Reconhecimento de Atividades Humanas utilizando Redes Neurais Recorrentes e sinais do acelerômetro de um *Smartphone* – 2021/2*

*Nota: Projeto da disciplina Tópicos Avançados de Aprendizagem de Máquina do 2o. Semestre de 2021

Aluno: Carlos Fonseca
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
henrique_fonseca@hotmail.com

Aluno: Willian Guerreiro
Programa de Pós Graduação em Eng. Elétrica
PPGEE/UFAM
Manaus/AM, Brasil
wguerreiro31@gmail.com

***Resumo*—As tecnologias para o reconhecimento de atividades humanas, por meio dos sensores de dispositivos móveis, possuem um vasto campo de aplicações, tais como o monitoramento de exercícios físicos, comando por gestos, quantificação da qualidade do sono, identificação de quedas, dentre outros. Neste trabalho, foram estudadas e implementadas arquiteturas de redes neurais recorrentes do tipo LSTM (*Long Short Term Memory*) e avaliado seu desempenho na tarefa de identificação de 09 padrões de atividades. Utilizou-se a base de dados pública UniMibSHAR[1] composta de 1724 observações, cada qual com 151 medições dos valores da aceleração nos 03 eixos x, y e z, obtidas por meio do acelerômetro de um *smartphone* Galaxy Nexus I9250. Dividindo-se o conjunto de dados para treinamento e teste, respectivamente em 85% e 15%, a arquitetura da rede LSTM proposta alcançou o máximo de 93,82% de acurácia no conjunto de teste.**

***Index Terms*—redes neurais recorrentes, LSTM, reconhecimento de atividades humanas.**

I. Introdução

Atualmente, uma parcela expressiva de idosos vivem sozinhos e, na ausência de familiares ou acompanhantes próximos para os auxiliarem, estão sujeitos a quedas e acidentes durante o desempenho de suas atividades diárias. Diante do crescente número de casos desses acidentes domésticos que, muitas vezes resultam em fraturas que levam à internação ou óbito, alguns hospitais de referência no Brasil, em parceria com as Universidades, estão concentrando esforços em pesquisar e desenvolver soluções acessíveis e de uso prático que ajudem a monitorar, remotamente, a atividade do idoso de forma que o mesmo possa ser assistido diante de uma situação de risco. Silva [2] propôs um trabalho voltado a esse tema e apresentou possíveis técnicas de Reconhecimento da Atividade Humana (HAR: *Human Activity Recognition*) para solucioná-lo.

HAR está se tornando um tópico promissor de pesquisa devido à facilidade de acesso aos sensores embarcados em dispositivos vestíveis. Mohammad [4] utilizou *smartphones* e *smartwatches*, conforme figura 1 [4], pois são econômicos, precisos e com baixo consumo de energia. Somado a isso, o avanço do poder computacional desses dispositivos tem permitido o uso Inteligência Artificial, mais especificamente, de modelos de redes neurais profundas, para detectar, interpretar e identificar movimentos humanos durante diferentes atividades, tais como andar, correr, saltar, comer, cair, deitar, sentar, etc.. Tal avanço tecnológico tem permitido o desenvolvimento de diversas aplicações úteis nas áreas de saúde, vigilância, *fitness*, casas inteligentes, dentre outras.

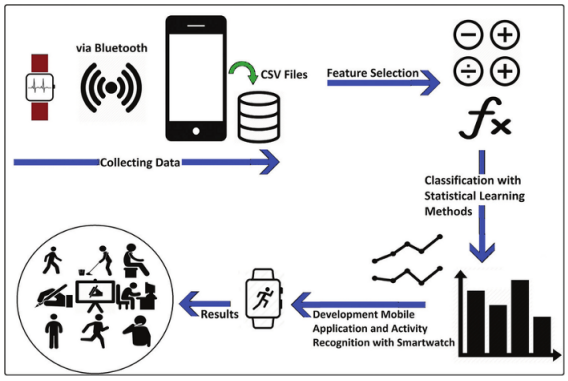

Figura 1. HAR utilizando *Smartphones* e *Smartwatches* [04]

Bragança [3], por sua vez, utilizou os sensores de um *smartphone* para mapear padrões de atividades humanas utilizando o algoritmo de representação simbólica HAR-SR. "O diferencial do método proposto, além da boa capacidade de generalização, está relacionado ao custo computacional, que é menor no processo de extração de características, visto que utiliza um número muito menor de características para gerar o modelo de classificação. Os resultados de avaliação usando três bases de dados reais (SHOAIB, UCI, WISDM) em seis cenários mostram que é possível classificar atividades com 93% de precisão" [3].

