



Atividade – Redes Neurais em TensorFlow

Detecção de Falhas em Sinais de Sensores com Número Variável de Sensores

1 . Objetivo

Treinar, avaliar e explicar um modelo em **TensorFlow/Keras** que identifique falhas em sinais de séries temporais coletadas de sensores. O modelo **deve aceitar qualquer quantidade de sensores** sem ser re-treinado quando uma nova quantidade de sensores for adicionados ou removidos.

2 . Conjunto de dados

Coluna	Descrição
Timestamp	Marca temporal a 1 Hz
sensor_1 ... sensor_N	Leituras não normalizadas (N pode variar ≥ 1)
label	Nome dos sensores que apresentam problemas

Estatística global do dataset original (não extraível desse sample):

```
minimum = -37 969.1569,
maximum = 37 128.0924,
Q1 = -29.3986,
Q2 = 55.3385,
Q3 = 319.5190,
média = 31.9291,
percentis = 16 058.15, 16 346.23, 16 600.95, 16 771.44, 32 860.72,
samplingRate = 1 Hz.
```

Os dados são fornecidos já **divididos em múltiplos CSV** (“chunks”) que correspondem a uma parte de um dataset maior. Por isso, é importante usarem as estatísticas acima. Como todos os sensores são idênticos, as estatísticas são válidas para todos eles.

Cada linha do dataset se refere a coleta de todos os sensores em um determinado timestamp e cada coluna corresponde ao tempo da coleta, nome do sensor ou label.



SEMPRE+ QUALIDADE INovação INCLUSÃO

3 . Requisitos técnicos

1. **Framework** – TensorFlow 2.x / Keras (obrigatório).
2. **Divisão temporal** – 70 % treino, 15 % validação, 15 % teste. Por serem dados temporais, é PROIBIDO embaralhar na divisão dos dados.
3. **Janela temporal** – Determine `WINDOW` (ex.: 240 amostras - recomendado) e `STRIDE`; justifique. Quanto maior o stride, menos dados; quanto menor, mais janelas idênticas.
4. **Estratégia para Nº de Sensores Variável**

Escolha **UMA** das duas abordagens e implemente-a:

Opção	Descrição resumida	Quando preferir
A. Set Encoder com Multi-Head Attention (<i>independente de num_sensors</i>)	<ul style="list-style-type: none">Encoder compartilhado (CNN/MLP) gera vetor <code>h_i</code> por sensor<code>MultiHeadAttention</code> entre vetores + <code>GlobalAvgPool1D</code> → <code>Dense(1, sigmoid)</code>	Quando deseja capturar interações entre sensores na mesma janela
B. LSTM compartilhada – “um sensor por vez”	<ul style="list-style-type: none">Dataset de janelas por sensor (<code>WINDOW, 1</code>)Embaralhe janelas de sensores distintos no batchRede LSTM/CNN 1-D única com pesos compartilhadosInferência: realizada por sensor com dado ordenado temporalmente	Quando quer simplicidade e latência linear, sem lidar com attention. Nesse caso, se quiser capturar interação entre sensores (recomendado) terá que ser feito manualmente via engenharia de features (ex: não usar o sinal bruto, mas sim a diferença contra a média de todos os sensores a cada timestamp).

5. **Modelo** – Qualquer arquitetura 1-D compatível (CNN, LSTM, ou híbrida).
6. **Perdas e métricas** – `binary_crossentropy` (ou outras), `Precision`, `Recall`, `F1-Score`. Se falhas forem raras, use `class_weight` ou `tf.keras.losses.BinaryFocalCrossentropy` ou outros loss focados em desbalanceamento.
7. **Treinamento** – Use `EarlyStopping` (`patience ≥ 20`) e `ReduceLROnPlateau` ou outros schedulers.

