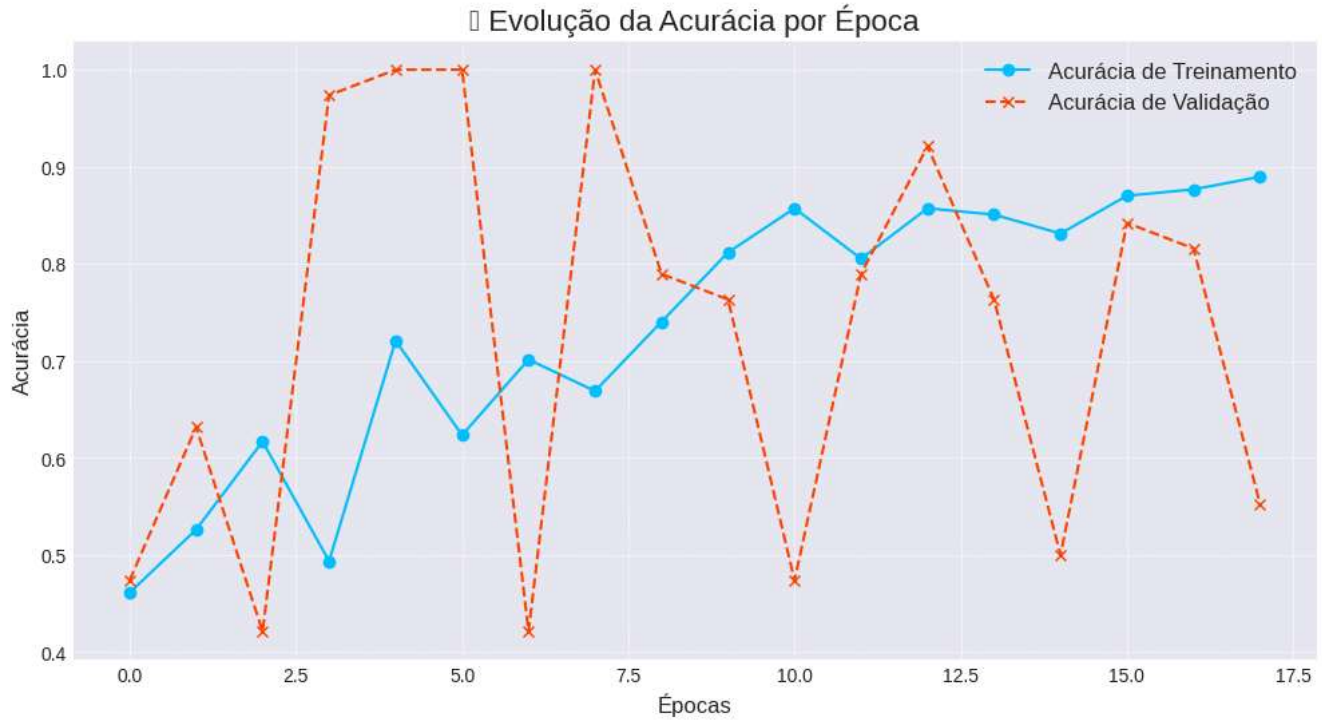
```
from google.colab import drive
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,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

drive.mount('/content/drive')

caminho_treino = '/content/drive/MyDrive/Arroz/Conjunto de dados de imagens de arroz/dataset/train'
caminho_teste = '/content/drive/MyDrive/Arroz/Conjunto de dados de imagens de arroz/dataset/test'

gerador_treino_com_aumento = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True
)

gerador_teste_sem_aumento = ImageDataGenerator(rescale=1./255)

iterador_treino = gerador_treino_com_aumento.flow_from_directory(
    caminho_treino,
    target_size=(64, 64),
    batch_size=16,
    class_mode='binary',
    shuffle=True
)

iterador_teste = gerador_teste_sem_aumento.flow_from_directory(
    caminho_teste,
    target_size=(64, 64),
    batch_size=16,
    class_mode='binary',
    shuffle=False
)

classes_array = np.array([0, 1])
pesos_calculados = compute_class_weight(
    class_weight='balanced',
    classes=classes_array,
    y=iterador_treino.classes
)
pesos_para_modelo = dict(zip(classes_array, pesos_calculados))
print("Pesos das classes:", pesos_para_modelo)

modelo_cnn = 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')
])

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

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

historico_treinamento = modelo_cnn.fit(
    iterador_treino,
    epochs=100,
    validation_data=iterador_teste,
    class_weight=pesos_para_modelo,
```

```
        callbacks=[parada_antecipada]
    )

    perda_teste, acuracia_teste = modelo_cnn.evaluate(iterador_teste)
    print(f"\n🔎 Acurácia final no conjunto de teste: {acuracia_teste:.4f}")

    plt.style.use('seaborn-v0_8-darkgrid')

    plt.figure(figsize=(12, 6))
    plt.plot(historico_treinamento.history['accuracy'], label='Acurácia de Treinamento', color='deepskyblue', marker='o', linestyle='-')
    plt.plot(historico_treinamento.history['val_accuracy'], label='Acurácia de Validação', color='orangered', marker='x', linestyle='--')
    plt.title('Evolução da Acurácia por Época', fontsize=16)
    plt.xlabel('Épocas', fontsize=12)
    plt.ylabel('Acurácia', fontsize=12)
    plt.legend(fontsize=12)
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.show()

    plt.figure(figsize=(12, 6))
    plt.plot(historico_treinamento.history['loss'], label='Perda de Treinamento', color='mediumvioletred', marker='o', linestyle='-')
    plt.plot(historico_treinamento.history['val_loss'], label='Perda de Validação', color='forestgreen', marker='x', linestyle='--')
    plt.title('Evolução da Perda (Erro) por Época', fontsize=16)
    plt.xlabel('Épocas', fontsize=12)
    plt.ylabel('Perda (Loss)', fontsize=12)
    plt.legend(fontsize=10)
    plt.grid(True, which='both', linestyle='--', linewidth=0.5)
    plt.show()

    rotulos_verdadeiros = iterador_teste.classes
    previsoes_prob = modelo_cnn.predict(iterador_teste)
    rotulos_previstos = (previsoes_prob > 0.5).astype("int32").flatten()

    print("\nMatriz de Confusão:")
    matriz_confusao = confusion_matrix(rotulos_verdadeiros, rotulos_previstos)

    plt.figure(figsize=(8, 6))
    sns.heatmap(matriz_confusao, annot=True, fmt='d', cmap='YlGnBu',
                xticklabels=iterador_treino.class_indices.keys(),
                yticklabels=iterador_treino.class_indices.keys())
    plt.title('📊 Matriz de Confusão', fontsize=15)
    plt.xlabel('Rótulo Previsto', fontsize=12)
    plt.ylabel('Rótulo Real', fontsize=12)
    plt.show()

