1.000
0e-05
Epoch 36/50
70/70 - 9s - 135ms/step - acc: 0.6571 - loss: 0.6144 - val_acc: 0.7333 - val_loss: 0.5422 - learning_rate: 1.000
0e-05
Epoch 37/50
70/70 - 8s - 120ms/step - acc: 0.6643 - loss: 0.6141 - val_acc: 0.7500 - val_loss: 0.5182 - learning_rate: 1.000
0e-05
Epoch 38/50
70/70 - 9s - 135ms/step - acc: 0.6500 - loss: 0.5974 - val_acc: 0.7333 - val_loss: 0.5348 - learning_rate: 1.000
0e-05
Epoch 39/50
70/70 - 9s - 135ms/step - acc: 0.7357 - loss: 0.5807 - val_acc: 0.7167 - val_loss: 0.5569 - learning_rate: 1.000
0e-05
Epoch 40/50
70/70 - 8s - 120ms/step - acc: 0.6214 - loss: 0.6425 - val_acc: 0.6500 - val_loss: 0.6702 - learning_rate: 1.000
0e-05
Epoch 41/50
70/70 - 10s - 136ms/step - acc: 0.6571 - loss: 0.6228 - val_acc: 0.5667 - val_loss: 0.6475 - learning_rate: 1.00
00e-05
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
Resultados guardados en 'results.csv'
Historial guardado en 'histories/extra_dense_history.json'
Modelo guardado en 'models/extra_dense_model.h5'
```

In [17]:

```
# config_name = "two_conv_per_level"
# cfg = models_config[config_name]

# run_single_experiment(
#     config_name=config_name,
#     cfg=cfg,
#     train_dataset=train_dataset,
#     validation_dataset=validation_dataset,
#     epochs=50,
#     results_path="results.csv",
#     history_dir="histories",
#     models_dir="models"
# )
```

```
=== Entrenando modelo 'two_conv_per_level' ===
```

Model: "3dcnn_two_conv_per_level"

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)         │ (None, 128, 128, 64,   │             0 │
│                                  │ 1)                     │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d (Conv3D)                  │ (None, 128, 128, 64,   │         1,792 │
│                                  │ 64)                    │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_1 (Conv3D)                │ (None, 128, 128, 64,   │       110,656 │
│                                  │ 64)                    │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d (MaxPooling3D)     │ (None, 64, 64, 32, 64) │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization              │ (None, 64, 64, 32, 64) │           256 │
│ (BatchNormalization)             │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_2 (Conv3D)                │ (None, 64, 64, 32, 64) │       110,656 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_3 (Conv3D)                │ (None, 64, 64, 32, 64) │       110,656 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_1 (MaxPooling3D)   │ (None, 32, 32, 16, 64) │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_1            │ (None, 32, 32, 16, 64) │           256 │
│ (BatchNormalization)             │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_4 (Conv3D)                │ (None, 32, 32, 16,     │       221,312 │
│                                  │ 128)                   │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_5 (Conv3D)                │ (None, 32, 32, 16,     │       442,496 │
│                                  │ 128)                   │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_2 (MaxPooling3D)   │ (None, 16, 16, 8, 128) │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_2            │ (None, 16, 16, 8, 128) │           512 │
│ (BatchNormalization)             │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_6 (Conv3D)                │ (None, 16, 16, 8, 256) │       884,992 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_7 (Conv3D)                │ (None, 16, 16, 8, 256) │     1,769,728 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_3 (MaxPooling3D)   │ (None, 8, 8, 4, 256)   │             0 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_3            │ (None, 8, 8, 4, 256)   │         1,024 │
│ (BatchNormalization)             │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ global_average_pooling3d         │ (None, 256)            │             0 │
│ (GlobalAveragePooling3D)         │                        │               │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                    │ (None, 512)            │       131,584 │
├──────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 1)              │           513 │
└──────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 3,786,433 (14.44 MB)
 Trainable params: 3,785,409 (14.44 MB)
 Non-trainable params: 1,024 (4.00 KB)
Epoch 1/50
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1735054384.137162   21191 service.cc:148] XLA service 0x7f7890002610 initialized for platform CUDA (
this does not guarantee that XLA will be used). Devices:
I0000 00:00:1735054384.137201   21191 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 4060, Comp
ute Capability 8.9
2024-12-24 16:33:04.206504: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR cr
ash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1735054384.426640   21191 cuda_dnn.cc:529] Loaded cuDNN version 90300
2024-12-24 16:33:09.432594: E external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm eng0{}
for conv (f32[2,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,64,128,128,64]{4,3,2,1,0}, f32[64,64,3,3,
3]{4,3,2,1,0}, f32[64]{0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012->bf012, custom_call_targ
et="__cudnn$convBiasActivationForward", backend_config={"cudnn_conv_backend_config":{"activation_mode":"kNone","
conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule":false,"operation_queue_
id":"0","wait_on_operation_queues":[]} is taking a while...
2024-12-24 16:33:11.539298: E external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took 3.1
39492635s
Trying algorithm eng0{} for conv (f32[2,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,64,128,128,64]{4,
3,2,1,0}, f32[64,64,3,3,3]{4,3,2,1,0}, f32[64]{0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012-
>bf012, custom_call_target="__cudnn$convBiasActivationForward", backend_config={"cudnn_conv_backend_config":{"ac
tivation_mode":"kNone","conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule"
:false,"operation_queue_id":"0","wait_on_operation_queues":[]} is taking a while...
E0000 00:00:1735054396.739921   21191 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal acc
uracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735054397.014451   21191 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal acc
uracy. There may be a missing warmup execution, please investigate in Nsight Systems.
2024-12-24 16:33:18.215631: E external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm eng0{}
for conv (f32[2,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,64,128,128,64]{4,3,2,1,0}, f32[64,64,3,3,
3]{4,3,2,1,0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012->bf012, custom_call_target="__cudnn$
convBackwardInput", backend_config={"cudnn_conv_backend_config":{"activation_mode":"kNone","conv_result_scale":1
,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule":false,"operation_queue_id":"0","wait_on_ope
ration_queues":[]} is taking a while...
2024-12-24 16:33:19.706127: E external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took 2.5
22923621s
Trying algorithm eng0{} for conv (f32[2,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,64,128,128,64]{4,
3,2,1,0}, f32[64,64,3,3,3]{4,3,2,1,0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012->bf012, cust
om_call_target="__cudnn$convBackwardInput", backend_config={"cudnn_conv_backend_config":{"activation_mode":"kNon
e","conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule":false,"operation_qu
eue_id":"0","wait_on_operation_queues":[]} is taking a while...
2024-12-24 16:33:25.706937: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warnin
g : Registers are spilled to local memory in function 'input_reduce_reduce_window_fusion_3', 32 bytes spill stor
es, 32 bytes spill loads

I0000 00:00:1735054405.719134   21191 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at
most once for the lifetime of the process.
2024-12-24 16:33:48.229762: E external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm eng0{}
for conv (f32[2,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,64,128,128,64]{4,3,2,1,0}, f32[64,64,3,3,
3]{4,3,2,1,0}, f32[64]{0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012->bf012, custom_call_targ
et="__cudnn$convBiasActivationForward", backend_config={"cudnn_conv_backend_config":{"activation_mode":"kRelu","
conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule":false,"operation_queue_
id":"0","wait_on_operation_queues":[]} is taking a while...
2024-12-24 16:33:50.328032: E external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took 3.1
30753194s
Trying algorithm eng0{} for conv (f32[2,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,64,128,128,64]{4,
3,2,1,0}, f32[64,64,3,3,3]{4,3,2,1,0}, f32[64]{0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012-
>bf012, custom_call_target="__cudnn$convBiasActivationForward", backend_config={"cudnn_conv_backend_config":{"ac
tivation_mode":"kRelu","conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule"
:false,"operation_queue_id":"0","wait_on_operation_queues":[]} is taking a while...
```

```
70/70 - 53s - 750ms/step - acc: 0.4929 - loss: 0.7022 - val_acc: 0.5000 - val_loss: 0.7047 - learning_rate: 1.00
00e-06
Epoch 2/50
70/70 - 22s - 311ms/step - acc: 0.5929 - loss: 0.6916 - val_acc: 0.5000 - val_loss: 0.7013 - learning_rate: 2.80
00e-06
Epoch 3/50
70/70 - 21s - 295ms/step - acc: 0.5857 - loss: 0.6912 - val_acc: 0.5000 - val_loss: 0.7000 - learning_rate: 4.60
00e-06
Epoch 4/50
70/70 - 21s - 294ms/step - acc: 0.5643 - loss: 0.6890 - val_acc: 0.5000 - val_loss: 0.7015 - learning_rate: 6.40
00e-06
Epoch 5/50
70/70 - 22s - 307ms/step - acc: 0.6000 - loss: 0.6834 - val_acc: 0.5000 - val_loss: 0.8309 - learning_rate: 8.20
00e-06
Epoch 6/50
70/70 - 20s - 292ms/step - acc: 0.6714 - loss: 0.6643 - val_acc: 0.5000 - val_loss: 1.0052 - learning_rate: 1.00
00e-05
Epoch 7/50
70/70 - 20s - 292ms/step - acc: 0.5214 - loss: 0.6984 - val_acc: 0.5000 - val_loss: 0.9853 - learning_rate: 1.00
00e-05
Epoch 8/50
70/70 - 21s - 307ms/step - acc: 0.6643 - loss: 0.6640 - val_acc: 0.5000 - val_loss: 0.9465 - learning_rate: 1.00
00e-05
Epoch 9/50
70/70 - 20s - 292ms/step - acc: 0.5857 - loss: 0.6772 - val_acc: 0.5000 - val_loss: 0.8349 - learning_rate: 1.00
00e-05
```

```
Epoch 10/50
70/70 - 20s - 292ms/step - acc: 0.6429 - loss: 0.6410 - val_acc: 0.5500 - val_loss: 0.7365 - learning_rate: 1.00
00e-05
Epoch 11/50
70/70 - 22s - 311ms/step - acc: 0.6357 - loss: 0.6522 - val_acc: 0.7167 - val_loss: 0.6530 - learning_rate: 1.00
00e-05
Epoch 12/50
70/70 - 21s - 296ms/step - acc: 0.6500 - loss: 0.6440 - val_acc: 0.7333 - val_loss: 0.5192 - learning_rate: 1.00
00e-05
Epoch 13/50
70/70 - 21s - 294ms/step - acc: 0.7143 - loss: 0.5951 - val_acc: 0.7333 - val_loss: 0.5098 - learning_rate: 1.00
00e-05
Epoch 14/50
70/70 - 20s - 293ms/step - acc: 0.6286 - loss: 0.6767 - val_acc: 0.5500 - val_loss: 0.6621 - learning_rate: 1.00
00e-05
Epoch 15/50
70/70 - 21s - 307ms/step - acc: 0.6643 - loss: 0.6337 - val_acc: 0.7000 - val_loss: 0.5324 - learning_rate: 1.00
00e-05
Epoch 16/50
70/70 - 20s - 292ms/step - acc: 0.6286 - loss: 0.6371 - val_acc: 0.6667 - val_loss: 0.5378 - learning_rate: 1.00
00e-05
Epoch 17/50
70/70 - 20s - 292ms/step - acc: 0.6071 - loss: 0.6662 - val_acc: 0.6833 - val_loss: 0.5565 - learning_rate: 1.00
00e-05
Epoch 18/50
70/70 - 21s - 307ms/step - acc: 0.6929 - loss: 0.6149 - val_acc: 0.6833 - val_loss: 0.5958 - learning_rate: 1.00
00e-05
Epoch 19/50
70/70 - 21s - 295ms/step - acc: 0.6929 - loss: 0.6054 - val_acc: 0.7500 - val_loss: 0.4917 - learning_rate: 1.00
00e-05
Epoch 20/50
70/70 - 20s - 293ms/step - acc: 0.6643 - loss: 0.6323 - val_acc: 0.5667 - val_loss: 0.8367 - learning_rate: 1.00
00e-05
Epoch 21/50
70/70 - 21s - 307ms/step - acc: 0.7000 - loss: 0.6067 - val_acc: 0.7000 - val_loss: 0.4917 - learning_rate: 1.00
00e-05
Epoch 22/50
70/70 - 20s - 292ms/step - acc: 0.6929 - loss: 0.6027 - val_acc: 0.6667 - val_loss: 0.5598 - learning_rate: 1.00
00e-05
Epoch 23/50
70/70 - 20s - 292ms/step - acc: 0.7143 - loss: 0.5894 - val_acc: 0.6500 - val_loss: 0.6940 - learning_rate: 1.00
00e-05
Epoch 24/50
70/70 - 21s - 307ms/step - acc: 0.6714 - loss: 0.6278 - val_acc: 0.6000 - val_loss: 0.6479 - learning_rate: 1.00
00e-05
Epoch 25/50
70/70 - 20s - 292ms/step - acc: 0.7286 - loss: 0.5859 - val_acc: 0.6000 - val_loss: 0.7409 - learning_rate: 1.00
00e-05
Epoch 26/50
70/70 - 21s - 295ms/step - acc: 0.7714 - loss: 0.5697 - val_acc: 0.7167 - val_loss: 0.4822 - learning_rate: 1.00
00e-05
Epoch 27/50
70/70 - 22s - 308ms/step - acc: 0.6429 - loss: 0.6217 - val_acc: 0.6000 - val_loss: 0.5855 - learning_rate: 1.00
00e-05
Epoch 28/50
70/70 - 20s - 292ms/step - acc: 0.7500 - loss: 0.5701 - val_acc: 0.5833 - val_loss: 0.7514 - learning_rate: 1.00
00e-05
Epoch 29/50
70/70 - 20s - 292ms/step - acc: 0.7214 - loss: 0.5886 - val_acc: 0.7500 - val_loss: 0.5204 - learning_rate: 1.00
00e-05
Epoch 30/50
70/70 - 21s - 307ms/step - acc: 0.6929 - loss: 0.5529 - val_acc: 0.5833 - val_loss: 0.6008 - learning_rate: 1.00
00e-05
Epoch 31/50
70/70 - 20s - 292ms/step - acc: 0.7571 - loss: 0.5611 - val_acc: 0.6167 - val_loss: 0.8184 - learning_rate: 1.00
00e-05
Epoch 32/50
70/70 - 20s - 292ms/step - acc: 0.7643 - loss: 0.5533 - val_acc: 0.6667 - val_loss: 0.7165 - learning_rate: 1.00
00e-05
Epoch 33/50
70/70 - 21s - 307ms/step - acc: 0.7286 - loss: 0.5325 - val_acc: 0.7333 - val_loss: 0.5902 - learning_rate: 1.00
00e-05
Epoch 34/50
70/70 - 20s - 292ms/step - acc: 0.7500 - loss: 0.5437 - val_acc: 0.7833 - val_loss: 0.6034 - learning_rate: 1.00
00e-05
Epoch 35/50
70/70 - 21s - 295ms/step - acc: 0.8214 - loss: 0.4713 - val_acc: 0.7500 - val_loss: 0.4490 - learning_rate: 1.00
00e-05
Epoch 36/50
70/70 - 22s - 311ms/step - acc: 0.7500 - loss: 0.5012 - val_acc: 0.8000 - val_loss: 0.3841 - learning_rate: 1.00
00e-05
Epoch 37/50
70/70 - 20s - 293ms/step - acc: 0.8000 - loss: 0.4463 - val_acc: 0.7500 - val_loss: 0.4871 - learning_rate: 1.00
```

```
00e-05
Epoch 38/50
70/70 - 20s - 292ms/step - acc: 0.7214 - loss: 0.5207 - val_acc: 0.7333 - val_loss: 0.4730 - learning_rate: 1.00
00e-05
Epoch 39/50
70/70 - 21s - 307ms/step - acc: 0.8071 - loss: 0.5003 - val_acc: 0.5833 - val_loss: 0.9691 - learning_rate: 1.00
00e-05
Epoch 40/50
70/70 - 20s - 292ms/step - acc: 0.7429 - loss: 0.5257 - val_acc: 0.5000 - val_loss: 1.7318 - learning_rate: 1.00
00e-05
Epoch 41/50
70/70 - 20s - 292ms/step - acc: 0.7571 - loss: 0.5241 - val_acc: 0.7833 - val_loss: 0.4338 - learning_rate: 1.00
00e-05
Epoch 42/50
70/70 - 21s - 307ms/step - acc: 0.7286 - loss: 0.5398 - val_acc: 0.8167 - val_loss: 0.4296 - learning_rate: 1.00
00e-05
Epoch 43/50
70/70 - 20s - 292ms/step - acc: 0.8000 - loss: 0.4814 - val_acc: 0.7333 - val_loss: 0.5536 - learning_rate: 1.00
00e-05
Epoch 44/50
70/70 - 20s - 292ms/step - acc: 0.8000 - loss: 0.4871 - val_acc: 0.7000 - val_loss: 0.6197 - learning_rate: 1.00
00e-05
Epoch 45/50
70/70 - 22s - 308ms/step - acc: 0.7857 - loss: 0.4733 - val_acc: 0.7667 - val_loss: 0.4777 - learning_rate: 1.00
00e-05
Epoch 46/50
70/70 - 20s - 292ms/step - acc: 0.8286 - loss: 0.4250 - val_acc: 0.8000 - val_loss: 0.4305 - learning_rate: 1.00
00e-05
Epoch 47/50
70/70 - 20s - 292ms/step - acc: 0.7429 - loss: 0.4844 - val_acc: 0.7000 - val_loss: 0.4576 - learning_rate: 1.00
00e-05
Epoch 48/50
70/70 - 22s - 308ms/step - acc: 0.8000 - loss: 0.4366 - val_acc: 0.8333 - val_loss: 0.4390 - learning_rate: 1.00
00e-05
Epoch 49/50
70/70 - 20s - 291ms/step - acc: 0.8000 - loss: 0.4336 - val_acc: 0.8167 - val_loss: 0.4378 - learning_rate: 1.00
00e-05
Epoch 50/50
70/70 - 20s - 292ms/step - acc: 0.8143 - loss: 0.4382 - val_acc: 0.7667 - val_loss: 0.4948 - learning_rate: 1.00
00e-05
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
Resultados guardados en 'results.csv'
Historial guardado en 'histories/two_conv_per_level_history.json'
Modelo guardado en 'models/two_conv_per_level_model.h5'
```