4 . Entregas



SEMPRE +
QUALIDADE
INovação
INCLUSÃO

Item	Conteúdo mínimo
Notebook (.ipynb)	<ul style="list-style-type: none">• limpeza NÃO É NECESSÁRIA.• criação das janelas• preparação do <code>tf.data</code> com shuffle apenas no treino• construção do modelo (A ou B)• treinamento (logs claros)• avaliação em Val e Test (matriz de confusão, curvas)
Relatório (dentro do notebook)	<ul style="list-style-type: none">• objetivos, dados usados, estatísticas• decisão da janela e da abordagem de sensores variáveis• arquitetura detalhada + parâmetros• resultados + interpretação• limitações e ideias futuras
Primeira célula	Bibliotecas utilizadas. Você pode também criar o arquivo <code>environment.yml</code>

5 . Critérios de avaliação (10 pts)

Critério	Peso
Análise exploratória e justificativa de janela / divisão	1,0
Implementação correta da estratégia A ou B em TensorFlow	1,0
Qualidade da pipeline <code>tf.data</code> e tratamento de classe rara	1,0
Métricas e visualizações + interpretação	1,0
Organização, clareza, reproduzibilidade	1,0

6 . Templates de partida

Opção A – Set Encoder + Attention

```
WINDOW = 240
inp = tf.keras.Input(shape=(None, WINDOW), ragged=True)      # (sensores,
tempo)

def enc1d(x):
    x = tf.keras.layers.Conv1D(32, 5, activation='relu')(x)
    return tf.keras.layers.GlobalAvgPool1D()(x)                  # (32,)

enc = tf.keras.layers.TimeDistributed(
    tf.keras.layers.Lambda(enc1d))(inp)                          # (sensors,
32)
att = tf.keras.layers.MultiHeadAttention(4, 32)(enc, enc)
```



SEMPRE+ QUALIDADE INovação INCLUSÃO

Universidade Federal do Maranhão
Professor Dr. Thales Levi Azevedo Valente
Engenharia da Computação
ECPXXXX- Fundamentos de Redes Neurais

```
pool = tf.keras.layers.GlobalAveragePooling1D()(att)
out = tf.keras.layers.Dense(1, activation='sigmoid')(pool)

model = tf.keras.Model(inp, out)
```

Opção B – LSTM compartilhada (janela por sensor)

```
WINDOW = 240
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(self.window_size,
1)),
    tf.keras.layers.LSTM(64, return_sequences=True,
kernel_initializer=HeNormal()),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.LSTM(64, kernel_initializer=HeNormal(),
kernel_regularizer=l2(0.01)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(64, activation='relu',
kernel_initializer=HeNormal(), kernel_regularizer=l2(0.01)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(1, activation='sigmoid')
# Dataset: cada item = janela de um sensor; batches embaralham sensores
distintos
```

7 . Entrega final

1. Suba o repositório no GitHub ou GitLab.
2. Envie o link + notebook executado em PDF até <**data-limite**>.
3. Verifique que o `README.md` possui instruções de reprodução em **conda ou pip**.

Boa prática e bom trabalho!



EXTRA DO EXTRA

NEW Opção C – Espectrogramas ou Escalogramas (Válido somente para quem fez 1 das aorgagens anteriores)

Transformar o sinal 1-D em **espectrogramas** (ou escalogramas) agrega valor didático porque a representação tempo-frequência realça padrões não-estacionários que ficam invisíveis no domínio temporal puro [NatureMDPI](#). Uma vez convertidos em imagens, podemos aproveitar arquiteturas CNN já consolidadas na visão computacional, que costumam superar redes 1-D em tarefas de classificação de vibrações [MediumarXiv](#). Estudos de diagnóstico em rolamentos e máquinas rotativas mostram saltos de acurácia quando o modelo recebe espectrogramas em vez da série crua [SpringerLinkScienceDirect](#). Além disso, cada sensor gera sua própria imagem, de modo que o pipeline continua válido mesmo se o número de sensores crescer ou diminuir [ResearchGate](#). Por fim, técnicas como Grad-CAM aplicadas ao espectrograma facilitam a visualização das faixas de frequência que dispararam o alarme, reforçando a interpretabilidade para os especialistas [SpringerLinkMDPI](#).

1. Conversão do sinal

1. **Gerar espectrograma STFT** com `tf.signal.stft`, `scipy.signal.spectrogram` ou `librosa.feature.melspectrogram` [TensorFlow Documentação SciPy Librosa](#).
2. **Alternativa:** escalograma por **CWT** para frequências muito baixas [KagglePMC](#).
3. Escolher janela FFT (`n_fft`) e `hop_length` adequados para a amostragem (1 Hz no sample → considerar down-mix ou CWT) [enDAQ Blog](#).

