UNISINOS

Programa de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Arturo de Souza

MoStress: um Modelo de Aprendizado Profundo para Deteção de Estresse a partir de Sinais Fisiológicos

São Leopoldo, 2023

Arturo de Souza

MOSTRESS: UM MODELO DE APRENDIZADO PROFUNDO PARA DETECÇÃO DE ESTRESSE A PARTIR DE SINAIS FISIOLÓGICOS

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Gabriel de Oliveira Ramos

Coorientador:
Prof. Dr. Sandro José Rigo

São Leopoldo
2023

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

de Souza, Arturo

MoStress: um Modelo de Aprendizado Profundo para Detecção de Estresse a partir de Sinais Fisiológicos / Arturo de Souza — 2023.

84 f.: il.; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2023.

"Orientador: Prof. Dr. Gabriel de Oliveira Ramos, Unidade Acadêmica de Pesquisa e Pós-Graduação".

1. Aprendizado de Máquina. 2. Detecção de Estresse. 3. Séries Temporais. 4. Sensores Vestíveis. 5. Aprendizado Profundo. 6. Redes Neurais Recorrentes. I. Título.

CDU 004.4

Bibliotecária responsável: Silvana Dornelles Studzinski — CRB 10/2524

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001
O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001

À minha família. No sentido mais amplo da palavra.

AGRADECIMENTOS

Um agradecimento especial a todos os meus orientadores acadêmicos! Vocês me inspiram muito e sou eternamente grato a todos!

Eu sigo naquela fé
Que talvez não mova montanhas
Mas arrasta multidões e esvazia camburões
Preenche salas de aula e corações vazios
— DJONGA

RESUMO

A pandemia de COVID-19 mostrou como a preparação e combate para situações de calamidades públocas tem papel crucial no desenvolvimento da sociedade moderna. Entretanto, o coronavírus não é a única doença pandêmica que atinge o globo: doenças de cunho psicológico também atingem uma grande parcela da população. Nos dias de hoje, o estresse, ansiedade e depressão são consideradas doenças psicológicas e aproximadamente 10.7% da população mundial sofre com algum desses distúrbios, assim, essas doenças possuem um potencial pandêmico e devem ser tratadas com a urgência necessária. Uma abordagem que pode ser utilizada para lidar com tais doenças é a detecção destes distúrbios a partir da utilização de algoritmos de aprendizado de máquina que utilizam séries temporais como dados de entrada. Tendo em vista a grande gama de sinais fisiológicos medidos pelos sensores modernos, como respiração, temperatura, batimentos cardíacos e outros; e tendo em vista também que estes sensores são cada vez mais populares na sociedade, a utilização de tais sinais para monitorar todos os tipos de doenças ganha grande relevância. Desta forma, tratar séries temporais que representam sinais fisiológicos com técnicas modernas de aprendizado de máquina pode gerar resultados substanciais na melhora da qualidade de vida da população, pois, com tais algoritmos, diversas doenças poderão ser classificadas mais rapidamente e de maneira mais eficaz, facilitando o diagnóstico de um profissional da saúde e reduzindo o risco de tais doenças chegarem a estados mais graves. Neste trabalho apresentaremos o MoStress, um modelo de aprendizado profundo que recebe um conjunto de séries temporais que representam sinais fisiológicos e faz a classificação de estresse. O MoStress é constituído por uma etapa de pré-processamento que consiste em utilizar a Transformada de Fourier para eliminação de ruídos, o *Rolling Z-Score* para normalização de dados, o janelamento por frequência de classes para classificação de janelas e o cálculo de pesos para reduzir o desbalanceamento de dados. Além disso, o MoStress também possui um rede neural que faz a classificação a partir dos dados pré-processados, onde essa rede neural pode ser constituída de uma rede neural recorrente, uma *Echo State Network* ou uma combinação da arquitetura do NBeats mais uma rede neural perceptron multicamadas. O MoStress utilizou dados fisiológicos públicos coletados pela Universidade de Siegen, na Alemanha (conjunto de dados chamado WESAD), onde este conjunto de dados possui 3 classes diferentes sendo elas *baseline*, *stress* e *amusement*. Dessa maneira, o MoStress utilizando os sinais fisiológicos de respiração, temperatura, eletrocardiograma, eletromiograma e atividade eletrodermal, coletados por um sensor localizado no tórax dos participantes, passando tais sinais pela etapa de pré-processamento e utilizando uma rede neural recorrente, obteve uma acurácia de 96.5% no problema de multi classificação com 3 classes e ainda obteve *recall*, *f1-score* e precisão de 96%, 93% e 94%, respectivamente, para a classe *stress*, mostrando boa performance de classificação com os dados pré-processados e uma rede neural recorrente.

Palavras-chave: Aprendizado de Máquina. Detecção de Estresse. Séries Temporais. Sensores Vestíveis. Aprendizado Profundo. Redes Neurais Recorrentes.

ABSTRACT

The COVID-19 pandemic showed how the preparation and fight against those diseases plays a crucial role on the modern society. However, the coronavirus is not the only pandemic diseases which afflicts the globe: mental illnesses also afflict a large number of the world population. Nowadays, stress, anxiety and depression are classified as mental illness and proximally 10.7% of the world population suffers with one of those diseases, therefore, mental illness might have a high pandemic potential and should be treated with the necessary urgency. One approach to deal with mental illness is to use machine learning algorithm which uses time series as input to detect those diseases. Considering the huge variety of physiologic measured by modern sensors, such as temperature, heart rate and others, and also considering the increase popularity of those sensors in our society, the use of those signals to monitor all kind of the diseases gains more relevance. In that sense, dealing with time series which represents physiologic signs with modern machine learning technics, may result in a substantial improvement of life quality of the population, because with those algorithms, several diseases might be classified quickly and more efficient, making more ease the health care professional diagnoses and avoiding the diseases to reach an worst scenario. This work introduce the MoStress, a deep learning model which get as input time series which represents physiologic signs and make stress classification. The MoStres is made by a pre-processing step, which consists in using Fourier Transform to clean noise, Rolling Z-Score to normalize the data, windowing by class frequency to window classification and weight calculation to deal with unbalance data. Besides that, the MoStress also have a deep neural network which make the classification using the pre-processed data, where this neural network consists on one of the following models: a recurrent neural network, a Echo State Network or a combination of the NBeats and a Multi Layer Perceptron network. The MoStress used public physiologic data collected by the Siegen University, in Germany (the dataset is named WESAD), where this dataset is constituted also by 3 different classes: baseline, stress amusement. Considering this, the MoStress using physiologic signals of respiration, temperature, electrocardiogram, electromyogram and electrodermal activity, collected via chest sensor after pre-process these data and using a recurrent neural network, achieved accuracy of 96.5% on the 3 class classification problem and also achieved recall, f1-score and precision of 96%, 93% and 94%, respectively, for the stress class, showing the good performance on classification problem with pre-processed data and a recurrent neural network.