    print("\nRelatório de Classificação Detalhado:")
    print(classification_report(rotulos_verdadeiros, rotulos_previstos, target_names=list(iterador_treino.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.
Pesos das classes: {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` cl
  self._warn_if_super_not_called()
Epoch 1/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - accuracy: 0.4094 - loss: 0.7409/usr/local/lib/python3.11/dist-packages/keras/src/trainers
  self._warn_if_super_not_called()
10/10 ━━━━━━━━━━━━━━━━━━━━ 5s 204ms/step - accuracy: 0.4141 - loss: 0.7395 - val_accuracy: 0.4737 - val_loss: 0.6785
Epoch 2/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 172ms/step - accuracy: 0.5154 - loss: 0.6773 - val_accuracy: 0.6316 - val_loss: 0.6658
Epoch 3/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 170ms/step - accuracy: 0.6302 - loss: 0.6377 - val_accuracy: 0.4211 - val_loss: 0.6692
Epoch 4/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 173ms/step - accuracy: 0.4766 - loss: 0.6779 - val_accuracy: 0.9737 - val_loss: 0.6320
Epoch 5/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 224ms/step - accuracy: 0.7577 - loss: 0.6480 - val_accuracy: 1.0000 - val_loss: 0.5918
Epoch 6/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 302ms/step - accuracy: 0.6208 - loss: 0.6395 - val_accuracy: 1.0000 - val_loss: 0.5398
Epoch 7/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 161ms/step - accuracy: 0.7479 - loss: 0.5814 - val_accuracy: 0.4211 - val_loss: 0.6889
Epoch 8/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 173ms/step - accuracy: 0.5653 - loss: 0.6222 - val_accuracy: 1.0000 - val_loss: 0.4653
Epoch 9/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 169ms/step - accuracy: 0.7511 - loss: 0.5302 - val_accuracy: 0.7895 - val_loss: 0.4631
Epoch 10/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 166ms/step - accuracy: 0.8331 - loss: 0.4688 - val_accuracy: 0.7632 - val_loss: 0.4454
Epoch 11/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 208ms/step - accuracy: 0.8662 - loss: 0.3997 - val_accuracy: 0.4737 - val_loss: 0.8852
Epoch 12/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 266ms/step - accuracy: 0.7934 - loss: 0.4213 - val_accuracy: 0.7895 - val_loss: 0.4234
Epoch 13/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 164ms/step - accuracy: 0.8273 - loss: 0.3604 - val_accuracy: 0.9211 - val_loss: 0.2861
Epoch 14/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 166ms/step - accuracy: 0.9036 - loss: 0.3027 - val_accuracy: 0.7632 - val_loss: 0.4201
Epoch 15/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 175ms/step - accuracy: 0.8675 - loss: 0.4033 - val_accuracy: 0.5000 - val_loss: 0.8832
Epoch 16/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 175ms/step - accuracy: 0.8517 - loss: 0.3332 - val_accuracy: 0.8421 - val_loss: 0.3532
Epoch 17/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 2s 179ms/step - accuracy: 0.8843 - loss: 0.3302 - val_accuracy: 0.8158 - val_loss: 0.3657
Epoch 18/100
10/10 ━━━━━━━━━━━━━━━━━━━━ 3s 255ms/step - accuracy: 0.8847 - loss: 0.2598 - val_accuracy: 0.5526 - val_loss: 0.9562
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.9605 - loss: 0.2135

🔎 Acurácia final no conjunto de teste: 0.9211
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128200 (\N{CHART WITH UPWARDS TREND}) mis
  fig.canvas.print_figure(bytes_io, **kw)
```




```
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128201 (\N{CHART WITH DOWNWARDS TREND}) m
  fig.canvas.print_figure(bytes_io, **kw)
```