Neste trabalho, implementamos uma arquitetura de Redes Neurais Recorrentes (RNNs) do tipo LSTM (*Long Short Term Memory*) para treinar os padrões de atividades humanas a partir do conjunto de dados UniMibSHAR[1] composta de

1724 observações, cada qual com 151 medições dos valores da aceleração nos 03 eixos x, y e z, obtidas por meio do acelerômetro de um *smartphone* Galaxy Nexus I9250.

Na seção 2 será apresentada uma breve Fundamentação Teórica sobre redes neurais recorrentes e LSTMs. Na seção 3, a Metologia adotada para e implementação em Python e Matlab, na seção 4 os Resultados obtidos e, por fim, na seção 5 as Conclusões sobre a aplicação do método no problema proposto.

II. Fundamentação Teórica

Uma rede neural recorrente (RNN) é um tipo de rede neural artificial que utiliza dados sequenciais ou de série temporal. São comumente usados em tradução de textos, processamento de linguagem natural (nlp), reconhecimento de fala, legenda de imagens e reconhecimento da atividade humana. Enquanto as redes neurais profundas tradicionais presumem que as entradas e saídas são independentes umas das outras, a saída das redes neurais recorrentes depende dos elementos anteriores dentro da sequência.

A. LSTM

Publicada em 1997 por Sepp Hochreiter e Jürgen Schmidhuber [5] a rede neural recorrente do tipo LSTM (*Long Short-Term Memory*) é uma arquitetura própria para classificar, processar e prever séries temporais com intervalos de tempo de duração desconhecida. As LSTMs possuem células nas camadas ocultas da rede neural, munidas de três portas - uma porta de entrada (*Input Gate*), uma porta de saída (*Output Gate*) e uma porta de esquecimento (*Forget Gate*), conforme ilustrado na figura 2 [7]. Essas portas controlam o fluxo de informações necessárias para prever a saída na rede. A memória de longo prazo é geralmente chamada de estado da célula.

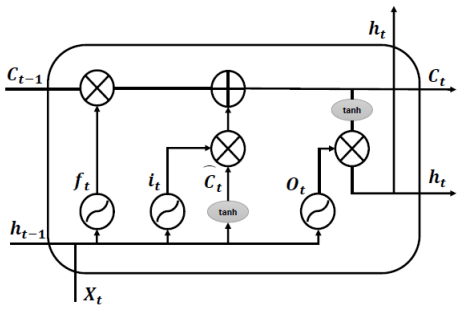

Figura 2. Rede Neural Recorrente tipo LSTM [7]

1) Forget Gate (f_t): possui a função de remover dos neurônios as informações que não são relevantes ao estado da célula. Isso é obtido pela equação 1, em que X_t (entrada no momento atual t) e h_{t-1} (saída de célula anterior) são alimentadas ao gate e multiplicadas por matrizes de peso, passando por uma função de ativação do tipo sigmóide que fornece uma saída binária. Se para um determinado estado de célula a saída for 0, a informação é esquecida e para a saída 1, a informação é retida para uso futuro.

$$f_t = \sigma(X_t W^f + h_{t-1} U^f) \tag{1}$$

2) Input Gate (i_t): A adição de informações úteis ao estado da célula é feita pelo input gate. Primeiro, a informação é regulada usando a função sigmoide que filtra os valores a serem lembrados de forma similar ao *forget gate* usando as entradas h_{t-1} e X_t conforme equação 2.

$$i_t = \sigma(X_t W^i + h_{t-1} U^i) \tag{2}$$

Em seguida, na equação 3, um vetor C'_t é criado usando a função $tanh$ que dá saída de -1 a +1, que contém todos os valores possíveis de h_{t-1} e x_t.