Keywords: Machine Learning. Stress Detection. Time Series. Wearable Sensors. Deep Learning. Recurrent Neural Networks.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Transtornos psicológicos como depressão, estresse e ansiedade, se tornaram comuns na sociedade moderna. Dados recentes mostram que por volta de 10.7% da população mundial sofre de pelo menos um tipo desses transtornos (SALONI DATTANI; ROSER, 2021). Se tomarmos o exemplo da ansiedade, entre 1990 e 2017, houve um aumento de 94 milhões de pessoas afetadas por tal transtorno, o que corresponde a um aumento de mais de 50% (SALONI DATTANI; ROSER, 2021).

Estas condições mentais também têm se mostrado extremamente perigosas para as pessoas por elas afetadas. A relação entre transtornos psicológicos e suicídio, apresentada por (FERRARI et al., 2014), indica que um indivíduo sofrendo de ansiedade tem 3 vezes mais chances de cometer suicídio do que uma pessoa mentalmente sadia. Se considerarmos a depressão, o risco de suicídio cresce 20 vezes. Com isso em mente, detectar esses tipos de pertubações mentais se torna uma tarefa crucial.

Muitos desses transtornos mentais afetam de alguma maneira nosso cérebro e podem causar danos graves à saúde. Como exemplo, podemos citar estudos já antigos, como (GREDEN, 2001), que aponta como depressão e estresse crônicos podem causar redução do volume hipocampal, alterações no córtex pré-frontal, redução de neurônios e outros consequências que podem levar a alterações de humor e como citado acima, a tentativas de suicídio. Desta maneira, transtornos psicológicos podem ser responsáveis por grandes alterações cerebrais que terão consequência em todo o corpo, como exemplo, pode alterar a condutância elétrica da pele, respiração, sinais de eletrocardiograma e outros, como mostrado em (HEALEY; PICARD, 2005).

Assim, doenças psicológicas podem alterar diversos sinais fisiológicos de maneira diferente, portanto, quantos mais sinais vitais possam ser coletados, maior será a gama de dados possíveis para serem analisados e utilizados no enfrentamento de tais doenças.

Uma abordagem que tem se mostrado promissora para a detecção de diversas doenças são os sensores vestíveis. A cada dia que passa esses dispositivos estão se tornando cada vez mais acessíveis ao público e também com uma crescente capacidade de medir os mais diversos sinais vitais, que por sua vez podem ser usados nas mais variadas tarefas. Por exemplo, durante a pandemia do COVID-19, diversas pesquisas se utilizaram sensores vestíveis e os sinais provenientes deles, na detecção do SARS-COV-2 (ATES et al., 2021). A eficácia do uso de tais dispositivos acabou por abrir portas para uma variedade maior deles no futuro, tais como máscaras inteligentes, camisetas inteligentes e outros, assim como descrito em (ATES et al., 2021). Por isso, os sensores vestíveis tendem a se diversificar e popularizar cada vez mais.

Outra tecnologia muito popular e que pode ajudar na diminuição de casos de transtornos mentais são os algoritmos de aprendizado de máquina. Estes algoritmos são capazes de pro-

ver valiosas informações para diversos tipos de problema utilizado uma grande gama de dados. Alguns exemplos de aprendizado de máquina utilizados em aspectos psicológicos podem ser encontrados usando postagens de mídias sociais visando a detecção de depressão (ORABI et al., 2018), usando imagens cerebrais na detecção de autismo (ESLAMI et al., 2019) ou até utilizando de sinais fisiológicos na classificação de emoções humanas (SANTAMARIA-GRANADOS et al., 2018).

A detecção de estresse também é uma popular aplicação de aprendizado de máquina utilizando dados médicos. Existem estudos que se mostraram capazes de utilizar modelos de aprendizado de máquina em medidas de sinais fisiológicos coletados por sensores vestíveis, como pressão sanguínea, batimentos cardíacos e condução elétrica da pele (GJORESKI et al., 2016a) para permitir a distinção entre pessoas com e sem estresse. Outros trabalhos foram capazes de classificar o estresse através de algoritmos supervisionados, como k-NN(SAEED et al., 2020), ou classificador Naive Bayes (ALHITARY et al., 2018).

Existem diversos estudos médicos na área de detecção de doenças psicológicas que descrevem como tais doenças afetam o corpo e quais consequências elas podem trazer para a vida cotidiana. Tais estudos, aliados com os dados de sinais vitais e algoritmos de aprendizado de máquina, abrem espaço para uma grande gama de estudos na área de computação aplicada que podem contribuir para a tarefa de detecção de estresse e também de outros distúrbio mentais. Não obstante tal oportunidade, é possível notar algumas lacunas no estudo de detecção de estresse a partir de sinais fisiológicos ,principalmente no âmbito do uso do estado da arte de algoritmos de aprendizado e que podem ser grandes aliados na melhora da qualidade de vida das pessoas.

Assim, este trabalho apresenta o MoStress, um modelo de aprendizado profundo que visa fazer a classificação de estresse a partir de séries temporais que representam sinais fisiológico. Este novo modelo busca não só utilizar técnicas de pré-processamento de dados e teoria de redes neurais para a classificação de estresse, como também busca inserir o estado da arte de rede neurais profundas no mesmo problema. Dessa maneira, o MoStress visa preencher a lacuna de utilização do estado da arte em redes neurais no problema de classificação de estresse a partir de sinais fisiológicos.

1.2 Questão de Pesquisa e Objetivos

Como descrito na Seção 1.1, podemos considerar que distúrbios psicológicos são doenças que afetam uma grande parte da população, podendo trazer danos severos à qualidade de vida das pessoas. Esta seção também mostrou que existem estudos não só na área da saúde como também na área de computação aplicada que visam estudar tais doenças. Assim, tendo em vista enorme gama de tais doenças, este estudo escolheu o estresse como distúrbio a ser estudado pelos seguintes motivos:

- O estresse pode ser um sintoma e um dos principais motivos de diversas outras doenças

psicológicas;

- Existem exames clínicos de indução de estresse, como o TSST (KIRSCHBAUM; PIRKE; HELLHAMMER, 1993) e outros que facilitam a criação de dados sobre estresse;
- Existem protocolos que realizam diagnósticos precisos se o paciente está ou não com altos níveis de estresse, como o PANAS (WATSON; CLARK; TELLEGEN, 1988) e outros;
- Existem bancos de dados públicos com dados sobre estresse, de modo que é possível utilizar tais dados para traçar paralelos com outros estudos e focar mais nos modelos utilizados.

Não obstante a existência de estudos importantes relacionados ao estresse, é possível notar uma lacuna na aplicação de algoritmos de aprendizado de máquina no estado da arte (tal lacuna será explorada em detalhes na Seção 3) e que utilizam sinais fisiológicos para a detecção de estresse, o que leva à seguinte pergunta de pesquisa: Como detectar estresse agudo a partir de sinais fisiólogos associados utilizando técnicas de aprendizado profundo?