In [15]:

```python
# config_name = "relu_to_sigmoid"
# cfg = models_config[config_name]

# run_single_experiment(
#     config_name=config_name,
#     cfg=cfg,
#     train_dataset=train_dataset,
#     validation_dataset=validation_dataset,
#     epochs=50,
#     results_path="results.csv",
#     history_dir="histories",
#     models_dir="models"
# )
```

```
=== Entrenando modelo 'relu_to_sigmoid' ===
```

Model: "3dcnn_relu_to_sigmoid"

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)        │ (None, 128, 128, 64,   │             0 │
│                                 │ 1)                     │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d (Conv3D)                 │ (None, 128, 128, 64,   │         1,792 │
│                                 │ 64)                    │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d (MaxPooling3D)    │ (None, 64, 64, 32, 64) │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization             │ (None, 64, 64, 32, 64) │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_1 (Conv3D)               │ (None, 64, 64, 32, 64) │       110,656 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_1 (MaxPooling3D)  │ (None, 32, 32, 16, 64) │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_1           │ (None, 32, 32, 16, 64) │           256 │
│ (BatchNormalization)            │                        │               │

├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_2 (Conv3D)               │ (None, 32, 32, 16,     │       221,312 │
│                                 │ 128)                   │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_2 (MaxPooling3D)  │ (None, 16, 16, 8, 128) │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_2           │ (None, 16, 16, 8, 128) │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_3 (Conv3D)               │ (None, 16, 16, 8, 256) │       884,992 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling3d_3 (MaxPooling3D)  │ (None, 8, 8, 4, 256)   │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_3           │ (None, 8, 8, 4, 256)   │         1,024 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ global_average_pooling3d        │ (None, 256)            │             0 │
│ (GlobalAveragePooling3D)        │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 512)            │       131,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 1)              │           513 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
```

```
 Total params: 1,352,897 (5.16 MB)
 Trainable params: 1,351,873 (5.16 MB)
 Non-trainable params: 1,024 (4.00 KB)
```

```
Epoch 1/50
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1735055511.070940   25315 service.cc:148] XLA service 0x7f76140186e0 initialized for platform CUDA (
this does not guarantee that XLA will be used). Devices:
I0000 00:00:1735055511.070975   25315 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 4060, Comp
ute Capability 8.9
2024-12-24 16:51:51.114819: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR cr
ash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1735055511.285850   25315 cuda_dnn.cc:529] Loaded cuDNN version 90300
I0000 00:00:1735055516.714718   25315 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at
most once for the lifetime of the process.
```

```
70/70 - 17s - 244ms/step - acc: 0.5000 - loss: 0.7277 - val_acc: 0.5000 - val_loss: 0.7290 - learning_rate: 1.00
00e-06
Epoch 2/50
70/70 - 10s - 136ms/step - acc: 0.5000 - loss: 0.7230 - val_acc: 0.5000 - val_loss: 0.7295 - learning_rate: 2.80
00e-06
Epoch 3/50
70/70 - 9s - 135ms/step - acc: 0.5000 - loss: 0.7175 - val_acc: 0.5000 - val_loss: 0.7210 - learning_rate: 4.600
0e-06
Epoch 4/50
70/70 - 8s - 120ms/step - acc: 0.5000 - loss: 0.7142 - val_acc: 0.5000 - val_loss: 0.7009 - learning_rate: 6.400
0e-06
Epoch 5/50
70/70 - 10s - 136ms/step - acc: 0.5286 - loss: 0.6910 - val_acc: 0.5000 - val_loss: 0.6991 - learning_rate: 8.20
00e-06
Epoch 6/50
70/70 - 9s - 135ms/step - acc: 0.5286 - loss: 0.6877 - val_acc: 0.5000 - val_loss: 0.8111 - learning_rate: 1.000
0e-05
Epoch 7/50
70/70 - 8s - 118ms/step - acc: 0.5857 - loss: 0.6849 - val_acc: 0.5000 - val_loss: 0.9131 - learning_rate: 1.000
0e-05
Epoch 8/50
70/70 - 9s - 134ms/step - acc: 0.5786 - loss: 0.6787 - val_acc: 0.5000 - val_loss: 0.9201 - learning_rate: 1.000
0e-05
Epoch 9/50
```

