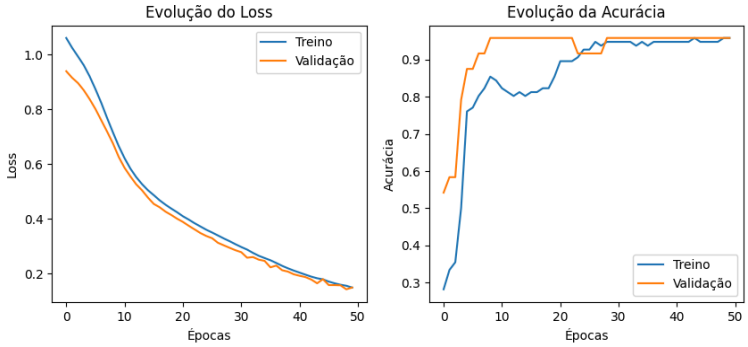
```python
# Passo 1: Importar bibliotecas e carregar dados
import tensorflow as tf
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Carregar o dataset Iris
iris = load_iris()
X = iris.data   # Características (sépalas e pétalas)
y = iris.target # Classes (0 = setosa, 1 = versicolor, 2 = virginica)

# Passo 2: Pré-processamento dos dados
# Dividir em treino (80%) e teste (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Normalizar os dados (muito importante para redes neurais)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Passo 3: Construir o modelo
# Criando uma rede neural simples com TensorFlow/Keras
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, activation="relu", input_shape=(4,)), # 1ª camada oculta (10 neurônios)
    tf.keras.layers.Dense(8, activation="relu"),                    # 2ª camada oculta
    tf.keras.layers.Dense(3, activation="softmax")                  # Saída (3 classes)
])

# Compilar o modelo (definir otimizador, função de perda e métrica)
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])

# Passo 4: Treinar o modelo
history = model.fit(X_train, y_train, epochs=50, batch_size=5, validation_split=0.2, verbose=1)
```

```python
# Passo 5: Avaliar o modelo
loss, accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Acurácia no conjunto de teste: {accuracy * 100:.2f}%")

# Passo 6: Fazer previsões
# Exemplo: prever a classe das primeiras 5 flores do conjunto de teste
predictions = model.predict(X_test[:5])
pred_classes = predictions.argmax(axis=1)

print("\nPrevisões:", pred_classes)
print("Classes reais:", y_test[:5])

# Passo 6: Fazer previsões (com nomes das flores)
# Prever as primeiras 5 flores do conjunto de teste
predictions = model.predict(X_test[:5])
pred_classes = predictions.argmax(axis=1)

# Converter números para nomes das flores
flower_names = iris.target_names

print("\nPrevisões:", [str(flower_names[i]) for i in pred_classes])
print("Classes reais:", [str(flower_names[i]) for i in y_test[:5]])

# Gráfico de Loss (função de perda)
plt.figure(figsize=(10,4))
plt.subplot(1,2,1)
plt.plot(history.history['loss'], label='Treino')
plt.plot(history.history['val_loss'], label='Validação')
plt.xlabel('Épocas')
plt.ylabel('Loss')
plt.title('Evolução do Loss')
plt.legend()
```

```python
# Gráfico de Accuracy (acurácia)
plt.subplot(1,2,2)
plt.plot(history.history['accuracy'], label='Treino')
plt.plot(history.history['val_accuracy'], label='Validação')
plt.xlabel('Épocas')
plt.ylabel('Acurácia')
plt.title('Evolução da Acurácia')
plt.legend()
plt.show()
```

```
Previsões: [1 0 2 1 1]
Classes reais: [1 0 2 1 1]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step

Previsões: ['versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor']
Classes reais: ['versicolor', 'setosa', 'virginica', 'versicolor', 'versicolor']
```



Execução do código:

- Bibliotecas importadas
- Dados carregados do dataset iris
- Feito o pré-processamento dos dados
- Modelo construído
- Modelo treinado e avaliado
- Feita as previsões
- Incluído gráfico de perda e acurácia