Para responder a essa, foram traçados os seguintes objetivos:

1. Entender como sinais fisiológicos podem ser tratados e utilizados em algoritmos de aprendizado de máquina.
2. Criar uma arquitetura que possa fazer a classificação de estresse de maneira satisfatória.
3. Empregar modelos de aprendizado de máquina que representem o estado da arte para lidar com séries temporais, buscando propor melhorias no desempenho destas técnicas, no domínio específico de detecção de estresse.

Tais objetivos foram utilizados para dar direção ao presente trabalho e auxiliar na criação dos experimentos que ajudem na criação da resposta de nossa pergunta de pesquisa.

1.3 Contribuições

Este trabalho investiga maneiras de serem feitas a classificação de estresse agudo utilizado séries temporais que representam sinais vitais utilizando algoritmos e arquiteturas de aprendizado de máquina que são o estado da arte. É esperado que essa investigação possa gerar mais possibilidades de como lidar com o estresse e possa ajudar na criação de mecanismos que possam combater a doença. Sendo assim, as principais contribuições desse trabalho são:

1. Uma arquitetura sequencial de aprendizado de máquina capaz de classificar estresse com exatidão, o MoStress. Essa arquitetura é composta de uma fase de pré-processamento que possui as seguinte etapas:
   (a) Transformada de Fourier para limpar os ruídos dos sinais temporais;

(b) Normalização *Rolling Z-Score* para normalizar os dados mantendo suas característica estatísticas;

(c) Determinação de classe de uma janela por frequência de classes dentro do período de janelamento;

(d) Geração de um vetor de pesos para cada classe com o intuito de mitigar problemas causados por dados desbalanceados.

Além da etapa de pré-processamento, essa arquitetura também é composta de uma rede neural profunda responsável por gerar os resultados de classificação.

2. A utilização de 3 arquiteturas diferentes de redes neurais profundas, sendo elas:

   (a) Uma Rede Neural Recorrente composta de neurônios GRU ou LSTM;

   (b) Uma rede *Echo State Networks*;

   (c) Uma adaptação do modelo do NBeats (modelo cujo objetivo é fazer a previsão de uma única série temporal) para fazer classificação de múltiplas séries temporais.

3. Um conjunto extenso de experimentos para validar a eficácia da arquitetura do MoStress para a classificação de estresse, onde os melhores resultados foram obtidos pelas Redes Neurais Recorrentes, com valores de 96.5% de acurácia na classificação entre *baseline*, *stress* e *amusement* e ainda 96, 93% e 94 de *recall*, *f1-score* e precisão, respectivamente, para a classe *stress*.

É importante notar que tais contribuições não exaurem as possibilidades de contribuições nestes mesmos tópicos tendo em vista a complexidade dos modelos utilizados e vasta teoria de redes neurais em geral. Sendo assim, podemos afirmar que existem várias maneiras de se construir modelos sequenciais que possam classificar o estresse e diversas maneiras de se utilizar ou adaptar as arquitetura do NBeats e das *Echo State Networks* para fazer a classificação de estresse. Tais possibilidades também serão exploradas e citadas em mais detalhes nos trabalhos futuros, descritos na Seção 6.2.

1.4 Estrutura do Texto

Este trabalho está estruturado em 6 seções. A Seção 2 começa explicando brevemente como o estresse se manifesta no corpo e em seguida segue para tópicos importantes de aprendizado de máquina como: a teoria por trás das RNN's e das ESN's, a arquitetura do Nbeats, a teoria sobre os tipos de otimizadores e também métricas de avaliação para problemas de classificação.

A Seção 3 mostra um panorama dos trabalhos na área de detecção de estresse utilizando aprendizado de máquina, dando uma ideia geral dos sinais que já foram utilizados, os modelos

que já foram testados e nos fornecendo uma ideia geral dos modelos que compõem o estado da arte nessa área.

Na Seção 4 é apresentado o MoStress, a principal contribuição desse trabalho. Nesta seção são detalhados e justificados todos os passos para se construir o MoStress. Esse capítulo detalha a etapa de pré-processamento na Seção 4.2, bem como as arquiteturas que já foram utilizadas pelo MoStress, nas Seções 4.3.1, 4.3.2 e 4.3.3 e também o método de avaliação do modelo na Seção 4.4.

Os experimentos realizados por esse trabalho se encontram na Seção 5, onde é descrito o conjunto de dados utilizados no experimento, na Seção 5.1.1.1 e Seção 5.1.2.2 são apresentados os modelos *baseline* que serão utilizados para comparação com outros experimentos; e nas Seções 5.1.2.3, 5.1.2.4 e 5.1.2.5 são descritos e discutidos as arquiteturas utilizadas nos experimentos. O fim desta seção também apresenta uma comparação entre os modelos utilizados na literatura e faz uma discussão sobre os resultados alcançados

Por fim, a Seção 6 encerra este trabalho resumindo suas limitações e contribuições, bem como expõe os trabalhos futuros e artigos publicados.

2 FUNDAMENTAÇÃO TEÓRICA

Para contextualizar os assuntos tratados nesse trabalho, esta seção vai apresentar o embasamento teórico necessário para as discussões que serão apresentadas. Assim, a Seção 2.1 começa dando um panorama geral de como o Estresse afeta o corpo humano e como isso pode afetar os sinais fisiológicos de uma pessoa.

Após entender o necessários sobre o estresse, a Seção 2.2 explica os principais aspectos de redes neurais abordados nesse trabalho. Portanto, as Seções 2.2.1, 2.2.2 e 2.2.3 explicam os principais aspectos da teoria por trás das Redes Neurais Recorrentes (RNN), das *Echo State Networks* (ESN) e do *Neural Basis Expansion Analysis or Interpretable Time Series Forecasting* (NBeats), respectivamente. Estes três modelos foram implementados no MoStress e portanto foram utilizados nos experimentos desse trabalho. Dessa maneira, é importante ter o entendimento teórico desses modelos.

Além dos modelos, esse trabalho também realizou experimento com diversos otimizadores. Dessa forma, a Seção 2.2.4 faz a explanação teórica dos otimizadores Adam, Nadam e RMSPropo, que foram os utilizados nessa dissertação.

Por fim, este trabalho tem como objeto de estudo um problema de classificação de estresse. Por conta disso, as Seções 2.2.5 e 2.2.6 explicam a teoria por trás da função de ativação utilizada para problemas de multi-classificação e as métricas de avaliação utilizadas em problemas de classificação, respectivamente.

2.1 Estresse

O estresse é uma resposta natural do corpo humano a qualquer demanda imposta sobre ele e que traz para o corpo a necessidade de se adaptar (SELYE, 1973). Por exemplo, essa demanda pode se apresentar de diversas maneiras diferentes, fazendo um paralelo: quando exposto ao calor, nosso corpo sua como mecanismo de arrefecimento, assim em situações que demandam grande atenção nosso corpo utiliza o estresse. Utilizando um exemplo de demanda mental como trabalhar ou estudar: um estudante que vai fazer uma prova tem uma crescente demanda de atenção que faz com que o corpo entregue muito mais energia para o cérebro e isso é causado pelo estresse.