```
70/70 - 11s - 159ms/step - acc: 0.5143 - loss: 0.6964 - val_acc: 0.5000 - val_loss: 0.8474 - learning_rate: 1.00
00e-05
Epoch 10/50
70/70 - 8s - 119ms/step - acc: 0.5714 - loss: 0.6771 - val_acc: 0.5000 - val_loss: 0.7280 - learning_rate: 1.000
0e-05
Epoch 11/50
70/70 - 10s - 136ms/step - acc: 0.6000 - loss: 0.6780 - val_acc: 0.6667 - val_loss: 0.6231 - learning_rate: 1.00
00e-05
Epoch 12/50
70/70 - 10s - 137ms/step - acc: 0.5643 - loss: 0.6831 - val_acc: 0.7167 - val_loss: 0.5929 - learning_rate: 1.00
00e-05
Epoch 13/50
70/70 - 10s - 136ms/step - acc: 0.6071 - loss: 0.6616 - val_acc: 0.7000 - val_loss: 0.5840 - learning_rate: 1.00
00e-05
Epoch 14/50
70/70 - 9s - 122ms/step - acc: 0.5429 - loss: 0.6831 - val_acc: 0.7000 - val_loss: 0.5789 - learning_rate: 1.000
0e-05
Epoch 15/50
70/70 - 10s - 136ms/step - acc: 0.6214 - loss: 0.6516 - val_acc: 0.7167 - val_loss: 0.5758 - learning_rate: 1.00
00e-05
Epoch 16/50
70/70 - 9s - 135ms/step - acc: 0.5786 - loss: 0.6954 - val_acc: 0.6667 - val_loss: 0.5801 - learning_rate: 1.000
0e-05
Epoch 17/50
70/70 - 10s - 139ms/step - acc: 0.5786 - loss: 0.6612 - val_acc: 0.6833 - val_loss: 0.5783 - learning_rate: 1.00
00e-05
Epoch 18/50
70/70 - 9s - 134ms/step - acc: 0.6857 - loss: 0.6598 - val_acc: 0.6833 - val_loss: 0.5909 - learning_rate: 1.000
0e-05
Epoch 19/50
70/70 - 10s - 136ms/step - acc: 0.6214 - loss: 0.6769 - val_acc: 0.7000 - val_loss: 0.5722 - learning_rate: 1.00
00e-05
Epoch 20/50
70/70 - 8s - 121ms/step - acc: 0.6571 - loss: 0.6306 - val_acc: 0.7000 - val_loss: 0.5616 - learning_rate: 1.000
0e-05
Epoch 21/50
70/70 - 10s - 136ms/step - acc: 0.6143 - loss: 0.6650 - val_acc: 0.6667 - val_loss: 0.5616 - learning_rate: 1.00
00e-05
Epoch 22/50
70/70 - 9s - 136ms/step - acc: 0.6571 - loss: 0.6456 - val_acc: 0.6833 - val_loss: 0.5516 - learning_rate: 1.000
0e-05
Epoch 23/50
70/70 - 9s - 135ms/step - acc: 0.5786 - loss: 0.6839 - val_acc: 0.6833 - val_loss: 0.5760 - learning_rate: 1.000
0e-05
Epoch 24/50
70/70 - 8s - 118ms/step - acc: 0.5571 - loss: 0.6635 - val_acc: 0.7167 - val_loss: 0.5688 - learning_rate: 1.000
0e-05
Epoch 25/50
70/70 - 11s - 151ms/step - acc: 0.6000 - loss: 0.6691 - val_acc: 0.7167 - val_loss: 0.5760 - learning_rate: 1.00
00e-05
Epoch 26/50
70/70 - 9s - 135ms/step - acc: 0.6857 - loss: 0.6514 - val_acc: 0.7333 - val_loss: 0.5525 - learning_rate: 1.000
0e-05
Epoch 27/50
70/70 - 8s - 118ms/step - acc: 0.6214 - loss: 0.6556 - val_acc: 0.7167 - val_loss: 0.5592 - learning_rate: 1.000
0e-05
Epoch 28/50
70/70 - 9s - 135ms/step - acc: 0.6143 - loss: 0.6333 - val_acc: 0.7167 - val_loss: 0.5427 - learning_rate: 1.000
0e-05
Epoch 29/50
70/70 - 10s - 141ms/step - acc: 0.6000 - loss: 0.6620 - val_acc: 0.7333 - val_loss: 0.5483 - learning_rate: 1.00
00e-05
Epoch 30/50
70/70 - 9s - 134ms/step - acc: 0.6071 - loss: 0.6694 - val_acc: 0.7500 - val_loss: 0.5536 - learning_rate: 1.000
0e-05
Epoch 31/50
70/70 - 8s - 117ms/step - acc: 0.6714 - loss: 0.6443 - val_acc: 0.7000 - val_loss: 0.5612 - learning_rate: 1.000
0e-05
Epoch 32/50
70/70 - 9s - 135ms/step - acc: 0.6429 - loss: 0.6499 - val_acc: 0.7167 - val_loss: 0.5390 - learning_rate: 1.000
0e-05
Epoch 33/50
70/70 - 9s - 135ms/step - acc: 0.5857 - loss: 0.6620 - val_acc: 0.7167 - val_loss: 0.5420 - learning_rate: 1.000
0e-05
Epoch 34/50
70/70 - 8s - 119ms/step - acc: 0.5643 - loss: 0.6820 - val_acc: 0.7167 - val_loss: 0.5545 - learning_rate: 1.000
0e-05
Epoch 35/50
70/70 - 9s - 135ms/step - acc: 0.5714 - loss: 0.6642 - val_acc: 0.7333 - val_loss: 0.5544 - learning_rate: 1.000
0e-05
Epoch 36/50
70/70 - 9s - 134ms/step - acc: 0.6714 - loss: 0.6345 - val_acc: 0.7333 - val_loss: 0.5462 - learning_rate: 1.000
0e-05
```

```
Epoch 37/50
70/70 - 8s - 120ms/step - acc: 0.6286 - loss: 0.6395 - val_acc: 0.7333 - val_loss: 0.5301 - learning_rate: 1.000
0e-05
Epoch 38/50
70/70 - 9s - 135ms/step - acc: 0.6071 - loss: 0.6502 - val_acc: 0.7500 - val_loss: 0.5353 - learning_rate: 1.000
0e-05
Epoch 39/50
70/70 - 9s - 134ms/step - acc: 0.6357 - loss: 0.6552 - val_acc: 0.7333 - val_loss: 0.5334 - learning_rate: 1.000
0e-05
Epoch 40/50
70/70 - 9s - 135ms/step - acc: 0.6357 - loss: 0.6673 - val_acc: 0.7333 - val_loss: 0.5574 - learning_rate: 1.000
0e-05
Epoch 41/50
70/70 - 8s - 121ms/step - acc: 0.6286 - loss: 0.6593 - val_acc: 0.7167 - val_loss: 0.5482 - learning_rate: 1.000
0e-05
Epoch 42/50
70/70 - 9s - 134ms/step - acc: 0.6429 - loss: 0.6450 - val_acc: 0.7000 - val_loss: 0.5383 - learning_rate: 1.000
0e-05
Epoch 43/50
70/70 - 9s - 134ms/step - acc: 0.5429 - loss: 0.6865 - val_acc: 0.7333 - val_loss: 0.5456 - learning_rate: 1.000
0e-05
Epoch 44/50
70/70 - 8s - 120ms/step - acc: 0.6714 - loss: 0.6193 - val_acc: 0.7167 - val_loss: 0.5233 - learning_rate: 1.000
0e-05
Epoch 45/50
70/70 - 10s - 136ms/step - acc: 0.6571 - loss: 0.6188 - val_acc: 0.7167 - val_loss: 0.5318 - learning_rate: 1.00
00e-05
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
Resultados guardados en 'results.csv'
Historial guardado en 'histories/relu_to_sigmoid_history.json'
Modelo guardado en 'models/relu_to_sigmoid_model.h5'
```

In [30]:

```
# config_name = "max_to_avg_pooling"
# cfg = models_config[config_name]

# run_single_experiment(
#     config_name=config_name,
#     cfg=cfg,
#     train_dataset=train_dataset,
#     validation_dataset=validation_dataset,
#     epochs=50,
#     results_path="results.csv",
#     history_dir="histories",
#     models_dir="models"
# )
```

