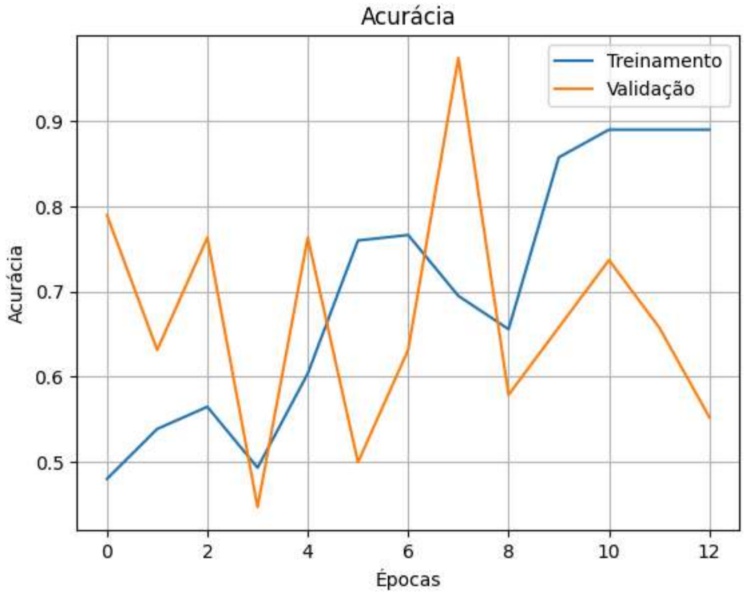
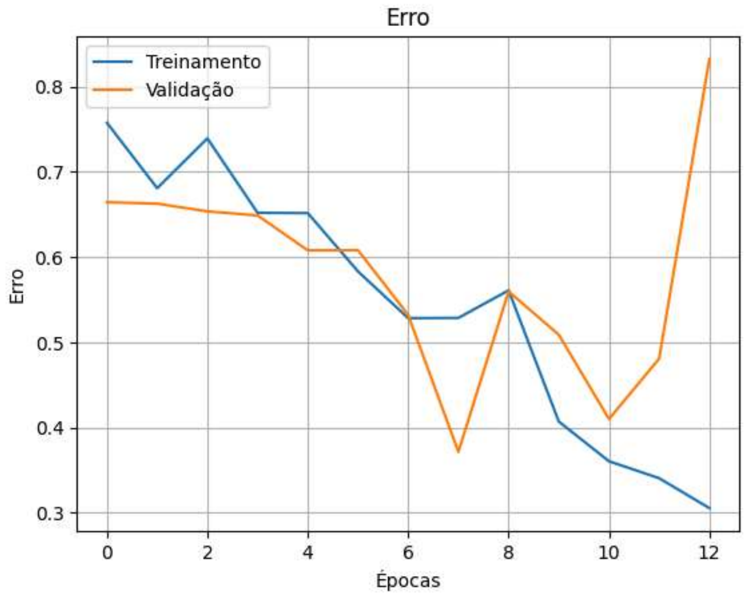
```python
from google.colab import drive
drive.mount('/content/drive')

import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, Input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

# Caminhos
train_dir = '/content/drive/MyDrive/arroz/Conjunto de dados de imagens de arroz/dataset/train'
test_dir = '/content/drive/MyDrive/arroz/Conjunto de dados de imagens de arroz/dataset/test'

# Geradores com data augmentation no treino
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

# Geradores
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(64, 64),
    batch_size=16,
    class_mode='binary',
    shuffle=True
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(64, 64),
    batch_size=16,
    class_mode='binary',
    shuffle=False
)

# Pesos das classes (inverso da frequência relativa)
from sklearn.utils.class_weight import compute_class_weight

classes = np.array([0, 1])  # 0: grao_quebrado, 1: graos_inteiros
weights = compute_class_weight(
    class_weight='balanced',
    classes=classes,
    y=train_generator.classes
)
class_weights = dict(zip(classes, weights))
print("Class weights:", class_weights)

# Modelo
model = Sequential([
    Input(shape=(64, 64, 3)),
    Conv2D(32, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Dropout(0.3),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Dropout(0.3),

    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.4),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer=Adam(learning_rate=0.0005),
              loss='binary_crossentropy',
              metrics=['accuracy'])

early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