Para entender melhor como o estresse afeta os sinais fisiológicos do corpo humano, o trabalho feito por (SCHNEIDERMAN; IRONSON; SIEGEL, 2005) explica que o estresse agudo gera uma série de mudanças no sistema nervoso, cardiovascular, endócrino e sistema imune. O processo começa quando os hormônios do estresse são produzidos pelo sistema nervoso simpático (SNS) e o eixo hipotálamo-hipófise-adrenal (HPA) por conta de uma situação de estresse (por exemplo: sofrer algum acidente, fazer uma prova e etc). Estes hormônios encadeiam uma série de reações que acabam por gerar dois processos principais: a lipólise e a conversão de glicogênio em glucose.

Lipólise é o processo de quebra de gordura em fontes de energia imediata e a conversão de glicogênio em glucose pode ser resumida como a produção de açúcar no sangue. Dessa maneira, a produção dos hormônios do estresse acabam por gerar um aumento de energia disponível no corpo. Esta energia é distribuída pelo organismo para os órgãos que mais precisam no momento. Assim, a pressão sanguínea no corpo com estresse aumenta de forma que a energia seja distribuída através do organismo.

Esse aumento da pressão sanguínea pode acontecer a partir de dois mecanismos diferentes: o miocárdico e o vascular. O mecanismo miocárdico consiste no aumento da pressão sanguínea através do aumento dos batimentos cardíacos. O mecanismo vascular consiste na diminuição da vascularidade do sangue, assim aumentando a pressão sanguínea.

Cada um desses mecanismos podem ser ativados por diferentes situação que geram estresse. Estudos laboratoriais mostram que tarefas como falar em público ou tarefas matemáticas, tarefas que requerem que o paciente faça algo ativamente, são tarefas que ativam o mecanismo miocárdico, e por conseguinte, causam um aumento nos batimentos cardíacos. Em outro caso, estudos laboratórias mostram que tarefas que exigem ações mais passivas dos voluntários como assistir um vídeo perturbador ou colocar o pé dentro de um balde de gelo, são tarefas que ativam o mecanismo vascular para o aumento da pressão sanguínea.

Além da distribuição de energia para o corpo os hormônios do estresse também ativam o sistema imune. Ao entrar em uma situação de estresse agudo, o organismo ativa as células do sistema imune consideradas as "primeiras linha de defesa"do organismo. Estas células entram na corrente sanguínea do corpo aumentando temporariamente o número de células imunes em circulação. A partir disso, essas células movem-se para tecidos mais prováveis de serem danificados em confrontos físicos como a pele e os ossos. Assim, no momento que tais células se posicionam em tais tecidos, o organismo está preparado para conter qualquer bactérias que posso entrar no corpo através de alguma ferida, visando facilitar o processo de cura do organismo.

Dessa maneira, situações de estresse geram hormônios que alteram o funcionamento geral do nosso corpo. Assim, em tais situações os sinais fisiológicos do organismo, como batimentos cardíacos, pressão sanguínea, frequência de respiração e outros, são alterados visando o bom funcionamento do organismo perante tal situação. Por isso, a medição dos sinais fisiológicos do organismo em uma situação de estresse podem ser utilizados identificar padrões de estresse.

2.2 Redes Neurais

Existem diversos tipos de algoritmos e técnicas de aprendizado de máquinas, desde modelos mais simples e estatísticos como regressões logística ou lineares até modelos mais complexos como redes neurais profundas, aprendizado por reforço e outros. Tendo em vista essa gama de modelos possíveis, este trabalho vai detalhar alguns aspectos da área de aprendizado de máquina que serão utilizados ao decorrer do texto, por isso as próximas seções vão descrever as Redes

Neurais Recorrentes (RNN), as *Echo State Networks* (ESN) e a arquitetura do NBeats.

Além desses tópicos, também serão descrita a teoria por trás de alguns dos otimizadores utilizados em redes neurais bem como aspectos utilizados em problemas de classificação, tendo em vista que este trabalho visa a classificação de estresse.

2.2.1 Redes Neurais Recorrentes | RNN

Técnicas modernas de aprendizado profundo podem ser usados como uma importante ferramenta para uma grande variedade de tarefas (GOODFELLOW; BENGIO; COURVILLE, 2016). Aprendizado supervisionado é um dos principais agentes no quesito de técnicas no estado-da-arte, como por exemplo o recente GPT-3 (BROWN et al., 2020), que pode ser utilizado no processamento de linguagem natural, redes neurais recorrentes que podem ser utilizadas na previsão de séries temporais (SEN; YU; DHILLON, 2019) e também aprendizado profundo utilizado no reconhecimento de imagens de tumores cerebrais (SULTAN; SALEM; AL-ATABANY, 2019).

Um importante tipo de rede neural artificial usada em aprendizado profundo são as Redes Neurais Recorrentes (*Recurrent Newral Networks* - RNN). RNNs são comumente usadas em análises de séries temporais, tendo em vista que sua principal característica é levar em conta o tempo passado para calcular sua saída, e que vai determinar o tipo de problema a ser resolvido, como problemas de classificação ou previsão de séries temporais. Um previsão feita por uma RNN pode ser descrita pela Equação 2.1:

$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}, \mathbf{x}^{(t)}; \theta), \tag{2.1}$$

onde o estado atual $\mathbf{s}^{(t)}$ é calculado baseado no estado passado $\mathbf{s}^{(t-1)}$, input $\mathbf{x}^{(t)}$, e no parâmetro θ do neurônio da rede. *Long Short-Term Memory* (LSTM) e *Gate Recurrent Unit* (GRU) são exemplos de RNNs. Estes tipo de redes possuem diferentes *gates*, que podem ser usados para aprender uma nova informação, atualizá-la, esquecê-la ou utilizá-la como saída de uma previsão baseada na sequência de entrada. Este mecanismo ajuda a prevenir o problema desaparecimento do gradiente, que após diversas interações de aprendizagem com o procedimento de *back-propagation* em sequências temporais, o valor do gradiente é muito pequeno e não possui relevância alguma na atualização dos parâmetros da rede (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.2.2 *Echo State Networks* | *ESN*

As *Echo State Neural Network* (ESNs) foram primeiramente introduzidas por (JAEGER, 2002). O termo "*Echo*", que significa "eco", significa que a ativação do estado **x(n)** de uma

RNN montada arbitrariamente é função da "história"dos dados de entrada na rede. De acordo com (JAEGER, 2002), uma ESN genérica é composta de uma rede neural com K neurônios de entrada, N neurônios de RNN internos, dispostos aleatoriamente e L neurônios de saída, como mostrado na Figura 1. Assim, a ativação de cada neurônio na unidade de tempo n pode ser descrita pelas Equações 2.2, 2.3 e 2.4, que se referem as entradas da rede, o estado interno da rede e a saída da rede, respectivamente.

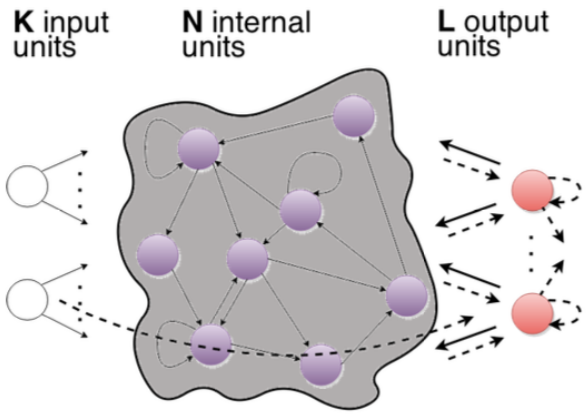


Figura 1: Arquitetura Geral das ESN's. Fonte: (MELANDRI, 2014)

$$\mathbf{u(n)} = (u_1(n), u_2(n), ..., u_K(n)) \tag{2.2}$$

$$\mathbf{x(n)} = (x_1(n), x_2(n), ..., x_N(n)) \tag{2.3}$$

$$\mathbf{y(n)} = (y_1(n), y_2(n), ..., y_L(n)) \tag{2.4}$$

As ligações entre os conjuntos de neurônios podem ser definidos nas seguintes matrizes:

1. Conexões de Entrada: definidos pela matriz $\mathbf{W}^{in} = (w_{ij}^{in})$, de dimensão NxK, onde a contração "in"se refere a palavra "input", "entrada"em português e que define a relação de pesos entre os neurônios de entrada e os internos;

2. Conexões Internas: definidos pela matriz $\mathbf{W} = (w_{ij})$, de dimensão NxN, que define a relação de pesos entre os neurônios internos;

3. Conexões de Saída: definidos pela matriz $\mathbf{W}^{out} = (w_{ij}^{out})$, de dimensão $Lx(K+N+L)$, onde a contração "out"se refere a palavra "output", "saída"em português e que define a relação de pesos entre a saída da rede, os neurônios de saída, os internos e os de entrada;

4. Conexões de *Backprojection*: definidos pela matriz $\mathbf{W}^{back} = (w_{ij}^{back})$, de dimensão NxL, onde a contração "back"se refere a palavra "backprojection", que nesse casso pode ser definido como retroalimentação em português e que define a relação de pesos entre os neurônios de saída e os internos.

Assim, considerando $f = (f_1, ..., f_N)$ e $f^{out} = (f_1^{out}, ..., f_L^{out})$ as funções de ativação dos neurônios internos e de saída, respectivamente, podemos definir as Equações 2.5 e 2.6, que regem as relações dentro das ESN's.

$$x(n+1) = f(W^{in}u(n+1) + Wx(n) + W^{back}y(n)) \tag{2.5}$$

$$y(n+1) = f^{out}(W^{out}(u(n+1), x(n+1), y(n))) \tag{2.6}$$

É interessante notar que nas ESN's os neurônios internos são dispostos de maneira aleatória, visando imitar estados "aleatórios"ou "caóticos"do sistema que a rede tenta aprender. Dessa maneira, a inicialização dos pesos de toda rede é aleatória e o treinamento da rede envolve apenas ajustar os pesos da saída da rede, ou seja, encontrar os valores da matriz W^{out}, o que torna o processo de aprendizagem muito mais veloz.

2.2.3 *Neural Basis Expansion Analysis or Interpretable Time Series Forecasting | N-Beats*

O NBeats (ORESHKIN et al., 2019) é um modelo utilizado para a previsão de séries temporais e sua arquitetura geral pode ser vista na Figura 2

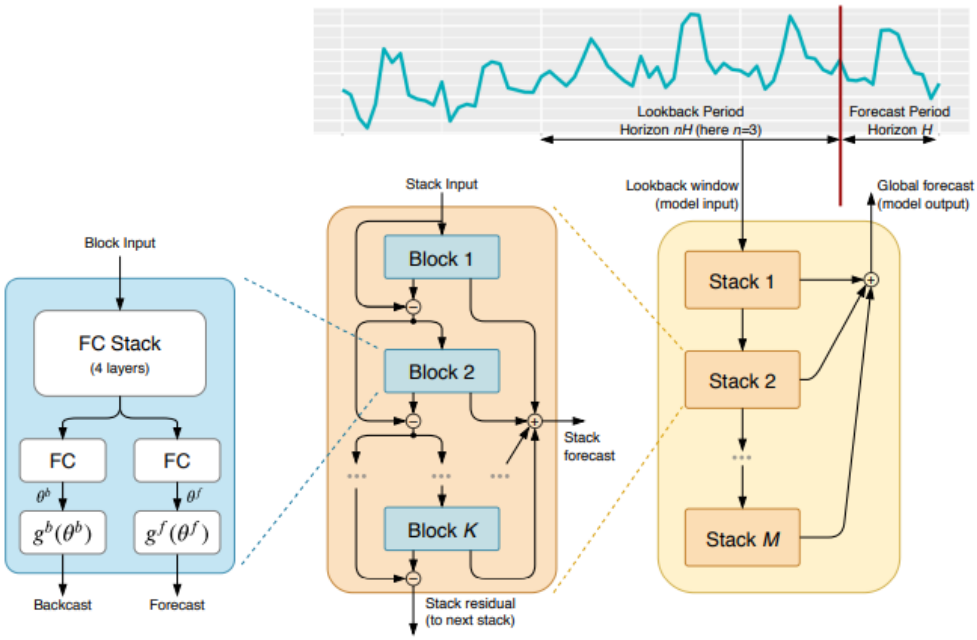


Figura 2: Arquitetura Geral do NBeats. Fonte: (ORESHKIN et al., 2019)

A arquitetura do NBeats consiste em dois elementos principais, os blocos e as pilhas, des-

critos em detalhes nas Seções 2.2.3.1 e 2.2.3.2.

Cada bloco recebe a série temporal ou um resíduo de série temporal do bloco anterior e utiliza como entrada. A partir desses sinais, cada bloco gera um *forecast* e um *backcast*, onde o primeiro é a tentativa do modelo de fazer uma previsão de uma série temporal para um tempo futuro e o segundo é a tentativa do modelo de prever um pedaço do seu sinal de entrada, ou seja, uma previsão dos dados de entrada.

A partir do *backcast*, se faz uma subtração desse sinal com o sinal de entrada do bloco e assim se obtém o resíduo de série temporal, e tal resíduo é utilizado como entrada do próximo bloco.

O *forecast* de um bloco é adicionado com o *forecast* de todos os outros blocos dentro da mesma pilha, de maneira a gerar a previsão geral do modelo. Cada pilha recebe como entrada o resíduo do último bloco da pilha anterior e gera como saída a soma de todos os *forecast* de todos os blocos internos e o resíduo do último bloco interno.

Para fazer a previsão da série temporal a arquitetura do Nbeats também define dois parâmetros importantes: o *horizon* e o *lookback*.