$$C'_t = \tanh\left(X_t W^c + h_{t-1} U^c\right) \tag{3}$$

A LSTM, então, atualiza os valores da célula C_t com base no estado anterior C_{t-1} e do vetor C'_t conforme equação 4:

$$C_t = \sigma(f_t \times C_{t-1} + i_t \times C'_t) \tag{4}$$

3) Output Gate (O_t): A tarefa de extrair informações úteis do estado da célula atual para serem apresentadas como uma saída é feita pelo *Output Gate*. Primeiro, a informação é regulada usando a função sigmóide que filtra os valores a serem lembrados usando as entradas $h_t - 1$ e X_t conforme equação 5.

$$O_t = \sigma(X_t W^o + h_{t-1} U^o) \tag{5}$$

Por último, um vetor é gerado aplicando a função $tanh$ na célula. Os valores do vetor e os valores regulados O_t são multiplicados para serem enviados como uma saída e entrada para a próxima célula h_t.

$$h_t = \tanh(C_t) \times O_t \tag{6}$$

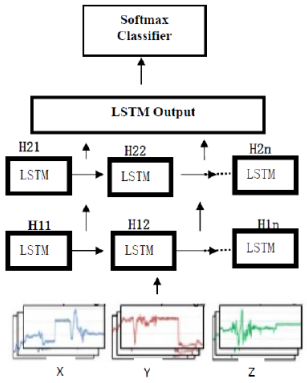

Figura 3. Estrutura de uma rede LSTM para HAR

Na figura 3, Zhong [6] apresenta a estrutura de um modelo de Reconhecimento de Atividades Humanas (HAR) baseado em LSTM cuja entrada é uma série temporal de aceleração 3D: X é a aceleração na direção x medida por acelerômetro do *smartphone*, Y é a direção y, Z é a direção z. Após as

camadas LTSM, o vetor de características passa por um multi classificador Softmax.

III. Metodologia

Nesta seção serão descritos os procedimentos metodológicos necessários à solução do problema proposto. Serão apresentadas as ferramentas de software, o método utilizado para identificação das atividades humanas com base em redes recorrentes, mais especificamente do tipo LSTM, a topologia de rede utilizada, bem como os parâmetros de simulação.

A. Ferramentas computacionais

A principal ferramenta utilizada neste trabalho foi o *Deep Learning Toolbox* do *software* MATLAB, que possui um conjunto de recursos voltados para o treinamento, projeto e simulação de redes recorrentes LSTM. A seguir são apresentadas as principais funções utilizadas neste trabalho.

1) cell: Cell array é um tipo de dado indexado cujo conteúdo é chamado de célula. Cada célula pode armazenar qualquer outro tipo de dado, entre vetores, matrizes e até mesmo outras células. Neste trabalho, vetores de células foram utilizados para o armazenamento dos lotes que representam cada movimento executado, onde cada lote é composto por 151 amostras de cada eixo do sensor acelerômetro.
2) randperm: Função utilizada para realizar a permutação de números inteiros de 1 a N, sendo N o parâmetro desta função. Para este trabalho, esta função foi utilizada para impor aleatoriedade ao conjunto de dados. A busca por uma base de treinamento mais representativa visa contribuir para que o modelo treinado generalize melhor e não sofra overfitting.
3) sequenceInputLayer: Como os dados de entrada se tratam de séries temporais, é necessário que a camada de entrada seja configurada para o modo sequencial.
4) lstmLayer: A camada LSTM (Long short-term memory) é inserida pela sua capacidade de aprender dependências de longo prazo entre a sequência temporal e as séries de dados. Para este trabalho esta camada foi parametrizada com o número de *hidden-layers* (blocos de memória), e o modo de saída como *last*, pois trata-se de um problema *sequence-label*, caso se tratasse de um problema *sequence-sequence*, o modo de saída seria parametrizado como *sequence*.
5) fullyConnectedLayer: A camada inteiramente conectada, responsável por multiplicar os dados na sua entrada por pesos e adicionar polarizações, foi parametrizada com o número de classes, isto é, foram reservados 9 neurônios para esta camada.
6) softmaxLayer: Penúltima camada, responsável por associar as respectivas ativações a um número que varia entre 0 e 1, significando, portanto, a probabilidade do elemento apresentado à rede pertencer à determinada classe.
7) classificationLayer: A partir da maior ativação da camada softmax, realiza a associação a determinada classe, significando portanto o resultado da inferência.
8) trainingOptions: Retorna um objeto com as definições dos parâmetros de treinamento, como função de perda, número de épocas, ambiente de execução (CPU ou GPU), e se o usuário deseja visualizar os gráficos de convergência.
9) trainNetwork: Através deste comando, parametrizado com os dados, as camadas e as opções de treinamento, é possível realizar o treinamento da rede de forma a obter os pesos e polarizações das respectivas camadas.
10) classify: Função utilizada para classificar os dados de teste utilizando a rede neural já treinada.
11) confusionmat: Função utilizada para realizar o cômputo da matriz de confusão, sendo parametrizada com o conjunto de teste e com a predição do modelo quando submetido a estes dados.
12) confusionchart: Função utilizada para exibir de maneira gráfica a matriz de confusão.

