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

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

```
from google.colab import drive

import os
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

import cv2

from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
```

```
path = '/content/drive/MyDrive/dataset_grao_arroz/'
classes = ['graos_inteiros', 'graos_quebrados']
img_size = 64
```

```
X = []
y = []

subpastas = ['train', 'test']

for idx, classe in enumerate(classes):
    total_classe = 0
    for sub in subpastas:
        pasta = os.path.join(path, classe, sub)
        print(f"Lendo: {pasta}")
        if not os.path.exists(pasta):
            print(f"Pasta não encontrada: {pasta}")
            continue
        arquivos = os.listdir(pasta)
        print(f" - Encontrados {len(arquivos)} arquivos em {sub}")
        for file in arquivos:
            img_path = os.path.join(pasta, file)
            img = cv2.imread(img_path)
            if img is None:
                print(f"Erro ao ler imagem: {img_path}")
                continue
            img = cv2.resize(img, (img_size, img_size))
            X.append(img)
            y.append(idx)
            total_classe += 1
    print(f"Total de imagens carregadas para a classe '{classe}': {total_classe}")

print(f"\nTotal geral de imagens carregadas: {len(X)}")
```

```
Lendo: /content/drive/MyDrive/dataset_grao_arroz/graos_inteiros/train
 - Encontrados 96 arquivos em train
Lendo: /content/drive/MyDrive/dataset_grao_arroz/graos_inteiros/test
 - Encontrados 22 arquivos em test
Total de imagens carregadas para a classe 'graos_inteiros': 118
Lendo: /content/drive/MyDrive/dataset_grao_arroz/graos_quebrados/train
 - Encontrados 58 arquivos em train
Lendo: /content/drive/MyDrive/dataset_grao_arroz/graos_quebrados/test
 - Encontrados 16 arquivos em test
Total de imagens carregadas para a classe 'graos_quebrados': 74

Total geral de imagens carregadas: 192
```

```
X = np.array(X) / 255.0
y = np.array(y)
```

```
x_train, x_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, stratify=y, random_state=42)
x_val, x_test, y_val, y_test = train_test_split(x_temp, y_temp, test_size=0.5, stratify=y_temp, random_s

print(f'Treino: {len(x_train)}, Validação: {len(x_val)}, Teste: {len(x_test)}')
```

```
Treino: 134, Validação: 29, Teste: 29
```

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)
datagen.fit(x_train)
```

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(img_size, img_size, 3)),
    MaxPooling2D(2,2),
    Dropout(0.25),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Dropout(0.25),

    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Dropout(0.3),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 62, 62, 32)	896
max_pooling2d_5 (MaxPooling2D)	(None, 31, 31, 32)	0
dropout_7 (Dropout)	(None, 31, 31, 32)	0
conv2d_6 (Conv2D)	(None, 29, 29, 64)	18,496
max_pooling2d_6 (MaxPooling2D)	(None, 14, 14, 64)	0
dropout_8 (Dropout)	(None, 14, 14, 64)	0
conv2d_7 (Conv2D)	(None, 12, 12, 128)	73,856
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_9 (Dropout)	(None, 6, 6, 128)	0
flatten_2 (Flatten)	(None, 4608)	0
dense_4 (Dense)	(None, 128)	589,952
dropout_10 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 1)	129

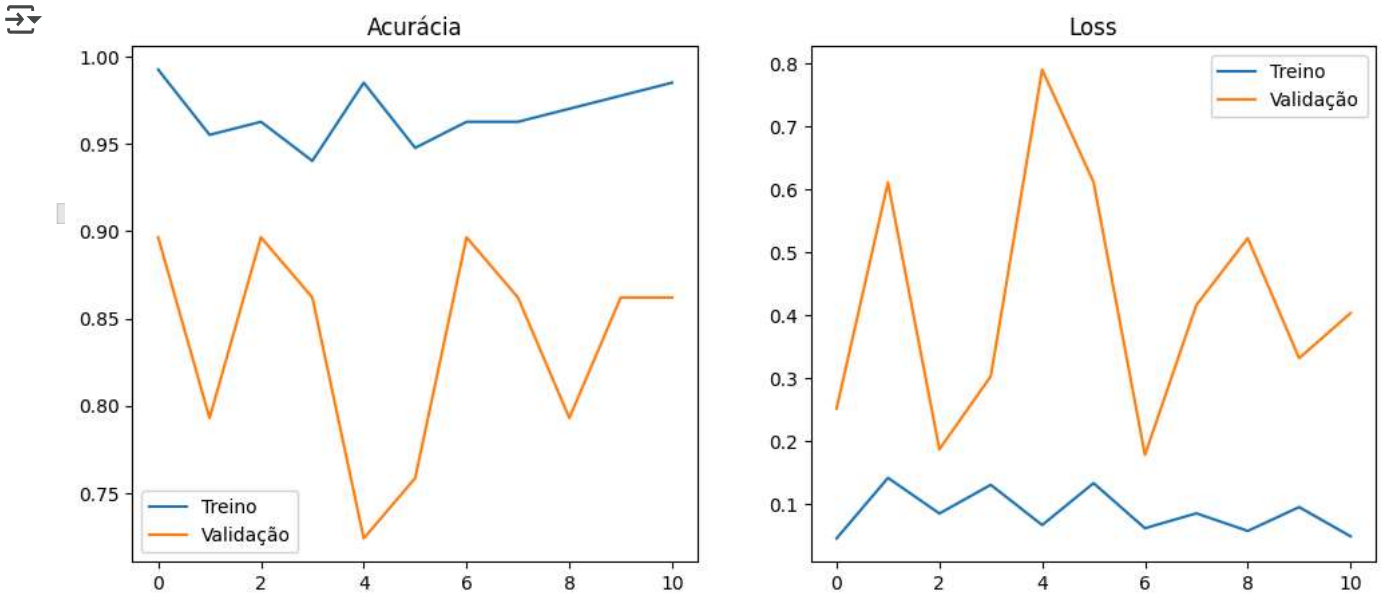
Total params: 683,329 (2.61 MB)
Trainable params: 683,329 (2.61 MB)
Non-trainable params: 0 (0.00 B)