Horizon é o termo que define quantos pontos da série temporal serão obtidos pela previsão do modelo e o *lookback* são quantos *horizon* da série temporal serão utilizados para treinamento do modelo.

2.2.3.1 Blocos

No Nbeats um bloco consiste em uma rede neural com 4 camadas de 512 neurônios mais uma camada de neurônios de saída.

A camada de saída possui um neurônio para cada ponto *horizon* que vai constituir a previsão da série temporal do modelo e também possui um neurônio pra cada ponto *lookback*, que vai ser responsável por constituir o *backcast* do modelo.

As funções g são utilizadas para determinar se o bloco de NBeats vai ser interpretável ou não. Assim, o tipo de bloco utilizado no modelo determina se está sendo utilizado o N-BEATS-G (NBeats Geral) ou o N-BEATS-I (NBeats interpretável).

No caso do N-BEATS-G a função g consiste apenas em uma combinação linear dos dados de entrada na camada com os coeficientes de cada neurônio, como na Equação 2.8. Já no N-BEATS-G, a função g não só faz a combinação linear como também cria blocos de sazonalidade, onde funções trigonométricas são aplicas nas saídas dos neurônios, e tendência, onde funções polinomiais são aplicadas nas saídas dos neurônios.

$$\hat{\mathbf{y}} = g(\theta) \tag{2.7}$$

$$\hat{\mathbf{y}} = \sum_{i=1}^{dim(\theta)} \theta_i \mathbf{v}_i \tag{2.8}$$

$$\hat{\mathbf{y}} = \sum_{i=0}^{p} \theta_i \mathbf{t}^i \tag{2.9}$$

$$\hat{\mathbf{y}} = \sum_{i=0}^{|\frac{H}{2}-1|} \theta_i cos(2\pi i \mathbf{t}) + \theta_{i+\frac{H}{2}} sin(2\pi i \mathbf{t}) \tag{2.10}$$

A Equação 2.7 define a saída $\hat{\mathbf{y}}$, que pode ser tanto referente ao *forecast* quando ao *backcast*. A Equação 2.8 mostra a saída de um bloco referente à um modelo N-BEATS-G e as Equações 2.10 e 2.9 se referem aos blocos de sazonalidade e tendências, respectivamente de um modelo N-BEATS-I.

2.2.3.2 Pilhas

As pilhas do Nbeats consistem em conjuntos de vários blocos. Cada pilha recebe a série temporal original ou o resíduo de uma pilha anterior e como saída gera um *forecast* que é soma dos *forecasts* de cada bloco interno da pilha. Além da previsão, a pilha também tem como o saída o último resíduo do último bloco interno da pilha. Os *forecasts* de cada pilha do modelo são somados e esta soma representa o *forecast* final do modelo.

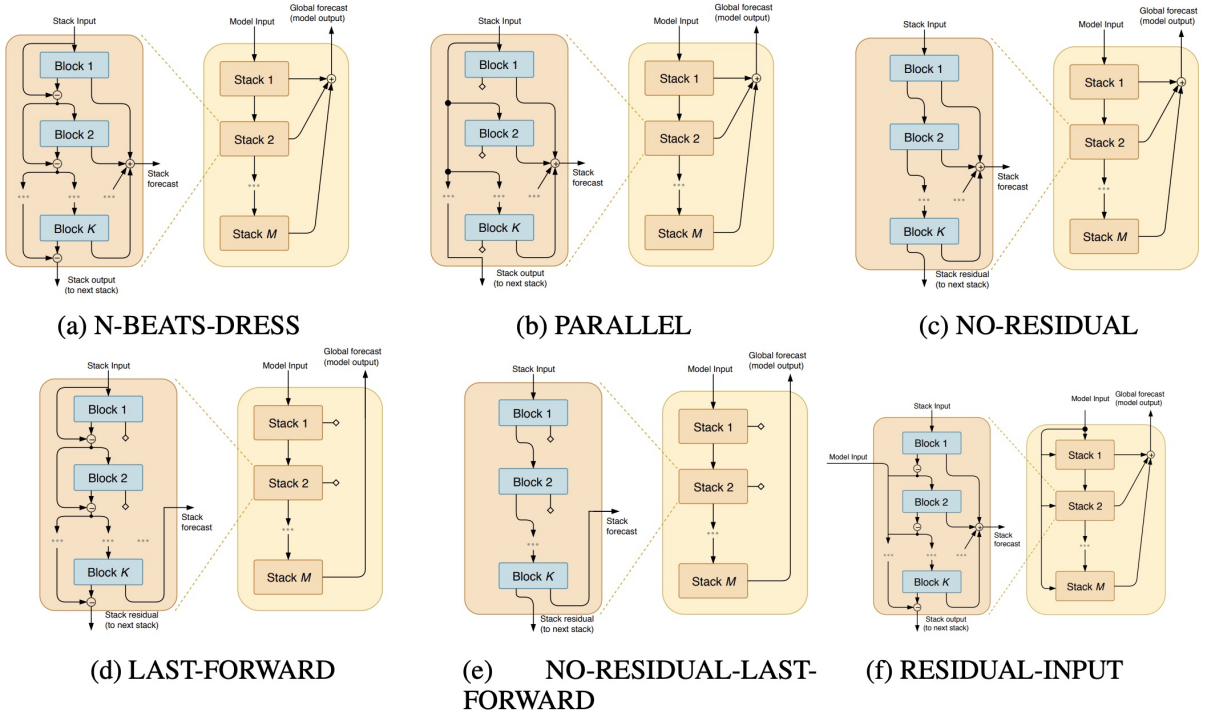
As pilhas não determinam um tipo específico do modelo e são apenas utilizadas como um conjunto de blocos para agrupar seus *forecasts* resultantes. Todavia, a maneira que os blocos e as pilhas são conectados entre si pode criar diversas variantes da arquitetura original.

A Figura 3 mostra as configurações demonstradas em (ORESHKIN et al., 2019). Tais arquiteturas foram estudadas e testas pelos pesquisadores do NBeats, porém, os melhores resultados foram obtidos com a arquitetura mostrada na Figura 2 e por isso essa arquitetura que será usada neste trabalho.

2.2.4 Otimizadores

Treinar uma rede neural artificial pode exigir bastante tempo, especialmente com grandes quantidades de dados ou com dados complexos. Dessa forma, a escolha de uma boa função otimizadora é importante para mitigar o tempo de treino e assim coletar os resultados.

Neste trabalho foram testados três dos principais algoritmos de optimização: RMSProp, Adam e Nadam. Estes três optimizadores tentam alcançar o mínimo local da maneira mais rápida e acurada possível, cada um dos três à sua maneira. Desta maneira, essas três opções são apropriadas para lidar com grandes volumes de dados.