B. Descrição do problema proposto

A base de dados UniMibSHAR[1] é constituída por 1724 observações, sendo cada observação caracterizada por uma série temporal de 151 amostras coletadas sob mesma taxa, dos eixos x, y e z de um sensor acelerômetro. Os sinais contidos em cada observação descrevem atividades, as quais integram um grupo de 9, descritas a seguir:

1) Levantando a partir de um estado sentado numa cadeira
2) Levantando a partir de um estado deitado na cama
3) Caminhando
4) Correndo
5) Descendo uma escada
6) Pulando
7) Subindo uma escada
8) Deitando numa cama a partir de um estado em pé
9) Sentando numa cadeira estando em pé

Desta maneira, as séries de dados são submetidas à primeira camada de uma rede recorrente LSTM, cuja arquitetura será descrita nas próximas sessões. Com este modelo objetiva-se classificar corretamente as atividades catalogadas na base de dados. O modelo será treinado com 85% dos dados e validado com os 15% remanescentes. Para este processo de aprendizado de máquina serão avaliadas a taxa de convergência, a matriz de confusão e a acurácia do modelo.

C. Análise dos dados

Tendo em vista que é desejável um conjunto de treinamento representativo no qual as classes distribuem-se de maneira uniforme, foi realizada uma análise de distribuição de classes (Figura 4).

Para esta análise foi elaborado histograma para o conjunto de dados estudado. Nele é possível observar um desbalanceamento entre as classes, fator este que deve ser observado durante a divisão dos conjuntos de treinamento e teste.

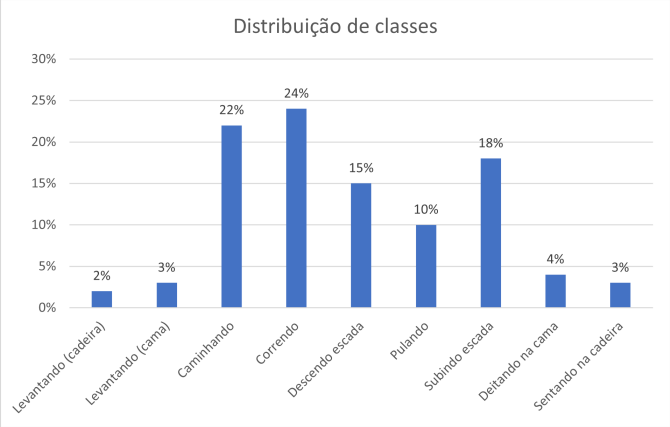

Figura 4. Distribuição de classes

D. Visualização dos dados

Para o correto entendimento da natureza dos dados e interpretação física do problema, foram elaborados gráficos para cada classe. A figura 5 ilustra a maneira como os sinais do sensor acelerômetro nos eixos x, y e z se relacionam de acordo com cada atividade executada.