```
early_stop = EarlyStopping(monitor='val_accuracy', patience=10, restore_best_weights=True)

history = model.fit(
    datagen.flow(x_train, y_train, batch_size=32),
    epochs=100,
    validation_data=(x_val, y_val),
    callbacks=[early_stop]
)
```

```
Epoch 1/100
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
  self._warn_if_super_not_called()
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 231ms/step - accuracy: 0.9909 - loss: 0.0480 - val_accuracy: 0.8966 - va
Epoch 2/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 239ms/step - accuracy: 0.9372 - loss: 0.1594 - val_accuracy: 0.7931 - va
Epoch 3/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 201ms/step - accuracy: 0.9516 - loss: 0.1034 - val_accuracy: 0.8966 - va
Epoch 4/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 217ms/step - accuracy: 0.9315 - loss: 0.1630 - val_accuracy: 0.8621 - va
Epoch 5/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 212ms/step - accuracy: 0.9798 - loss: 0.0690 - val_accuracy: 0.7241 - va
Epoch 6/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 257ms/step - accuracy: 0.9793 - loss: 0.0720 - val_accuracy: 0.7586 - va
Epoch 7/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 2s 468ms/step - accuracy: 0.9659 - loss: 0.0605 - val_accuracy: 0.8966 - va
Epoch 8/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 2s 242ms/step - accuracy: 0.9675 - loss: 0.0965 - val_accuracy: 0.8621 - va
Epoch 9/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 2s 214ms/step - accuracy: 0.9816 - loss: 0.0441 - val_accuracy: 0.7931 - va
Epoch 10/100
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 216ms/step - accuracy: 0.9741 - loss: 0.1160 - val_accuracy: 0.8621 - va
Epoch 11/100
```

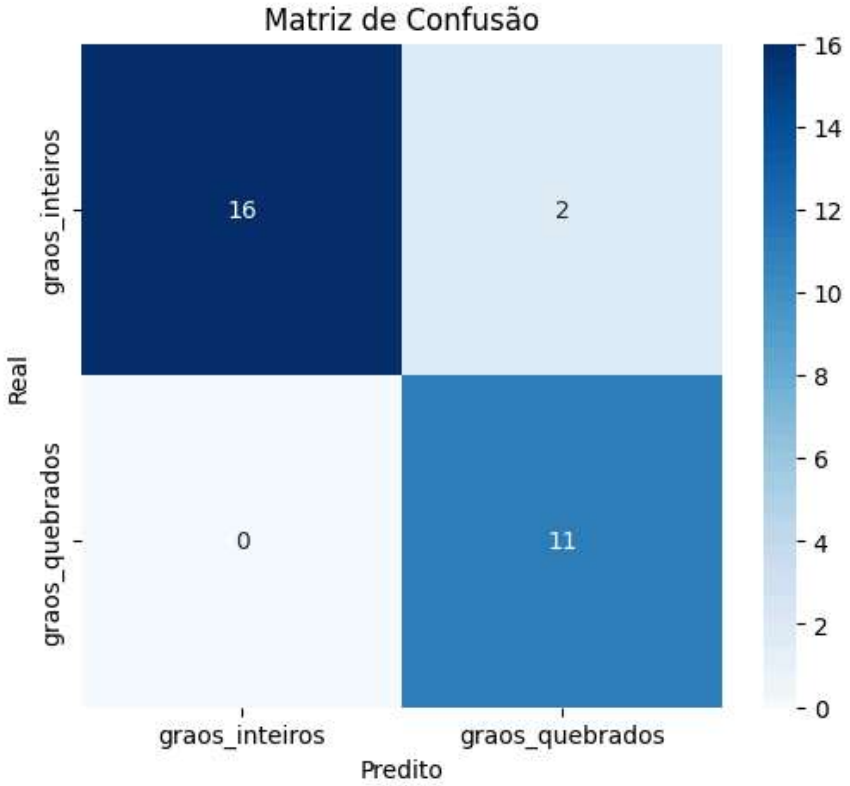
```
5/5 ━━━━━━━━━━━━━━━━━━━━ 1s 228ms/step - accuracy: 0.9779 - loss: 0.0577 - val_accuracy: 0.8621 - va
```

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Treino')
plt.plot(history.history['val_accuracy'], label='Validação')
plt.title('Acurácia')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Treino')
plt.plot(history.history['val_loss'], label='Validação')
plt.title('Loss')
plt.legend()
plt.show()
```




```
y_pred_prob = model.predict(x_test)
y_pred = (y_pred_prob > 0.5).astype(int)

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6,5))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', xticklabels=classes, yticklabels=classes)
plt.title("Matriz de Confusão")
plt.xlabel("Predito")
plt.ylabel("Real")
plt.show()

print(classification_report(y_test, y_pred, target_names=classes))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
```



	precision	recall	f1-score	support
graos_inteiros	1.00	0.89	0.94	18
graos_quebrados	0.85	1.00	0.92	11
accuracy			0.93	29
macro avg	0.92	0.94	0.93	29
weighted avg	0.94	0.93	0.93	29