(a) N-BEATS-DRESS (b) PARALLEL (c) NO-RESIDUAL

(d) LAST-FORWARD (e) NO-RESIDUAL-LAST-FORWARD (f) RESIDUAL-INPUT

Figura 3: Possíveis Configurações de Arquitetura do NBeats. Fonte: (ORESHKIN et al., 2019)

Também é importante salientar que estes três optimizadores são baseados no algoritmo do gradiente descendente, mostrado na Equação 2.11, onde η é a taxa de aprendizado, θ e o ponto a ser atualizado pelo gradiente descendente e $J(.)$ é a função de custo. A utilização de *momentum* para otimização do gradiente descendente é mostrado nas Equações 2.12 e 2.13, onde m é o vetor de *momentum* e β é o parâmetro que controla o quão rápido a velocidade terminal pode alcançar.

$$\theta \leftarrow \theta - \eta * \nabla_\theta J(\theta) \tag{2.11}$$

$$m \leftarrow \beta m - \eta * \nabla_\theta J(\theta) \tag{2.12}$$

$$\theta \leftarrow \theta + m \tag{2.13}$$

2.2.4.1 *Root Mean Square Prop* | RMSProp

Função de optimização RMSProp tenta minimizar o componente vertical do movimento atualizado a componente horizontal em direção ao mínimo global. Para fazer isso, o RMSProp acumula o quadrado dos gradientes antigos, como podemos ver nas Equações 2.14 e 2.15.

$$s \leftarrow \beta s - (1 - \beta) * \nabla_\theta J(\theta)^2 \tag{2.14}$$

$$\theta \leftarrow \theta + \frac{\eta * \nabla_\theta J(\theta)}{\sqrt{s + \epsilon}} \tag{2.15}$$

Onde β é a taxa de decaimento, s é a média exponencial quadrada dos gradientes passados e ϵ é o termo de suavização.

2.2.4.2 Adam

O otimizador Adam combina a otimização do *momentum* do gradiente descendente com o acumulo do quadrado dos gradientes das iterações anteriores, como descrito nas Equações 2.16, 2.17, 2.18, e 2.19. Com essa característica, o Adam em geral é melhor utilizado para lidar com gradientes esparsos em problemas com dados complexos ou com um grande número de *features*.

$$m \leftarrow \beta_1 m - (1 - \beta_1) * \nabla_\theta J(\theta) \tag{2.16}$$

$$s \leftarrow \beta_2 s - (1 - \beta_2) * \nabla_\theta J(\theta)^2 \tag{2.17}$$

$$\hat{m} \leftarrow \frac{m}{1 - \beta_1^T} \; ; \; \hat{s} \leftarrow \frac{s}{1 - \beta_2^T} \tag{2.18}$$

$$\theta \leftarrow \theta + \frac{\eta * \hat{m}}{\sqrt{\hat{s} + \epsilon}} \tag{2.19}$$

2.2.4.3 Nadam

A função de optimização Nadam consiste no Adam utilizando *Nesterov Accelerated Gradient* (NAG). No NAG, ao invés de mensurar o gradiente no ponto θ, a medida ocorre no ponto $\theta + \beta m$, como podemos ver na Equação 2.20.

$$m \leftarrow \beta m - \eta * \nabla_\theta J(\theta + \beta m) \tag{2.20}$$

A principal característica por trás do NAG é que a optimização de *momentum* já está apontada para a direção correta, então nesta técnica se usa uma localização um pouco mais a frente (uma posição próxima de θ) para moderadamente acelerar a velocidade de convergência.

2.2.5 Função de Ativação

Aprendizado supervisionado também pode ser utilizado para problemas de classificação com múltiplas classes. Estes problemas consistem na classificação do dado em no mínimo três classes diferentes. Em problemas como esse, a função de ativação da última camada é responsável por calcular a probabilidade de determinado dado pertencer à determinada classe. Neste trabalho, foi utilizada a função de ativação *softmax*, que calcula a probabilidade g da saída z pertencer à classe i dentro de K classes, como mostra a Equação 2.21:

$$g(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.21}$$

A *softmax* pode ser entendida como uma generalização da função de ativação sigmoid, mostrada na Equação 2.22, que é utilizada para casos de classificação binária.

$$g(\mathbf{x}) = \frac{e^{\mathbf{x}}}{1 + e^{\mathbf{x}}} \tag{2.22}$$

Dessa maneira, utilizar *softmax* como função de ativação para problemas de classificação com mais de uma classe se torna uma ferramenta de extrema importância para problemas classificação de dados.

2.2.6 Métricas de Avaliação de Classificação

Existem diversas métricas que podem ser usadas para cálculo de performance de algoritmos de aprendizado de máquina; todavia, é crucial mencionar as cinco que vão ser utilizadas nas seções futuras. Estas métricas são Acurácia, Precisão, Recall (ou Sensibilidade), Especificidade e F1 Score.

Quadro 1: Matriz de Confusão

		Classe Predita	
		Positivo	Negativo
Classe Real	Positivo	VP	FN
	Negativo	FP	VN

A acurácia é primariamente usada como saída categóricas e discreta, e é determinada pela Equação 2.23, onde VP representa o número de verdadeiros positivos, VN verdadeiros negativos, FP falsos positivos e por último FN o número de falsos negativos. Em resumo, é o número de acertos feitos corretamente dividido pelo número total de previsões. Esta métrica

pode ser interpretada como uma performance geral do modelo. Ela responde a pergunta: de todas as classificações feitas, quantas o modelo classificou corretamente?

$$acuracia = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.23}$$

A segunda métrica, Precisão, mostrada na Equação 2.24, é um indicativo de quão exatas as predições são, usando todos os casos positivos; uma precisão baixa significa uma grande presença de falsos positivos. Esta abordagem é particularmente útil em situações que erros nas classificações positiva são mais custosos que aqueles cometidos nas classificações negativas, como no tratamento de doenças, por exemplo. Assim, em resumo, essa métrica busca responder: de todas as classificações positivas feitas, quantas estão corretas?

$$precisao = \frac{VP}{VP + FP} \tag{2.24}$$

O Recall ou Sensibilidade, demonstrado na Equação 2.25, complementa a Precisão, pois representa a completude do modelo, ou a presença de falsos negativos, e também é comumente chamada de sensibilidade do modelo. Este método é particularmente importante em situações onde classificar resultados positivos se torna mais importante que classificar resultados negativos. Esta métrica busca compreender de todas as classes positivas com o valor esperado, quantas classificações foram corretas.