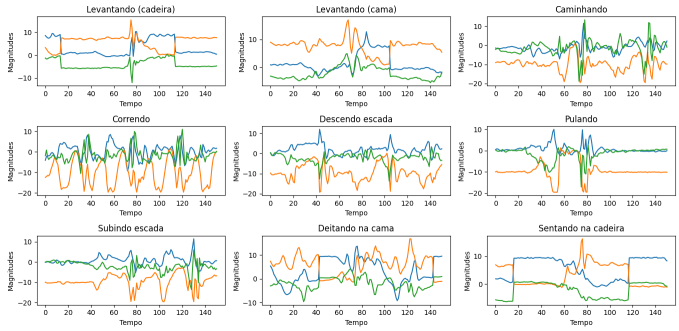

Figura 5. Amostras de aceleração

Através da análise da figura 5, pode-se verificar que o gráfico que representa o movimento "Correndo" (localizado na linha 2 e coluna 1 da figura), retrata de maneira visual o comportamento do eixo y (curva amarela), que alterna de forma cíclica durante a prática da corrida, isto é, de modo geral há variação vertical no movimento do corpo.

E. Implementação do modelo

Para a construção do modelo de aprendizado de máquina, foi desenvolvido um script na linguagem MATLAB, tendo o suporte do pacote de aprendizado profundo disponibilizado em sua biblioteca.

Inicialmente foi realiza a leitura do conjunto de dados, após isto as observações foram reordenadas randomicamente e por fim os dados foram particionados. Esta divisão foi feita de forma que 85% dos dados foram destinados à pasta de treinamento.

Após a definição dos conjuntos de treino e teste, a arquitetura da rede foi definida. Para a realização do fluxo de aprendizado de máquina, foram seguidos os seguintes passos:

- Definição das camadas: Nesta aplicação foram utilizadas as camadas de entrada sequencial, LSTM, camada inteiramente conectada e função softmax na última camada, para fins de classificação.
- Escolha da função de otimização: Através da função "trainningOptions" foi possível especificar o método "ADAM" como otimizador da função de custo.
- Início do processo de treinamento e escolha do melhor modelo com base na acurácia ao realizar a inferência no conjunto de teste.

F. Arquitetura da rede utilizada

Alguns parâmetros se fizeram necessários na definição da arquitetura da rede recorrente LSTM. Foi consultada a topologia padrão expressa em [8] durante a construção do algoritmo. A figura 6 ilustra os principais elementos de construção da rede, a citar: *Number of Hidden Units*, *Number of Features* e *Number of Time Steps*.

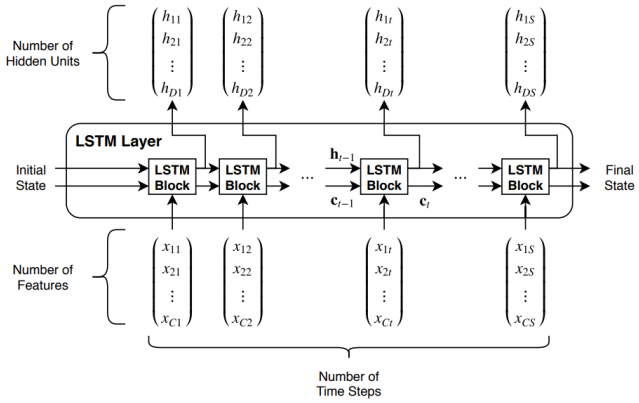

Figura 6. Arquitetura da camada de rede LSTM

Para o problema em questão, tem-se que o número de *features* ou características é igual a 3, uma vez que se tratam dos 3 eixos do sensor acelerômetro. O número de amostras temporais ou *time steps* é igual a 151, pela definição da janela de dados previamente definida. Entretanto, o número de *Hidden Units* foi tomado de maneira experimental, assumindo os valores 100, 200 e 250. O critério de escolha do número de estados escondidos ou *Hidden Units* será apresentado na sessão de resultados.

A figura 7 ilustra as camadas utilizadas no modelo. Tem-se que o modelo pressupõe em sua entrada um pacote com as dimensões 1x151x3, isto é, uma janela de tempo com 151 amostras de cada um dos 3 eixos do sensor.

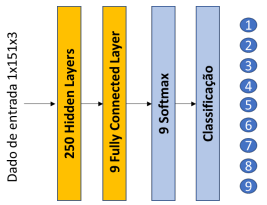

Figura 7. Camadas do modelo de rede recorrente

IV. Resultados

Os resultados dos experimentos foram obtidos a partir da execução do algoritmo em Notebook Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz, sendo a unidade GPU NVIDIA GeForce MX110 o hardware utilizado para o treinamento da rede neural.