```
=== Entrenando modelo 'max_to_avg_pooling' ===
```

Model: "3dcnn_max_to_avg_pooling"

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ input_layer (InputLayer)        │ (None, 128, 128, 64,   │             0 │
│                                 │ 1)                     │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d (Conv3D)                 │ (None, 128, 128, 64,   │         1,792 │
│                                 │ 64)                    │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ average_pooling3d               │ (None, 64, 64, 32, 64) │             0 │
│ (AveragePooling3D)              │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization             │ (None, 64, 64, 32, 64) │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_1 (Conv3D)               │ (None, 64, 64, 32, 64) │       110,656 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ average_pooling3d_1             │ (None, 32, 32, 16, 64) │             0 │
│ (AveragePooling3D)              │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_1           │ (None, 32, 32, 16, 64) │           256 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_2 (Conv3D)               │ (None, 32, 32, 16,     │       221,312 │
│                                 │ 128)                   │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ average_pooling3d_2             │ (None, 16, 16, 8, 128) │             0 │
│ (AveragePooling3D)              │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_2           │ (None, 16, 16, 8, 128) │           512 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv3d_3 (Conv3D)               │ (None, 16, 16, 8, 256) │       884,992 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ average_pooling3d_3             │ (None, 8, 8, 4, 256)   │             0 │
│ (AveragePooling3D)              │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ batch_normalization_3           │ (None, 8, 8, 4, 256)   │         1,024 │
│ (BatchNormalization)            │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ global_average_pooling3d        │ (None, 256)            │             0 │
│ (GlobalAveragePooling3D)        │                        │               │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 512)            │       131,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 1)              │           513 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 1,352,897 (5.16 MB)
 Trainable params: 1,351,873 (5.16 MB)
 Non-trainable params: 1,024 (4.00 KB)
Epoch 1/50
```

```
2024-12-24 16:59:59.650665: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warnin
g : Registers are spilled to local memory in function 'input_reduce_select_fusion_3', 88 bytes spill stores, 88
bytes spill loads
ptxas warning : Registers are spilled to local memory in function 'input_reduce_select_fusion_2', 8 bytes spill
stores, 8 bytes spill loads
ptxas warning : Registers are spilled to local memory in function 'input_reduce_reduce_window_fusion_3', 40 byte
s spill stores, 40 bytes spill loads
```

```
70/70 - 15s - 217ms/step - acc: 0.4929 - loss: 0.6961 - val_acc: 0.5000 - val_loss: 0.6983 - learning_rate: 1.00
00e-06
Epoch 2/50
70/70 - 9s - 130ms/step - acc: 0.5357 - loss: 0.6927 - val_acc: 0.5000 - val_loss: 0.7020 - learning_rate: 2.800
0e-06
Epoch 3/50
70/70 - 10s - 145ms/step - acc: 0.5929 - loss: 0.6830 - val_acc: 0.5000 - val_loss: 0.6945 - learning_rate: 4.60
00e-06
Epoch 4/50
70/70 - 10s - 145ms/step - acc: 0.6143 - loss: 0.6741 - val_acc: 0.5000 - val_loss: 0.7073 - learning_rate: 6.40
00e-06
Epoch 5/50
70/70 - 9s - 127ms/step - acc: 0.5571 - loss: 0.6879 - val_acc: 0.5000 - val_loss: 0.7097 - learning_rate: 8.200
0e-06
Epoch 6/50
70/70 - 10s - 146ms/step - acc: 0.6286 - loss: 0.6670 - val_acc: 0.4833 - val_loss: 0.6917 - learning_rate: 1.00
00e-05
Epoch 7/50
70/70 - 10s - 148ms/step - acc: 0.5714 - loss: 0.6853 - val_acc: 0.5000 - val_loss: 0.7159 - learning_rate: 1.00
00e-05
Epoch 8/50
70/70 - 9s - 129ms/step - acc: 0.6714 - loss: 0.6503 - val_acc: 0.5000 - val_loss: 0.8325 - learning_rate: 1.000
0e-05
Epoch 9/50
70/70 - 10s - 148ms/step - acc: 0.5571 - loss: 0.6942 - val_acc: 0.6167 - val_loss: 0.6558 - learning_rate: 1.00
00e-05
Epoch 10/50
70/70 - 10s - 148ms/step - acc: 0.6000 - loss: 0.6692 - val_acc: 0.6833 - val_loss: 0.6284 - learning_rate: 1.00
00e-05
Epoch 11/50
70/70 - 9s - 132ms/step - acc: 0.6143 - loss: 0.6724 - val_acc: 0.7333 - val_loss: 0.5969 - learning_rate: 1.000
0e-05
Epoch 12/50
70/70 - 10s - 145ms/step - acc: 0.5643 - loss: 0.6733 - val_acc: 0.8167 - val_loss: 0.6075 - learning_rate: 1.00
00e-05
Epoch 13/50
70/70 - 10s - 146ms/step - acc: 0.6214 - loss: 0.6659 - val_acc: 0.7333 - val_loss: 0.5535 - learning_rate: 1.00
00e-05
Epoch 14/50
70/70 - 9s - 129ms/step - acc: 0.6286 - loss: 0.6530 - val_acc: 0.7500 - val_loss: 0.5559 - learning_rate: 1.000
0e-05
Epoch 15/50
70/70 - 10s - 145ms/step - acc: 0.6143 - loss: 0.6602 - val_acc: 0.7333 - val_loss: 0.5576 - learning_rate: 1.00
00e-05
Epoch 16/50
70/70 - 10s - 144ms/step - acc: 0.5214 - loss: 0.6898 - val_acc: 0.7333 - val_loss: 0.5770 - learning_rate: 1.00
00e-05
Epoch 17/50
70/70 - 10s - 145ms/step - acc: 0.6643 - loss: 0.6439 - val_acc: 0.7167 - val_loss: 0.5486 - learning_rate: 1.00
00e-05
Epoch 18/50
70/70 - 9s - 131ms/step - acc: 0.6286 - loss: 0.6534 - val_acc: 0.7167 - val_loss: 0.5366 - learning_rate: 1.000
0e-05
Epoch 19/50
70/70 - 10s - 145ms/step - acc: 0.6429 - loss: 0.6422 - val_acc: 0.7167 - val_loss: 0.5380 - learning_rate: 1.00
00e-05
Epoch 20/50
70/70 - 10s - 146ms/step - acc: 0.6429 - loss: 0.6509 - val_acc: 0.6667 - val_loss: 0.5737 - learning_rate: 1.00
00e-05
Epoch 21/50
70/70 - 9s - 127ms/step - acc: 0.6500 - loss: 0.6593 - val_acc: 0.7000 - val_loss: 0.5526 - learning_rate: 1.000
0e-05
Epoch 22/50
70/70 - 10s - 144ms/step - acc: 0.5714 - loss: 0.6695 - val_acc: 0.6667 - val_loss: 0.5655 - learning_rate: 1.00
00e-05
Epoch 23/50
70/70 - 10s - 144ms/step - acc: 0.6643 - loss: 0.6619 - val_acc: 0.7000 - val_loss: 0.5729 - learning_rate: 1.00
00e-05
Epoch 24/50
70/70 - 9s - 128ms/step - acc: 0.6143 - loss: 0.6564 - val_acc: 0.6833 - val_loss: 0.6030 - learning_rate: 1.000
0e-05
Epoch 25/50
70/70 - 10s - 144ms/step - acc: 0.6214 - loss: 0.6513 - val_acc: 0.6667 - val_loss: 0.6070 - learning_rate: 1.00
00e-05
Epoch 26/50
70/70 - 10s - 144ms/step - acc: 0.6286 - loss: 0.6382 - val_acc: 0.7833 - val_loss: 0.5781 - learning_rate: 1.00
00e-05
Epoch 27/50
70/70 - 9s - 128ms/step - acc: 0.5786 - loss: 0.6575 - val_acc: 0.7167 - val_loss: 0.5571 - learning_rate: 1.000
0e-05
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
Resultados guardados en 'results.csv'
Historial guardado en 'histories/max_to_avg_pooling_history.json'
Modelo guardado en 'models/max_to_avg_pooling_model.h5'
```

In [15]:

```
# config_name = "residual_blocks"
# cfg = models_config[config_name]

# run_single_experiment(
#     config_name=config_name,
#     cfg=cfg,
#     train_dataset=train_dataset,
#     validation_dataset=validation_dataset,
#     epochs=50,
#     results_path="results.csv",
#     history_dir="histories",
#     models_dir="models"
# )
```

```
=== Entrenando modelo 'residual_blocks' ===
```