```
# Treinamento
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=test_generator,
    class_weight=class_weights,
    callbacks=[early_stop]
)

# Avaliação
loss, acc = model.evaluate(test_generator)
print(f"\n🔎 Acurácia no teste: {acc:.4f}")

# Gráficos
plt.plot(history.history['accuracy'], label='Treinamento')
plt.plot(history.history['val_accuracy'], label='Validação')
plt.title('Acurácia')
plt.xlabel('Épocas')
plt.ylabel('Acurácia')
plt.legend()
plt.grid(True)
plt.show()

plt.plot(history.history['loss'], label='Treinamento')
plt.plot(history.history['val_loss'], label='Validação')
plt.title('Erro')
plt.xlabel('Épocas')
plt.ylabel('Erro')
plt.legend()
plt.grid(True)
plt.show()

# Avaliação detalhada
y_true = test_generator.classes
y_pred = (model.predict(test_generator) > 0.5).astype("int32").flatten()

print("\nMatriz de Confusão:")
cm = confusion_matrix(y_true, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=train_generator.class_indices, yticklabels=train_generator.class_indices)
plt.xlabel('Predito')
plt.ylabel('Real')
plt.show()

print("\nRelatório de Classificação:")
print(classification_report(y_true, y_pred, target_names=list(train_generator.class_indices.keys())))
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
Found 154 images belonging to 2 classes.
Found 38 images belonging to 2 classes.
Class weights: {np.int64(0): np.float64(1.3275862068965518), np.int64(1): np.float64(0.8020833333333334)}
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` c
  self._warn_if_super_not_called()
Epoch 1/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 189ms/step - accuracy: 0.5138 - loss: 0.7549 - val_accuracy: 0.7895 - val_loss: 0.6643
Epoch 2/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 156ms/step - accuracy: 0.5181 - loss: 0.6937 - val_accuracy: 0.6316 - val_loss: 0.6626
Epoch 3/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 153ms/step - accuracy: 0.5671 - loss: 0.6727 - val_accuracy: 0.7632 - val_loss: 0.6536
Epoch 4/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 249ms/step - accuracy: 0.5598 - loss: 0.6409 - val_accuracy: 0.4474 - val_loss: 0.6488
Epoch 5/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 4s 155ms/step - accuracy: 0.5682 - loss: 0.6683 - val_accuracy: 0.7632 - val_loss: 0.6079
Epoch 6/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 150ms/step - accuracy: 0.7921 - loss: 0.5828 - val_accuracy: 0.5000 - val_loss: 0.6080
Epoch 7/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 160ms/step - accuracy: 0.6846 - loss: 0.5457 - val_accuracy: 0.6316 - val_loss: 0.5321
Epoch 8/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 170ms/step - accuracy: 0.7598 - loss: 0.4908 - val_accuracy: 0.9737 - val_loss: 0.3715
Epoch 9/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 203ms/step - accuracy: 0.6322 - loss: 0.6015 - val_accuracy: 0.5789 - val_loss: 0.5596
Epoch 10/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 242ms/step - accuracy: 0.8157 - loss: 0.4182 - val_accuracy: 0.6579 - val_loss: 0.5089
Epoch 11/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 162ms/step - accuracy: 0.8774 - loss: 0.3548 - val_accuracy: 0.7368 - val_loss: 0.4100
Epoch 12/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 199ms/step - accuracy: 0.8674 - loss: 0.3827 - val_accuracy: 0.6579 - val_loss: 0.4809
Epoch 13/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 164ms/step - accuracy: 0.8953 - loss: 0.3044 - val_accuracy: 0.5526 - val_loss: 0.8324
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 54ms/step - accuracy: 0.9634 - loss: 0.3689

🔎 Acurácia no teste: 0.9737
```






```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step
```

Matriz de Confusão: