## ****1. Configuração do Ambiente****

O código foi desenvolvido para ser executado no Google Colab, utilizando o Google Drive para armazenamento dos dados. Certifique-se de ter uma conta no Google e acesso ao Google Drive.

### ****1.1 Montando o Google Drive****

Antes de carregar os dados, é necessário montar o Google Drive para acessar os arquivos CSV:

from google.colab import drive

drive.mount('/content/drive')

## ****2. Instalação das Dependências****

Certifique-se de que todas as bibliotecas necessárias estão instaladas. Execute o seguinte comando no Google Colab para instalar pacotes ausentes:

!pip install tensorflow pandas numpy seaborn

As principais bibliotecas utilizadas incluem:

* **pandas**: Manipulação de dados
* **numpy**: Cálculos matemáticos
* **tensorflow.keras**: Construção e treinamento da rede neural
* **seaborn/matplotlib**: Visualização dos resultados

## ****3. Carregamento dos Dados****

Os dados estão armazenados em arquivos CSV dentro do Google Drive, organizados em duas pastas:

* **Motores saudáveis**: /content/drive/MyDrive/motor\_dataset/saudavel
* **Motores com falha**: /content/drive/MyDrive/motor\_dataset/problemas

A função abaixo carrega os dados de cada pasta e os rotula corretamente:

import pandas as pd

import os

def load\_data(folder\_path, label):

data\_list = []

for file in os.listdir(folder\_path):

if file.endswith(".csv"):

df = pd.read\_csv(os.path.join(folder\_path, file))

df.columns = df.columns.str.strip()

if "Time Stamp" in df.columns:

df = df.drop(columns=["Time Stamp"])

df['label'] = label

data\_list.append(df)

return pd.concat(data\_list, ignore\_index=True)

path\_healthy = "/content/drive/MyDrive/motor\_dataset/saudavel"

path\_faulty = "/content/drive/MyDrive/motor\_dataset/problemas"

df\_healthy = load\_data(path\_healthy, label=0)

df\_faulty = load\_data(path\_faulty, label=1)

df = pd.concat([df\_healthy, df\_faulty], ignore\_index=True)

## ****4. Processamento dos Dados****

Separar os atributos (features) e rótulos (labels):

X = df[['Current-A', 'Current-B', 'Current-C']].values

y = df['label'].values

Dividir os dados em treino (70%) e teste (30%):

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, stratify=y, random\_state=42)

Ajustar formato para entrada da CNN 1D:

X\_train = X\_train.reshape(-1, 3, 1)

X\_test = X\_test.reshape(-1, 3, 1)

## ****5. Construção e Treinamento da Rede Neural****

O modelo é uma CNN 1D com as seguintes camadas:

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv1D, Flatten, Dense, Input, Dropout

from tensorflow.keras.regularizers import l2

model = Sequential([

Input(shape=(3, 1)),

Conv1D(filters=32, kernel\_size=2, activation='relu', kernel\_regularizer=l2(0.001)),

Dropout(0.5),

Flatten(),

Dense(64, activation='relu', kernel\_regularizer=l2(0.01)),

Dropout(0.3),

Dense(1, activation='sigmoid')

])

Compilar e treinar o modelo com **Early Stopping** para evitar overfitting:

from tensorflow.keras.callbacks import EarlyStopping

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

early\_stop = EarlyStopping(monitor='val\_loss', patience=10, restore\_best\_weights=True)

history = model.fit(

X\_train, y\_train,

epochs=50,

validation\_data=(X\_test, y\_test),

batch\_size=32,

callbacks=[early\_stop]

)

## ****6. Avaliação do Modelo****

Acurácia no conjunto de teste:

loss, accuracy = model.evaluate(X\_test, y\_test)

print(f"Acurácia no conjunto de teste: {accuracy:.2f}")

Criar a matriz de confusão para avaliar os erros do modelo:

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

y\_pred = (model.predict(X\_test) > 0.5).astype("int32")

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")

plt.xlabel("Previsto")

plt.ylabel("Real")

plt.title("Matriz de Confusão")

plt.show()

## ****7. Visualização do Treinamento****

Gráfico de perda:

plt.plot(history.history['loss'], label='Treino')

plt.plot(history.history['val\_loss'], label='Validação')

plt.xlabel('Épocas')

plt.ylabel('Loss')

plt.legend()

plt.title("Gráfico de Perda")

plt.show()

Gráfico de acurácia:

plt.plot(history.history['accuracy'], label='Treino')

plt.plot(history.history['val\_accuracy'], label='Validação')

plt.xlabel('Épocas')

plt.ylabel('Acurácia')

plt.legend()

plt.title("Gráfico de Acurácia")

plt.show()

## ****8. Teste com Novos Dados****

Para testar o modelo com um novo conjunto de falhas:

path\_new\_faulty = "/content/drive/MyDrive/motor\_dataset/validacao"

df\_new\_faulty = load\_data(path\_new\_faulty, label=1)

X\_new\_faulty = df\_new\_faulty[['Current-A', 'Current-B', 'Current-C']].values.reshape(-1, 3, 1)

loss\_new, acc\_new = model.evaluate(X\_new\_faulty, df\_new\_faulty['label'].values)

print(f"Acurácia nos novos dados de falha: {acc\_new:.2f}")