Model: "3dcnn_residual_blocks"

```
┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape      ┃    Param # ┃ Connected to      ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ input_layer         │ (None, 128, 128,  │          0 │ -                 │
│ (InputLayer)        │ 64, 1)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d (Conv3D)     │ (None, 128, 128,  │        896 │ input_layer[0][0] │
│                     │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalization │ (None, 128, 128,  │        128 │ conv3d[0][0]      │
│ (BatchNormalizatio… │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation          │ (None, 128, 128,  │          0 │ batch_normalizat… │
│ (Activation)        │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_2 (Conv3D)   │ (None, 128, 128,  │         64 │ input_layer[0][0] │
│                     │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_1 (Conv3D)   │ (None, 128, 128,  │     27,680 │ activation[0][0]  │
│                     │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │        128 │ conv3d_2[0][0]    │
│ (BatchNormalizatio… │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │        128 │ conv3d_1[0][0]    │
│ (BatchNormalizatio… │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ add (Add)           │ (None, 128, 128,  │          0 │ batch_normalizat… │
│                     │ 64, 32)           │            │ batch_normalizat… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_1        │ (None, 128, 128,  │          0 │ add[0][0]         │
│ (Activation)        │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ max_pooling3d       │ (None, 64, 64,    │          0 │ activation_1[0][… │
│ (MaxPooling3D)      │ 32, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 64, 64,    │        128 │ max_pooling3d[0]… │
│ (BatchNormalizatio… │ 32, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_3 (Conv3D)   │ (None, 64, 64,    │     55,360 │ batch_normalizat… │
│                     │ 32, 64)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 64, 64,    │        256 │ conv3d_3[0][0]    │
│ (BatchNormalizatio… │ 32, 64)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_2        │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (Activation)        │ 32, 64)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_5 (Conv3D)   │ (None, 64, 64,    │      2,112 │ batch_normalizat… │
│                     │ 32, 64)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_4 (Conv3D)   │ (None, 64, 64,    │    110,656 │ activation_2[0][… │
│                     │ 32, 64)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 64, 64,    │        256 │ conv3d_5[0][0]    │
│ (BatchNormalizatio… │ 32, 64)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 64, 64,    │        256 │ conv3d_4[0][0]    │
│ (BatchNormalizatio… │ 32, 64)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ add_1 (Add)         │ (None, 64, 64,    │          0 │ batch_normalizat… │
│                     │ 32, 64)           │            │ batch_normalizat… │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
```

```
│ activation_3          │ (None, 64, 64,        │            0 │ add_1[0][0]          │
│ (Activation)          │ 32, 64)               │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ max_pooling3d_1       │ (None, 32, 32,        │            0 │ activation_3[0][…    │
│ (MaxPooling3D)        │ 16, 64)               │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ batch_normalizatio…   │ (None, 32, 32,        │          256 │ max_pooling3d_1[…    │
│ (BatchNormalizatio…   │ 16, 64)               │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ conv3d_6 (Conv3D)     │ (None, 32, 32,        │      221,312 │ batch_normalizat…    │
│                       │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ batch_normalizatio…   │ (None, 32, 32,        │          512 │ conv3d_6[0][0]       │
│ (BatchNormalizatio…   │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ activation_4          │ (None, 32, 32,        │            0 │ batch_normalizat…    │
│ (Activation)          │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ conv3d_8 (Conv3D)     │ (None, 32, 32,        │        8,320 │ batch_normalizat…    │
│                       │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ conv3d_7 (Conv3D)     │ (None, 32, 32,        │      442,496 │ activation_4[0][…    │
│                       │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ batch_normalizatio…   │ (None, 32, 32,        │          512 │ conv3d_8[0][0]       │
│ (BatchNormalizatio…   │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ batch_normalizatio…   │ (None, 32, 32,        │          512 │ conv3d_7[0][0]       │
│ (BatchNormalizatio…   │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ add_2 (Add)           │ (None, 32, 32,        │            0 │ batch_normalizat…    │
│                       │ 16, 128)              │              │ batch_normalizat…    │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ activation_5          │ (None, 32, 32,        │            0 │ add_2[0][0]          │
│ (Activation)          │ 16, 128)              │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ max_pooling3d_2       │ (None, 16, 16, 8,     │            0 │ activation_5[0][…    │
│ (MaxPooling3D)        │ 128)                  │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ batch_normalizatio…   │ (None, 16, 16, 8,     │          512 │ max_pooling3d_2[…    │
│ (BatchNormalizatio…   │ 128)                  │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ global_average_poo…   │ (None, 128)           │            0 │ batch_normalizat…    │
│ (GlobalAveragePool…   │                       │              │                      │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ dense (Dense)         │ (None, 512)           │       66,048 │ global_average_p…    │
├───────────────────────┼───────────────────────┼──────────────┼──────────────────────┤
│ dense_1 (Dense)       │ (None, 1)             │          513 │ dense[0][0]          │
└───────────────────────┴───────────────────────┴──────────────┴──────────────────────┘
 Total params: 939,041 (3.58 MB)
 Trainable params: 937,249 (3.58 MB)
 Non-trainable params: 1,792 (7.00 KB)
Epoch 1/50
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1735056348.897209   30876 service.cc:148] XLA service 0x7fc878003f20 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
I0000 00:00:1735056348.897246   30876 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 4060, Compute Capability 8.9
2024-12-24 17:05:48.978756: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1735056349.352163   30876 cuda_dnn.cc:529] Loaded cuDNN version 90300
2024-12-24 17:05:49.810280: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warning : Registers are spilled to local memory in function 'gemm_fusion_dot_3500', 4 bytes spill stores, 4 bytes spill loads

E0000 00:00:1735056350.765990   30876 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735056350.860744   30876 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735056350.953668   30876 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735056356.841007   30876 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735056356.965337   30876 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
2024-12-24 17:06:00.457739: E external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm eng0{} for conv (f32[32,32,3,3,3]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,32,128,128,64]{4,3,2,1,0}, f32[2,32,128,128,64]{4,3,2,1,0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012->bf012, custom_call_target="__cudnn$convBackwardFilter", backend_config={"cudnn_conv_backend_config":{"activation_mode":"kNone","conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule":false,"operation_queue_id":"0","wait_on_operation_queues":[]} is taking a while...

2024-12-24 17:06:00.793588: E external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took 1.372615521s
Trying algorithm eng0{} for conv (f32[32,32,3,3,3]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[2,32,128,128,64]{4,3,2,1,0}, f32[2,32,128,128,64]{4,3,2,1,0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012->bf012, custom_call_target="__cudnn$convBackwardFilter", backend_config={"cudnn_conv_backend_config":{"activation_mode":"kNone","conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule":false,"operation_queue_id":"0","wait_on_operation_queues":[]} is taking a while...
2024-12-24 17:06:05.281364: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warning : Registers are spilled to local memory in function 'input_add_reduce_fusion_5', 12 bytes spill stores, 12 bytes spill loads
ptxas warning : Registers are spilled to local memory in function 'input_reduce_reduce_window_fusion_2', 36 bytes spill stores, 36 bytes spill loads

I0000 00:00:1735056365.300942   30876 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
```

```
70/70 - 35s - 494ms/step - acc: 0.6000 - loss: 0.6944 - val_acc: 0.5000 - val_loss: 0.7016 - learning_rate: 1.00
00e-06
Epoch 2/50
70/70 - 13s - 185ms/step - acc: 0.5214 - loss: 0.6949 - val_acc: 0.5000 - val_loss: 0.7134 - learning_rate: 2.80
00e-06
Epoch 3/50
70/70 - 14s - 201ms/step - acc: 0.5857 - loss: 0.6905 - val_acc: 0.5000 - val_loss: 0.7265 - learning_rate: 4.60
00e-06
Epoch 4/50
70/70 - 13s - 184ms/step - acc: 0.5857 - loss: 0.6853 - val_acc: 0.5000 - val_loss: 0.7681 - learning_rate: 6.40
00e-06
Epoch 5/50
70/70 - 14s - 201ms/step - acc: 0.5857 - loss: 0.6863 - val_acc: 0.5000 - val_loss: 0.8154 - learning_rate: 8.20
00e-06
Epoch 6/50
70/70 - 14s - 201ms/step - acc: 0.5786 - loss: 0.6829 - val_acc: 0.5000 - val_loss: 0.8656 - learning_rate: 1.00
00e-05
Epoch 7/50
70/70 - 13s - 183ms/step - acc: 0.6143 - loss: 0.6693 - val_acc: 0.5000 - val_loss: 0.8301 - learning_rate: 1.00
00e-05
Epoch 8/50
70/70 - 14s - 200ms/step - acc: 0.5857 - loss: 0.6750 - val_acc: 0.5000 - val_loss: 0.7557 - learning_rate: 1.00
00e-05
Epoch 9/50
70/70 - 14s - 202ms/step - acc: 0.5857 - loss: 0.6748 - val_acc: 0.5833 - val_loss: 0.6600 - learning_rate: 1.00
00e-05
Epoch 10/50
70/70 - 12s - 178ms/step - acc: 0.5786 - loss: 0.6778 - val_acc: 0.7333 - val_loss: 0.6216 - learning_rate: 1.00
00e-05
Epoch 11/50
70/70 - 14s - 202ms/step - acc: 0.6714 - loss: 0.6513 - val_acc: 0.6833 - val_loss: 0.6030 - learning_rate: 1.00
00e-05
Epoch 12/50
70/70 - 13s - 185ms/step - acc: 0.6143 - loss: 0.6509 - val_acc: 0.6667 - val_loss: 0.6101 - learning_rate: 1.00
00e-05
Epoch 13/50
70/70 - 14s - 201ms/step - acc: 0.6571 - loss: 0.6549 - val_acc: 0.6500 - val_loss: 0.6090 - learning_rate: 1.00
00e-05
Epoch 14/50

70/70 - 14s - 201ms/step - acc: 0.6143 - loss: 0.6794 - val_acc: 0.6333 - val_loss: 0.6123 - learning_rate: 1.00
00e-05
Epoch 15/50
70/70 - 13s - 186ms/step - acc: 0.5857 - loss: 0.6675 - val_acc: 0.6500 - val_loss: 0.5996 - learning_rate: 1.00
00e-05
Epoch 16/50
70/70 - 14s - 201ms/step - acc: 0.6429 - loss: 0.6482 - val_acc: 0.6167 - val_loss: 0.6667 - learning_rate: 1.00
00e-05
Epoch 17/50
70/70 - 13s - 185ms/step - acc: 0.6286 - loss: 0.6617 - val_acc: 0.6500 - val_loss: 0.6226 - learning_rate: 1.00
00e-05
Epoch 18/50
70/70 - 14s - 202ms/step - acc: 0.5857 - loss: 0.6670 - val_acc: 0.7000 - val_loss: 0.5895 - learning_rate: 1.00
00e-05
Epoch 19/50
70/70 - 13s - 185ms/step - acc: 0.6071 - loss: 0.6718 - val_acc: 0.7000 - val_loss: 0.5916 - learning_rate: 1.00
00e-05
Epoch 20/50
70/70 - 14s - 203ms/step - acc: 0.6643 - loss: 0.6392 - val_acc: 0.6667 - val_loss: 0.5763 - learning_rate: 1.00
00e-05
Epoch 21/50
70/70 - 13s - 186ms/step - acc: 0.6071 - loss: 0.6718 - val_acc: 0.7167 - val_loss: 0.5705 - learning_rate: 1.00
00e-05
Epoch 22/50
70/70 - 14s - 203ms/step - acc: 0.6357 - loss: 0.6629 - val_acc: 0.7167 - val_loss: 0.5642 - learning_rate: 1.00
00e-05
Epoch 23/50
70/70 - 14s - 203ms/step - acc: 0.6571 - loss: 0.6338 - val_acc: 0.7000 - val_loss: 0.5556 - learning_rate: 1.00
00e-05
Epoch 24/50
70/70 - 13s - 184ms/step - acc: 0.6000 - loss: 0.6637 - val_acc: 0.7167 - val_loss: 0.5634 - learning_rate: 1.00
00e-05
Epoch 25/50
70/70 - 14s - 202ms/step - acc: 0.6571 - loss: 0.6304 - val_acc: 0.6667 - val_loss: 0.5537 - learning_rate: 1.00
00e-05
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
Resultados guardados en 'results.csv'
Historial guardado en 'histories/residual_blocks_history.json'
Modelo guardado en 'models/residual_blocks_model.h5'
```

In [15]:

```
# config_name = "inception_blocks"
# cfg = models_config[config_name]
# run_single_experiment(
#     config_name=config_name,
#     cfg=cfg,
#     train_dataset=train_dataset,
#     validation_dataset=validation_dataset,
#     epochs=50,
#     results_path="results.csv",
#     history_dir="histories",
#     models_dir="models"
# )
```

```
=== Entrenando modelo 'inception_blocks' ===
Model: "3dcnn_inception_blocks"

┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━┓
┃ Layer (type)        ┃ Output Shape      ┃    Param # ┃ Connected to      ┃
┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━┩
│ input_layer         │ (None, 128, 128,  │          0 │ -                 │
│ (InputLayer)        │ 64, 1)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_1 (Conv3D)   │ (None, 128, 128,  │         16 │ input_layer[0][0] │
│                     │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_3 (Conv3D)   │ (None, 128, 128,  │         16 │ input_layer[0][0] │
│                     │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │         32 │ conv3d_1[0][0]    │
│ (BatchNormalizatio… │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │         32 │ conv3d_3[0][0]    │
│ (BatchNormalizatio… │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_1        │ (None, 128, 128,  │          0 │ batch_normalizat… │
│ (Activation)        │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_3        │ (None, 128, 128,  │          0 │ batch_normalizat… │
│ (Activation)        │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ max_pooling3d       │ (None, 128, 128,  │          0 │ input_layer[0][0] │
│ (MaxPooling3D)      │ 64, 1)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d (Conv3D)     │ (None, 128, 128,  │         16 │ input_layer[0][0] │
│                     │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_2 (Conv3D)   │ (None, 128, 128,  │      1,736 │ activation_1[0][… │
│                     │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_4 (Conv3D)   │ (None, 128, 128,  │      8,008 │ activation_3[0][… │
│                     │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_5 (Conv3D)   │ (None, 128, 128,  │         16 │ max_pooling3d[0]… │
│                     │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalization │ (None, 128, 128,  │         32 │ conv3d[0][0]      │
│ (BatchNormalizatio… │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │         32 │ conv3d_2[0][0]    │
│ (BatchNormalizatio… │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │         32 │ conv3d_4[0][0]    │
│ (BatchNormalizatio… │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │         32 │ conv3d_5[0][0]    │
│ (BatchNormalizatio… │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_6 (Conv3D)   │ (None, 128, 128,  │         64 │ input_layer[0][0] │
│                     │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation          │ (None, 128, 128,  │          0 │ batch_normalizat… │
│ (Activation)        │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_2        │ (None, 128, 128,  │          0 │ batch_normalizat… │
│ (Activation)        │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_4        │ (None, 128, 128,  │          0 │ batch_normalizat… │
│ (Activation)        │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_5        │ (None, 128, 128,  │          0 │ batch_normalizat… │
│ (Activation)        │ 64, 8)            │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio… │ (None, 128, 128,  │        128 │ conv3d_6[0][0]    │
│ (BatchNormalizatio… │ 64, 32)           │            │                   │
├─────────────────────┼───────────────────┼────────────┼───────────────────┤
```

```
│ concatenate          │ (None, 128, 128,  │          0 │ activation[0][0], │
│ (Concatenate)        │ 64, 32)           │            │ activation_2[0][… │
│                      │                   │            │ activation_4[0][… │
│                      │                   │            │ activation_5[0][… │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ add (Add)            │ (None, 128, 128,  │          0 │ batch_normalizat… │
│                      │ 64, 32)           │            │ concatenate[0][0] │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_6         │ (None, 128, 128,  │          0 │ add[0][0]         │
│ (Activation)         │ 64, 32)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ max_pooling3d_1      │ (None, 64, 64,    │          0 │ activation_6[0][… │
│ (MaxPooling3D)       │ 32, 32)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │        128 │ max_pooling3d_1[… │
│ (BatchNormalizatio…  │ 32, 32)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_8 (Conv3D)    │ (None, 64, 64,    │        528 │ batch_normalizat… │
│                      │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_10 (Conv3D)   │ (None, 64, 64,    │        528 │ batch_normalizat… │
│                      │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │         64 │ conv3d_8[0][0]    │
│ (BatchNormalizatio…  │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │         64 │ conv3d_10[0][0]   │
│ (BatchNormalizatio…  │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_8         │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (Activation)         │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_10        │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (Activation)         │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ max_pooling3d_2      │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (MaxPooling3D)       │ 32, 32)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_7 (Conv3D)    │ (None, 64, 64,    │        528 │ batch_normalizat… │
│                      │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_9 (Conv3D)    │ (None, 64, 64,    │      6,928 │ activation_8[0][… │
│                      │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_11 (Conv3D)   │ (None, 64, 64,    │     32,016 │ activation_10[0]… │
│                      │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_12 (Conv3D)   │ (None, 64, 64,    │        528 │ max_pooling3d_2[… │
│                      │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │         64 │ conv3d_7[0][0]    │
│ (BatchNormalizatio…  │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │         64 │ conv3d_9[0][0]    │
│ (BatchNormalizatio…  │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │         64 │ conv3d_11[0][0]   │
│ (BatchNormalizatio…  │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │         64 │ conv3d_12[0][0]   │
│ (BatchNormalizatio…  │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ conv3d_13 (Conv3D)   │ (None, 64, 64,    │      2,112 │ batch_normalizat… │
│                      │ 32, 64)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_7         │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (Activation)         │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_9         │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (Activation)         │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_11        │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (Activation)         │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ activation_12        │ (None, 64, 64,    │          0 │ batch_normalizat… │
│ (Activation)         │ 32, 16)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ batch_normalizatio…  │ (None, 64, 64,    │        256 │ conv3d_13[0][0]   │
│ (BatchNormalizatio…  │ 32, 64)           │            │                   │
├──────────────────────┼───────────────────┼────────────┼───────────────────┤
│ concatenate_1        │ (None, 64, 64,    │          0 │ activation_7[0][… │
│ (Concatenate)        │ 32, 64)           │            │ activation_9[0][… │
│                      │                   │            │ activation_11[0]… │
│                      │                    │             │ activation_12[0]…  │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ add_1 (Add)          │ (None, 64, 64,     │           0 │ batch_normalizat…  │
│                      │ 32, 64)            │             │ concatenate_1[0]…  │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ activation_13        │ (None, 64, 64,     │           0 │ add_1[0][0]        │
│ (Activation)         │ 32, 64)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ max_pooling3d_3      │ (None, 32, 32,     │           0 │ activation_13[0]…  │
│ (MaxPooling3D)       │ 16, 64)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         256 │ max_pooling3d_3[…  │
│ (BatchNormalizatio…  │ 16, 64)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ conv3d_15 (Conv3D)   │ (None, 32, 32,     │       2,080 │ batch_normalizat…  │
│                      │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ conv3d_17 (Conv3D)   │ (None, 32, 32,     │       2,080 │ batch_normalizat…  │
│                      │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         128 │ conv3d_15[0][0]    │
│ (BatchNormalizatio…  │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         128 │ conv3d_17[0][0]    │
│ (BatchNormalizatio…  │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ activation_15        │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│ (Activation)         │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ activation_17        │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│ (Activation)         │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ max_pooling3d_4      │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│ (MaxPooling3D)       │ 16, 64)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ conv3d_14 (Conv3D)   │ (None, 32, 32,     │       2,080 │ batch_normalizat…  │
│                      │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ conv3d_16 (Conv3D)   │ (None, 32, 32,     │      27,680 │ activation_15[0]…  │
│                      │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ conv3d_18 (Conv3D)   │ (None, 32, 32,     │     128,032 │ activation_17[0]…  │
│                      │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ conv3d_19 (Conv3D)   │ (None, 32, 32,     │       2,080 │ max_pooling3d_4[…  │
│                      │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         128 │ conv3d_14[0][0]    │
│ (BatchNormalizatio…  │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         128 │ conv3d_16[0][0]    │
│ (BatchNormalizatio…  │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         128 │ conv3d_18[0][0]    │
│ (BatchNormalizatio…  │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         128 │ conv3d_19[0][0]    │
│ (BatchNormalizatio…  │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ conv3d_20 (Conv3D)   │ (None, 32, 32,     │       8,320 │ batch_normalizat…  │
│                      │ 16, 128)           │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ activation_14        │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│ (Activation)         │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ activation_16        │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│ (Activation)         │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ activation_18        │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│ (Activation)         │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ activation_19        │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│ (Activation)         │ 16, 32)            │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ batch_normalizatio…  │ (None, 32, 32,     │         512 │ conv3d_20[0][0]    │
│ (BatchNormalizatio…  │ 16, 128)           │             │                    │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ concatenate_2        │ (None, 32, 32,     │           0 │ activation_14[0]…  │
│ (Concatenate)        │ 16, 128)           │             │ activation_16[0]…  │
│                      │                    │             │ activation_18[0]…  │
│                      │                    │             │ activation_19[0]…  │
├──────────────────────┼────────────────────┼─────────────┼────────────────────┤
│ add_2 (Add)          │ (None, 32, 32,     │           0 │ batch_normalizat…  │
│                       │ 16, 128)             │              │ concatenate_2[0]… │
├───────────────────────┼──────────────────────┼──────────────┼───────────────────┤
│ activation_20         │ (None, 32, 32,       │            0 │ add_2[0][0]       │
│ (Activation)          │ 16, 128)             │              │                   │
├───────────────────────┼──────────────────────┼──────────────┼───────────────────┤
│ max_pooling3d_5       │ (None, 16, 16, 8,    │            0 │ activation_20[0]… │
│ (MaxPooling3D)        │ 128)                 │              │                   │
├───────────────────────┼──────────────────────┼──────────────┼───────────────────┤
│ batch_normalizatio…   │ (None, 16, 16, 8,    │          512 │ max_pooling3d_5[… │
│ (BatchNormalizatio…   │ 128)                 │              │                   │
├───────────────────────┼──────────────────────┼──────────────┼───────────────────┤
│ global_average_poo…   │ (None, 128)          │            0 │ batch_normalizat… │
│ (GlobalAveragePool…   │                      │              │                   │
├───────────────────────┼──────────────────────┼──────────────┼───────────────────┤
│ dense (Dense)         │ (None, 512)          │       66,048 │ global_average_p… │
├───────────────────────┼──────────────────────┼──────────────┼───────────────────┤
│ dense_1 (Dense)       │ (None, 1)            │          513 │ dense[0][0]       │
└───────────────────────┴──────────────────────┴──────────────┴───────────────────┘
```

```
 Total params: 295,089 (1.13 MB)
 Trainable params: 293,521 (1.12 MB)
 Non-trainable params: 1,568 (6.12 KB)
Epoch 1/50
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1735056750.525103   32993 service.cc:148] XLA service 0x7f8f8c0029a0 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
I0000 00:00:1735056750.525164   32993 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 4060, Compute Capability 8.9
2024-12-24 17:12:30.693137: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1735056751.453684   32993 cuda_dnn.cc:529] Loaded cuDNN version 90300
2024-12-24 17:12:32.167337: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warning : Registers are spilled to local memory in function 'gemm_fusion_dot_7214', 4 bytes spill stores, 4 bytes spill loads

E0000 00:00:1735056758.070992   32993 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735056758.178965   32993 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735056759.646778   32993 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735056759.743866   32993 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal accuracy. There may be a missing warmup execution, please investigate in Nsight Systems.
2024-12-24 17:12:56.956482: I external/local_xla/xla/stream_executor/cuda/cuda_asm_compiler.cc:397] ptxas warning : Registers are spilled to local memory in function 'input_add_reduce_fusion_10', 8 bytes spill stores, 8 bytes spill loads
ptxas warning : Registers are spilled to local memory in function 'input_add_reduce_fusion_9', 4 bytes spill stores, 4 bytes spill loads
ptxas warning : Registers are spilled to local memory in function 'input_add_reduce_fusion_8', 24 bytes spill stores, 24 bytes spill loads

I0000 00:00:1735056777.003568   32993 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the process.
```

```
70/70 - 47s - 668ms/step - acc: 0.4571 - loss: 0.6982 - val_acc: 0.5000 - val_loss: 0.7165 - learning_rate: 1.00
00e-06
Epoch 2/50
70/70 - 14s - 193ms/step - acc: 0.4857 - loss: 0.6946 - val_acc: 0.5000 - val_loss: 0.7214 - learning_rate: 2.80
00e-06
Epoch 3/50
70/70 - 12s - 178ms/step - acc: 0.5500 - loss: 0.6924 - val_acc: 0.5167 - val_loss: 0.7002 - learning_rate: 4.60
00e-06
Epoch 4/50
70/70 - 13s - 193ms/step - acc: 0.6071 - loss: 0.6881 - val_acc: 0.4833 - val_loss: 0.7054 - learning_rate: 6.40
00e-06
Epoch 5/50
70/70 - 12s - 175ms/step - acc: 0.5286 - loss: 0.6976 - val_acc: 0.5000 - val_loss: 0.7206 - learning_rate: 8.20
00e-06
Epoch 6/50
70/70 - 13s - 192ms/step - acc: 0.6214 - loss: 0.6852 - val_acc: 0.5000 - val_loss: 0.7193 - learning_rate: 1.00
00e-05
Epoch 7/50
70/70 - 12s - 177ms/step - acc: 0.5500 - loss: 0.6847 - val_acc: 0.5000 - val_loss: 0.6978 - learning_rate: 1.00
00e-05
Epoch 8/50
70/70 - 14s - 194ms/step - acc: 0.6143 - loss: 0.6801 - val_acc: 0.5667 - val_loss: 0.6771 - learning_rate: 1.00
00e-05
Epoch 9/50
70/70 - 14s - 195ms/step - acc: 0.6000 - loss: 0.6816 - val_acc: 0.7333 - val_loss: 0.6659 - learning_rate: 1.00
00e-05
Epoch 10/50
70/70 - 12s - 178ms/step - acc: 0.6429 - loss: 0.6727 - val_acc: 0.6833 - val_loss: 0.6557 - learning_rate: 1.00
00e-05

Epoch 11/50
70/70 - 14s - 195ms/step - acc: 0.5571 - loss: 0.6779 - val_acc: 0.7000 - val_loss: 0.6497 - learning_rate: 1.00
00e-05
Epoch 12/50
70/70 - 12s - 178ms/step - acc: 0.5857 - loss: 0.6844 - val_acc: 0.7000 - val_loss: 0.6436 - learning_rate: 1.00
00e-05
Epoch 13/50
70/70 - 14s - 195ms/step - acc: 0.6286 - loss: 0.6700 - val_acc: 0.7000 - val_loss: 0.6382 - learning_rate: 1.00
00e-05
Epoch 14/50
70/70 - 14s - 196ms/step - acc: 0.6214 - loss: 0.6709 - val_acc: 0.6833 - val_loss: 0.6336 - learning_rate: 1.00
00e-05
Epoch 15/50
70/70 - 12s - 177ms/step - acc: 0.5571 - loss: 0.6795 - val_acc: 0.7333 - val_loss: 0.6332 - learning_rate: 1.00
00e-05
Epoch 16/50
70/70 - 14s - 195ms/step - acc: 0.6000 - loss: 0.6752 - val_acc: 0.6833 - val_loss: 0.6306 - learning_rate: 1.00
00e-05
Epoch 17/50
70/70 - 12s - 177ms/step - acc: 0.5643 - loss: 0.6866 - val_acc: 0.6833 - val_loss: 0.6263 - learning_rate: 1.00
00e-05
Epoch 18/50
70/70 - 14s - 195ms/step - acc: 0.6714 - loss: 0.6642 - val_acc: 0.7000 - val_loss: 0.6208 - learning_rate: 1.00
00e-05
Epoch 19/50
70/70 - 12s - 178ms/step - acc: 0.6000 - loss: 0.6664 - val_acc: 0.7000 - val_loss: 0.6173 - learning_rate: 1.00
00e-05
Epoch 20/50
70/70 - 14s - 195ms/step - acc: 0.6000 - loss: 0.6578 - val_acc: 0.6833 - val_loss: 0.6134 - learning_rate: 1.00
00e-05
Epoch 21/50
70/70 - 14s - 195ms/step - acc: 0.6143 - loss: 0.6640 - val_acc: 0.7000 - val_loss: 0.6115 - learning_rate: 1.00
00e-05
Epoch 22/50
70/70 - 12s - 178ms/step - acc: 0.6143 - loss: 0.6613 - val_acc: 0.6833 - val_loss: 0.6089 - learning_rate: 1.00
00e-05
Epoch 23/50
70/70 - 14s - 195ms/step - acc: 0.5857 - loss: 0.6635 - val_acc: 0.7000 - val_loss: 0.6073 - learning_rate: 1.00
00e-05
Epoch 24/50
70/70 - 12s - 178ms/step - acc: 0.5571 - loss: 0.6797 - val_acc: 0.7000 - val_loss: 0.6069 - learning_rate: 1.00
00e-05
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. T
his file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_m
odel.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
```

```
Resultados guardados en 'results.csv'
Historial guardado en 'histories/inception_blocks_history.json'
Modelo guardado en 'models/inception_blocks_model.h5'
```