O processo de obtenção de resultados compreendeu os experimentos realizados a fim de serem estabelecidos os parâmetros do modelo, dentre eles o número de *Hidden Units* da camada LSTM. Na tabela I são apresentados os dados de acurácia do modelo quando são utilizados 100, 200 e 250 *Hidden Units*.

Tabela I
Definição da quantidade de Hidden Units

Parâmetro	Exper. 1	Exper. 2	Exper. 3
Hidden Units	100	200	250
Tempo de treinamento	23s	46s	56s
Número de épocas	30	30	30
Acurácia	88%	92,66%	*93,82%

*maior resultado obtido neste paper

A partir dos resultados obtidos, foi possível observar aumento do tempo de treinamento à medida que mais *Hidden Units* foram adicionados. Em contrapartida, é notável o ganho de acurácia, de tal forma que para os demais experimentos, considerou-se o número de estados escondidos como sendo 250.

O processo de treinamento da rede pôde ser acompanhado em tempo real por meio das curvas de acurácia e perda. Na figura 8 é exibido o gráfico obtido para a sessão de melhor desempenho do modelo.

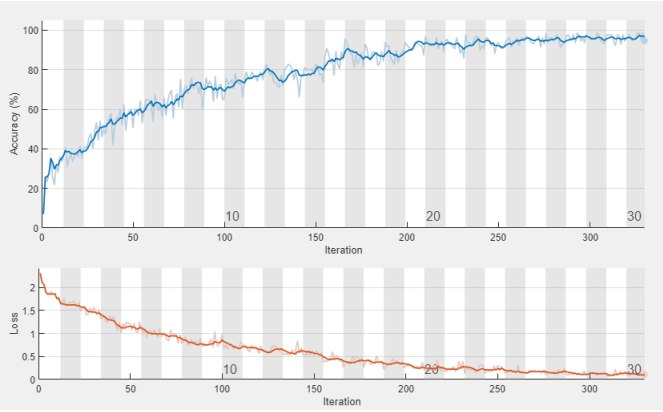

Figura 8. Curvas de treinamento

Ao final do processo de treinamento, o modelo foi utilizado para realizar predições no conjunto de validação. De posse das classes esperadas e das classes preditas, foi possível metrificar o desempenho do modelo. A figura 9, ilustra a matriz de confusão gerada para o modelo de melhor desempenho, isto é, para o modelo que resulta em 93,82% de acurácia.

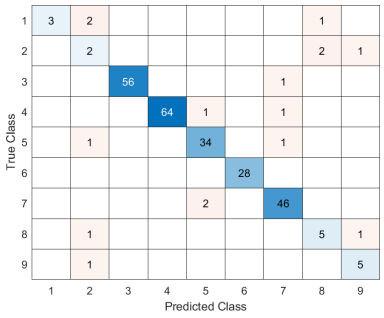

Figura 9. Matriz de confusão

V. Conclusões

A implementação da rede recorrente LSTM mostrou-se adequada à resolução do problema apresentado. Dentre os diferenciais deste tipo de rede, está o fato de permitir séries de dados temporais em sua entrada, o que possibilita seu uso em aplicações que envolvem sinais das mais variadas naturezas. Os conceitos vistos em sala de aula se fizeram presentes no entendimento da arquitetura dos blocos de memória LSTM e seu impacto no desempenho das redes recorrentes, o que pode ser visto na tabela I. A acurácia obtida de 93,82% é refletida na matriz de confusão calculada para o conjunto de validação, a qual pode ser vista na figura 9. Da matriz de confusão foi possível depreender que as classes de melhor desempenho foram 3, 4, 5, 6 e 7, o que alinha-se com o fato de serem as classes com maior representatividade no conjunto de dados, isto é, o modelo dispunha de mais exemplos para o treinamento.

Referências

[1] http://www.sal.disco.unimib.it/technologies/unimib-shar/

[2] Silva, E. ”Desenvolvimento de um Sistema de Detecção de Quedas para Idosos”, Dissertação do Mestrado Profissional em Ciência e Tecnologia de Saúde, Universidade Estadual da Paraíba, 2018.