2. Formação do dataset de imagens

- Cada janela → matriz (`freq_bins`, `time_bins`); normalize em dB ou [0, 1].
- Empilhe como canal único ou aplique colormap; salve em RAM como `tf.Tensor`.
- Crie `tf.data.Dataset` usando `tf.data.Dataset.from_tensor_slices` ou `tf.keras.utils.image_dataset_from_directory` [TensorFlow](#).

3. Modelo CNN sugerido

```
inp = tf.keras.Input(shape=(H, W, 1))
```



SEMPRE +
QUALIDADE
INovação
INCLUSÃO

Universidade Federal do Maranhão
Professor Dr. Thales Levi Azevedo Valente
Engenharia da Computação
ECPXXX- Fundamentos de Redes Neurais

```
x = tf.keras.layers.Conv2D(32, 3, activation='relu')(inp)  
x = tf.keras.layers.MaxPool2D(2)(x)  
x = tf.keras.layers.Conv2D(64, 3, activation='relu')(x)  
x = tf.keras.layers.GlobalAveragePooling2D()(x)  
out = tf.keras.layers.Dense(1, activation='sigmoid')(x)  
model = tf.keras.Model(inp, out)
```

- Ou transfer-learning (MobileNetV2, EfficientNet) congelando pesos iniciais [MediumWiley Online Library](#).
- Perda/métricas idênticas às opções A/B.

4. Vantagens & limitações

Pró

Captura **padrões espectrais** compartilhados entre sensores [ScienceDirectNature](#)

Aproveita **CNN 2-D madura** em TF — mesmo código de visão [TensorFlow](#)

Contra

Aumento de custo computacional;
requer cuidado com artefatos se $f_s = 1$ Hz.

Converte cada sensor
separadamente → inferência linear
no n.º de sensores.



Bibliotecas recomendadas

Foco	Biblioteca / Função
STFT / Mel	tf.signal.stft, tf.signal.linear_to_mel_weight_matrix, librosa.feature.melspectrogram TensorFlowLibrosa
Espectrograma clássico	scipy.signal.spectrogram Documentação SciPy



SEMPRE+ QUALIDADE INovação INCLUSÃO

Universidade Federal do Maranhão
Professor Dr. Thales Levi Azevedo Valente
Engenharia da Computação
ECPXXXX- Fundamentos de Redes Neurais

Foco	Biblioteca / Função
CWT / Escalograma	pywt.cwt ou notebook Kaggle de CWT em PyTorch adaptado p/ TF Kaggle
Visualização	matplotlib.pyplot.imshow ou tf.summary.image

Avaliação (atualizada)

Critério adicional para Opção C	Peso
Conversão correta p/ imagem + justificativa de parâmetros de TF	1,0
CNN 2-D em TensorFlow (arquitetura, regularização)	2,0

(Substitui o peso correspondente da implementação no quadro geral de 10 pts).

Fontes principais

- Vibration analysis via spectrogram + CNN [ScienceDirect](#)
- Nature 2025 study on image embeddings for vibration [Nature](#)
- TensorFlow audio tutorial (STFT, mel) [TensorFlow](#)
- SciPy spectrogram docs [Documentação SciPy](#)
- Librosa mel-spectrogram API [Librosa](#)
- Wavelet scalogram example [Kaggle](#)
- ImportChris guide to mel-spectrograms [Medium](#)
- Medium audio-classification with TF [Medium](#)
- ReductStore blog on vibration spectrograms [reduct.store](#)
- EnDAQ primer on FFT, PSD, spectrograms [enDAQ Blog](#)



SEMPRE+ QUALIDADE
INovação
INCLUSÃO

Universidade Federal do Maranhão
Professor Dr. Thales Levi Azevedo Valente
Engenharia da Computação
ECPXXXX- Fundamentos de Redes Neurais

Incorpore estes pontos à versão final do enunciado para que os alunos possam optar por processar **imagens de espectrograma** em vez de sinais 1-D, garantindo plena compatibilidade com variações no número de sensores.