Results

Once the models are trained, we proceed to analyze their performance by visualizing accuracy and loss for both the training and validation datasets across epochs. This helps us understand how the models learn over time and identify potential issues such as overfitting or underfitting.

Additionally, we use the best-performing model to make predictions on new or unseen data. This final step evaluates the model's practical utility and provides insights into its real-world performance.

Functions

In [12]:

```python
def plot_training_history(history, config_name):
    fig, ax = plt.subplots(1, 2, figsize=(20, 5))
    ax = ax.ravel()

    for i, metric in enumerate(["acc", "loss"]):
        ax[i].plot(history[metric])
        ax[i].plot(history["val_" + metric])
        ax[i].set_title(f"{config_name} - {metric.capitalize()}")
        ax[i].set_xlabel("Epochs")
        ax[i].set_ylabel(metric.capitalize())
        ax[i].legend(["Train", "Validation"])
        ax[i].grid(True)

    plt.tight_layout()
    plt.show()
```

In [13]:

```python
def make_predictions(model, x_val = x_val, class_names = ["normal", "abnormal"]):
    predictions = model.predict(np.expand_dims(x_val[0], axis=0))[0]
    scores = [1 - predictions[0], predictions[0]]

    for score, name in zip(scores, class_names):
        print(
            f"This model is {100 * score:.2f}% confident that the CT scan is {name}."
        )
```

Performance

Best model

As a result of all the training processes, we consolidate the performance metrics of each configuration into a summary table. This table is loaded from the `results.csv` file and displays key information, such as the model configuration, a brief description, and the final training and validation accuracy. By comparing these results, we can identify the best-performing configuration and gain insights into the effectiveness of each architectural modification.

In [14]:

```python
df_results = pd.read_csv("results.csv")
df_results
```

Out[14]:

	config_name	description	final_train_acc	final_val_acc
0	original	Modelo base sin modificaciones	0.692857	0.750000
1	extra_dense	Añade capa Dense de 100 unidades antes de la s...	0.657143	0.566667
2	two_conv_per_level	Agrega una segunda capa Conv3D en cada bloque	0.814286	0.766667
3	relu_to_sigmoid	Cambia la activación ReLU por sigmoid en todas...	0.657143	0.716667
4	max_to_avg_pooling	Usa AveragePooling3D en lugar de MaxPooling3D	0.578571	0.716667
5	residual_blocks	Reemplaza capas conv por bloques residuales	0.657143	0.666667
6	inception_blocks	Reemplaza capas conv por bloques inception 3D	0.557143	0.700000

As shown, the best model (the one that generalizes best) is identified by selecting the configuration with the highest validation accuracy. Below are the details of the best-performing model:

- **Configuration Name**: `two_conv_per_level`
- **Description**: Adds a second Conv3D layer at each block.
- **Final Training Accuracy**: 81.43%
- **Final Validation Accuracy**: 76.67%

In [15]:

```python
best_model_row = df_results[df_results['final_val_acc'] == df_results['final_val_acc'].max()].iloc[0]

print("Mejor modelo:\n")
for key, value in best_model_row.items():
    print(f"{key}: {value}")
```

```
Mejor modelo:

config_name: two_conv_per_level
description: Agrega una segunda capa Conv3D en cada bloque
final_train_acc: 0.8142856955528259
final_val_acc: 0.7666666507720947
```

Analysis of the stories

On the other hand, it is interesting to analyze the training dynamics of each configuration. For this purpose, we load and visualize the training history stored for each model. These visualizations allow us to observe the behavior of accuracy and loss over epochs for both the training and validation datasets.

To achieve this, we iterate through the configurations listed in the results table (`df_results["config_name"]`) and:

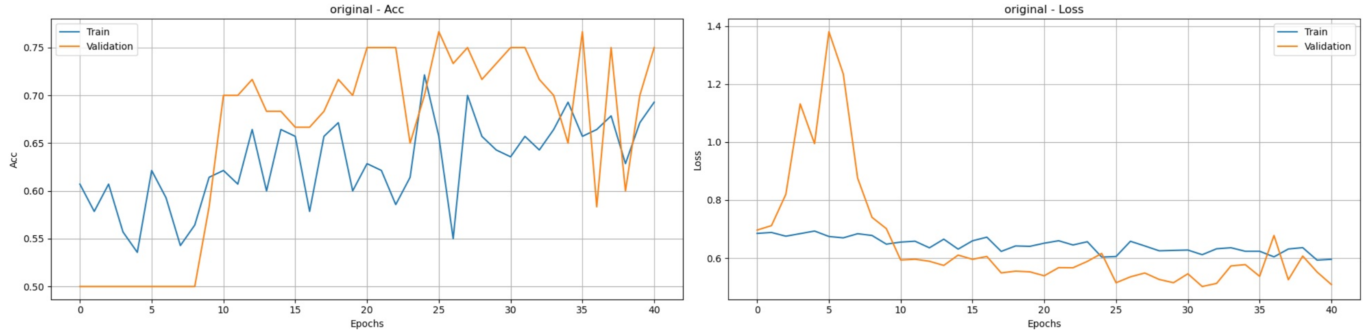
- Load the corresponding training history from the `histories` directory.
- For each configuration, plot the training and validation metrics using `plot_training_history` to provide a clear comparison.

These plots help identify trends such as convergence speed, overfitting, or underfitting, offering deeper insights into the strengths and weaknesses of each model.

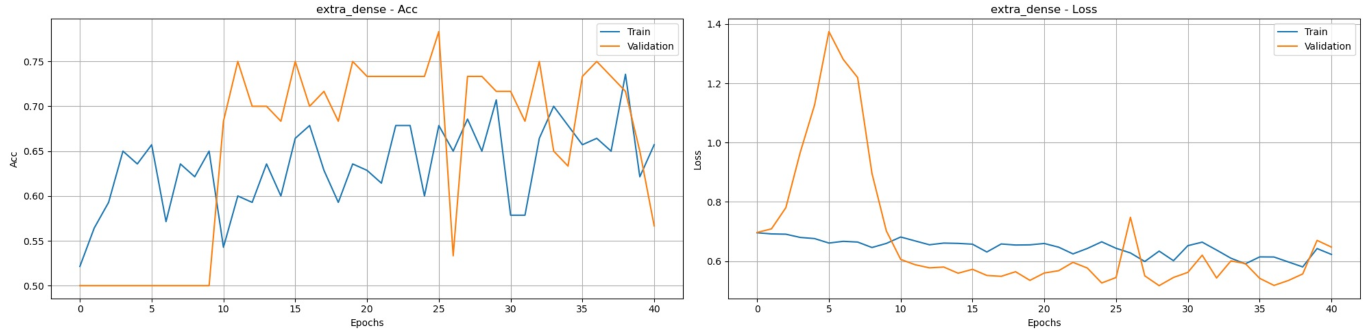
In [16]:

```python
history_dir = "histories"

for config_name in df_results["config_name"]:
    history_path = os.path.join(history_dir, f"{config_name}_history.json")
    if os.path.exists(history_path):
        with open(history_path, "r") as f:
            history_data = json.load(f)

        print(f"=== Gráficos para configuración: {config_name} ===")
        plot_training_history(history_data, config_name)
    else:
        print(f"Historial no encontrado para {config_name}: {history_path}")
```