[3] Bragança, H. ”Reconhecimento de Atividades Humanas usando medidas estatísticas dos sensores inerciais dos smartphones”, Dissertação do Programa de Pós-Graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas, 2019.

[4] Mohammad, R. https://medium.com/@rubeen.786.mr/human-activity-recognition-har-db5c1432cd98

[5]Hochreiter, S. e Schmidhuber, J. ”Long short-term memory”. Neural Computation 9 (8): 1735–1780, 1997.

[6] Zhong, K. ”LSTM Networks for Mobile Human Activity Recognition”, International Conference on Artificial Intelligence: Technologies and Applications (ICAITA), 2016.

[7] Hajihashemi, V. et al. ”Automated Human Activity Recognition by a modified hybrid Recurrent Neural Network”. Doctoral Symposium in Informatics Engineering (DSIE’21), 10 pag., Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 6-7 April, 2021

[8] https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html

ANEXO I

```
% Projeto 3: Aprendizado de M quina
% Implementa o de uma rede recorrente LSTM
% para classificacao de atividade humana
% utilizando sensores aceler metros
% Alunos: Carlos Fonseca e Willian Guerreiro

% Load Sequence Data
% Load the human activity recognition data.
%The data contains seven time series of sensor data
% obtained from a smartphone worn on the body.
% Each sequence has three features and varies in length.
% The three features correspond to the accelerometer readings
% in three different directions.
%load HumanActivityTrain

load data.mat
X
XTrain = X';

load labels.mat
Y
YTrain = categorical(Y');

TrainSet = 0.85;
TestSet  = 0.15;

X3D = cell(uint16(1724*TrainSet),1);
Y3D = zeros(uint16(1724*TrainSet),1);

X3D_Teste = cell(uint16(1724*TestSet),1);
Y3D = zeros(uint16(1724*TestSet),1);



shuffled = randperm(1724);

for i = 1:uint16(1724*TrainSet)
    X3D(i) = {[XTrain(shuffled(i), 1:151); XTrain(shuffled(i), 152:302); XTrain(shuffled(i), 303:453)]};
end
Y3D = YTrain(shuffled(1:uint16(1724*TrainSet)));

for i = uint16(1724*TrainSet+1):1724
    X3D_Teste(i - uint16(1724*TrainSet)) = {[XTrain(shuffled(i), 1:151); XTrain(shuffled(i), 152:302); XTrain(shuffled(i), 303:453)]};
end
Y3D_Teste = YTrain(shuffled(uint16(1724*TrainSet+1):1724));

% Define LSTM Network Architecture
% Define the LSTM network architecture.
% Specify the input to be sequences of size 3
% (the number of features of the input data).
% Specify an LSTM layer with 200 hidden units,
% and output the full sequence. Finally, specify
% five classes by including a fully connected layer
% of size 5, followed by a softmax layer and
% a classification layer.

 numFeatures = 3;
 numHiddenUnits = 250;
 numClasses = 9;

 layers = [ ...
     sequenceInputLayer(numFeatures)
     lstmLayer(numHiddenUnits,'OutputMode','last')
     fullyConnectedLayer(numClasses)
     softmaxLayer
     classificationLayer];

 %Specify the training options. Set the solver
 % to 'adam'. Train for 60 epochs. To prevent
 % the gradients from exploding, set the gradient
 % threshold to 2.

 options = trainingOptions('adam', ...
     'MaxEpochs',30, ...
     'GradientThreshold',2, ...
     'Verbose',0, ...
     'ExecutionEnvironment','gpu',...
     'Plots','training-progress');

%Train the LSTM network with the specified training
% options using trainNetwork. Each mini-batch contains
% the whole training set, so the plot is updated once
% per epoch. The sequences are very long, so it might
% take some time to process each mini-batch and update
% the plot.

net = trainNetwork(X3D,Y3D,layers,options);

%Test LSTM Network
% Classify the test data using classify.

YPred = classify(net,X3D_Teste);

% Calculate the accuracy of the predictions.
acc = sum(YPred == Y3D_Teste)./numel(Y3D_Teste)

C = confusionmat(Y3D_Teste, YPred)

confusionchart(C)
```