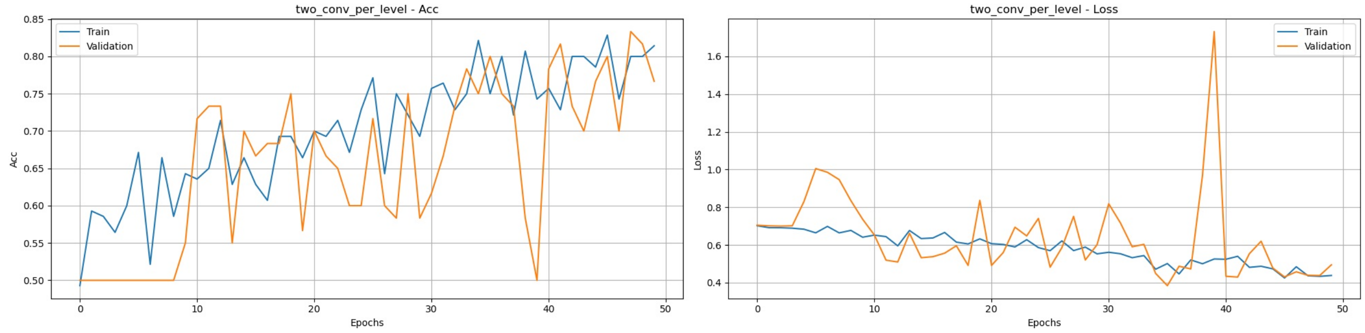
```
=== Gráficos para configuración: original ===
```



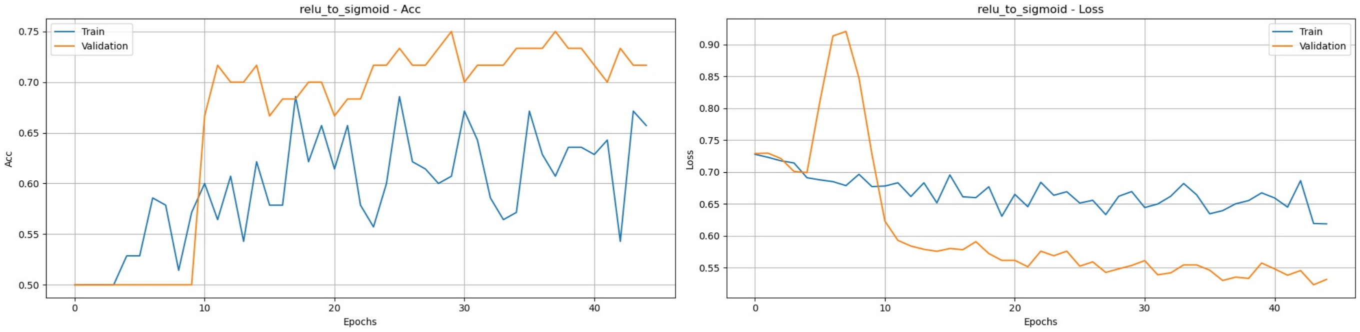
```
=== Gráficos para configuración: extra_dense ===
```



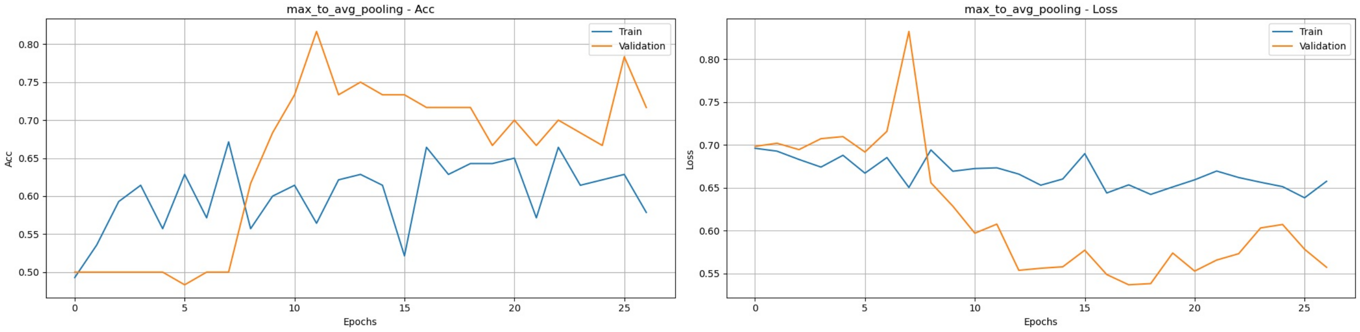
```
=== Gráficos para configuración: two_conv_per_level ===
```



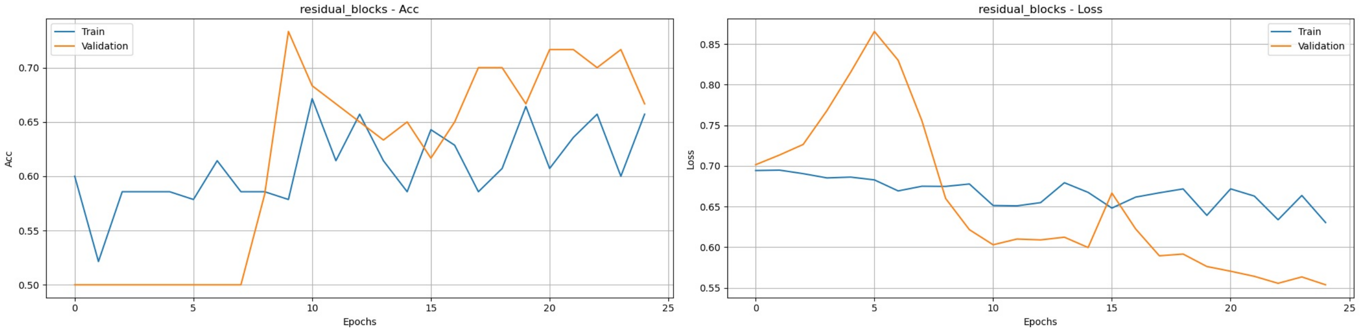
```
=== Gráficos para configuración: relu_to_sigmoid ===
```



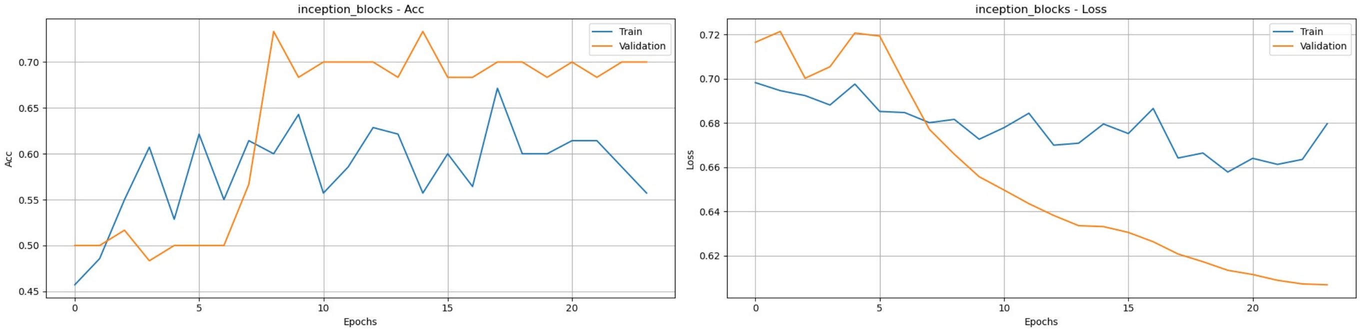
```
=== Gráficos para configuración: max_to_avg_pooling ===
```



```
=== Gráficos para configuración: residual_blocks ===
```



```
=== Gráficos para configuración: inception_blocks ===
```



General Observations from Training and Validation Plots

- **Fluctuations in Validation Accuracy**: Many models exhibit considerable fluctuations in validation accuracy across epochs. This behavior is often an indication of insufficient data, making it challenging for the model to generalize consistently.
- **Better Validation Accuracy than Training Accuracy**: For some configurations, validation accuracy surpasses training accuracy. While this might seem counterintuitive, it can occur due to data augmentation or dropout regularization, which increase variability in training but not in validation. However, it may also hint at overfitting to the validation set due to limited data.
- **Insufficient Data for Complex Models**: The complexity of some configurations (e.g., residual and inception blocks) seems excessive given the size of the dataset. This likely contributes to the observed instability in training and validation metrics, as these architectures require more data to fully leverage their capacity.

Best model predictions

Finally, we load the best-performing model, as identified from the results, and use it to make predictions:

In [19]:
```
best_model_name = best_model_row["config_name"]
```

```
models_dir = "models"
model_path = f"{models_dir}/{best_model_name}_model.h5"

best_model = load_model(model_path)
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` w
ill be empty until you train or evaluate the model.
```

```
In [20]: make_predictions(best_model)
```

```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1735057812.004002   35285 service.cc:148] XLA service 0x7fd17c013dc0 initialized for platform CUDA (
this does not guarantee that XLA will be used). Devices:
I0000 00:00:1735057812.004043   35285 service.cc:156]   StreamExecutor device (0): NVIDIA GeForce RTX 4060, Comp
ute Capability 8.9
2024-12-24 17:30:12.019426: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:268] disabling MLIR cr
ash reproducer, set env var `MLIR_CRASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1735057812.068356   35285 cuda_dnn.cc:529] Loaded cuDNN version 90300
E0000 00:00:1735057812.831935   35285 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal acc
uracy. There may be a missing warmup execution, please investigate in Nsight Systems.
E0000 00:00:1735057812.924528   35285 gpu_timer.cc:82] Delay kernel timed out: measured time has sub-optimal acc
uracy. There may be a missing warmup execution, please investigate in Nsight Systems.
2024-12-24 17:30:15.458282: E external/local_xla/xla/service/slow_operation_alarm.cc:65] Trying algorithm eng0{}
for conv (f32[1,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[1,64,128,128,64]{4,3,2,1,0}, f32[64,64,3,3,
3]{4,3,2,1,0}, f32[64]{0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012->bf012, custom_call_targ
et="__cudnn$convBiasActivationForward", backend_config={"cudnn_conv_backend_config":{"activation_mode":"kRelu","
conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule":false,"operation_queue_
id":"0","wait_on_operation_queues":[]} is taking a while...
2024-12-24 17:30:16.018423: E external/local_xla/xla/service/slow_operation_alarm.cc:133] The operation took 1.5
77133748s
Trying algorithm eng0{} for conv (f32[1,64,128,128,64]{4,3,2,1,0}, u8[0]{0}) custom-call(f32[1,64,128,128,64]{4,
3,2,1,0}, f32[64,64,3,3,3]{4,3,2,1,0}, f32[64]{0}), window={size=3x3x3 pad=1_1x1_1x1_1}, dim_labels=bf012_oi012-
>bf012, custom_call_target="__cudnn$convBiasActivationForward", backend_config={"cudnn_conv_backend_config":{"ac
tivation_mode":"kRelu","conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scale":0},"force_earliest_schedule"
:false,"operation_queue_id":"0","wait_on_operation_queues":[]} is taking a while...
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 5s 5s/step
This model is 6.82% confident that the CT scan is normal.
This model is 93.18% confident that the CT scan is abnormal.
```

```
I0000 00:00:1735057817.032426   35285 device_compiler.h:188] Compiled cluster using XLA!  This line is logged at
most once for the lifetime of the process.
```