A Especifidade, demonstrada na Equação 2.26, avalia os resultados negativos obtidos, nela é calculada quantos resultados verdadeiramente negativos foram obtidos, dentre o total de resultados negativos. Dessa maneira, essa métrica tenta responder a pergunta: das classes negativas, quantas foram classificadas corretamente?

$$recall/sensibilidade = \frac{VP}{VP + FN} \tag{2.25}$$

$$especificidade = \frac{VN}{VN + FP} \tag{2.26}$$

Por último, F1 Score combina tanto a precisão quando o recall em uma métrica que captura o balanceamento entre a completude do recall e a exatidão da precisão. Esta abordagem é geralmente usada no lugar da acurácia em problemas de classificação e seu cálculo pode ser visto na Equação 2.27.

$$F1Score = \frac{2 \times precisao \times recall}{precisao + recall} \tag{2.27}$$

3 TRABALHOS RELACIONADOS

Para entender onde o estudo de classificação de estresse a partir de sinais fisiológicos se encontra na literatura acadêmica, é necessário fazer um estudo de trabalhos publicados na área, entender quais os rumos que estão sendo tomados e assim elaborar hipóteses que possam trazer contribuições para a área. Dessa maneira, a principal fonte de pesquisa sobre trabalhos relacionados utilizado por esta dissertação se encontra em (MELCHIADES et al., forthcoming). Neste estudo estão reunidos os principais e mais recentes trabalhos na área de classificação de estresse utilizando sinais fisiológicos e aprendizado de máquina, além de trazer as principais contribuições possíveis, lacunas a serem trabalhadas.

Em (MELCHIADES et al., forthcoming) é feita uma pesquisa para entender qual é o estado da arte na área de aprendizado de máquina na classificação de estresse a partir de sinais fisiológicos. Para isso este trabalho utiliza a palavra chave de busca "`stress health "machine learning-network -posttraumatic`" nos mecanismos online de busca do *Google Scholar* e no EBSCO. Além disso, este trabalho filtrou os estudos que foram publicados a partir de 2015, para assim garantir contribuições mais atuais e relevantes.

Após este procedimento, foram selecionados os trabalhos considerados mais importantes e foi aplicada a técnica de *Snowballing*, a fim de aumentar a gama e a qualidade de trabalhos a serem estudados. Assim, foi encontrado um compilado especializado e de muita valia na área e que será discutido em detalhes nas próximas seções.

3.1 Aprendizado de Máquina para Classificação de Estresse

O trabalho feito por (MELCHIADES et al., forthcoming) trouxe um panorama geral da área de classificação de estresse a partir de sinais fisiológicos utilizando aprendizado de máquina. O resumo deste panorama pode ser resumido com a tabela mostrada no Quadro 2.

Quadro 2: Panorama Geral de Aprendizado de Máquina para Classificação de Estresse. Fonte: (MELCHIADES et al., forthcoming)

Artigo	Modelo	Biomarcadores	Protocolo de Classificação de Estresse
(GJORESKI et al., 2016b)	RF, SVM	BVP, HR, GSR, ST, Accel.	Montreal Imaging Stress Task
(SAEED et al., 2020)	k-NN, NB, SVM, LR, MLP	EEG	PSS-10 questionnaire, expert labelling
(ALHITARY et al., 2018)	NB, MLP, RF, K^*	HR, PWA, GSR, Diâmetro da Pulpila, Respiração	STAI
(NATH; THAPLIYAL; CABAN-HOLT, 2021)	RF, LR, k-NN, SVM, LSTM	GSR, BVP	Cortisol, TSST
(LOTFAN et al., 2019)	SVM	EEG	TSST
(DELMASTRO; DI MARTINO; DOLCIOTTI, 2020)	BN, SVM, k-NN, C4.5 Decision Tree, RF, AdaBoost	HR, HRV, GSR	SCWT
(DI MARTINO; DELMASTRO, 2020)	NARX, RF, LSB, LSTM	Questionário Criado Pelos Autores	PANAS, STAI
(SCHMIDT et al., 2018)	DT, RD, AB, LDA, k-NN	ECG, GSR, EMG, BVP, Temp., Accel.	STAI, SAM, SSSQ
(HUYSMANS et al., 2018)	SOM	SC, ECG	SCWT, Math, Stress talk
(HOVSEPIAN et al., 2015)	SVM	ECG, Respiração	EMA, PSS adaptation
(BATTALIO et al., 2021)	SVM	ECG, Respiração, Smartphone	EMA
(SANTOS SIERRA et al., 2011)	Fuzzy inference	HR, GSR	Unspecified stimulating and threatening tasks
(PEDROTTI et al., 2014)	Neural network	Diâmetro da Pulpila	Driving test
(NAPOLETANO; ROSSI, 2018)	SVM	HR, Respiração	Simulated driving
(SEO et al., 2019)	Neural network (CNN + LSTM)	ECG, Respiração	SCWT
(KESHAN; PARIMI; BICHINDARITZ, 2015)	NB, LR, MLP, SVM, k-NN, RF	ECG	Driving test
(SUBHANI et al., 2017)	LR, SVM, NB	EEG	Mental arithmetic test
(ZHAI; BARRETO, 2006)	SVM	GSR, BVP, Diâmetro da Pulpila, Temperatura da Pele	Paced Stroop Test
(SANO; PICARD, 2013)	SVM, k-NN, PCA	Accel., Condutância da Pele, Uso de Smarphone	PSS, PSQI
(SOUZA et al., 2022)	LSTM, GRU	ECG, GSR, EMG, Resp., Temp.	WESAD dataset
(LI; LIU, 2020)	CNN, MLP	ECG, GSR, EMG, Resp., Temp.	WESAD dataset
(SIIRTOLA, 2019)	LDA, QDA, RF	GSR, Accel., Temp., BVP, HR	WESAD dataset
(MCDUFF; GONTAREK; PICARD, 2014)	SVM	HR, Respiração, HRV	Mental arithmetic test

No Quadro 2, podemos notar 3 aspectos principais:

1. Os tipos de modelos de aprendizado de máquinas utilizados para classificação de estresse;
2. Os sinais fisiológicos utilizados para fazer a detecção de estresse;
3. O tipo de protocolo utilizado para induzir estresse nos voluntários dos experimentos.

Todos esses aspectos são importantes para contextualizar a área de aprendizado de máquinas na classificação de estresse, bem como entender quais são as possíveis lacunas e como este trabalho visa preenchê-las.

3.2 Modelos de Aprendizado de Máquina

Analisando a primeira coluna do Quadro 2, temos uma lista dos algoritmos de aprendizado de máquina utilizados para classificar estresse encontrados na literatura. O primeiro aspecto a se mencionar dessa lista é a simplicidade da maioria dos modelos utilizados.

Considerando esse aspecto de simplicidade, é possível observar que existem algoritmos que podem ser considerados como modelos estatísticos como *Principal Component Analysis* (PCA), utilizado em (SANO; PICARD, 2013) ou Regressão Linear (LR), utilizado em (SAEED et al., 2020), (NATH; THAPLIYAL; CABAN-HOLT, 2021), e (SUBHANI et al., 2017).

Observando modelos mais complexos, é possível observar um alto uso de modelos clássicos de aprendizado de máquinas, como *Random Forest* (RF), *Decision Tree* (DT), *Support Vector Machine* (SVM), *Naive Bayes* (NB), *k Nearest Neighbours* (k-NN) e outros.

É importante notar que muitos desses modelos clássicos de aprendizado de máquinas precisam utilizar características específicas dos sinais como dados de entrada. Por isso, antes de alimentar tais modelos, muitas vezes é necessária uma extração das principais características dos dados coletados para depois utilizar de fato o modelo.

A extração das principais características de um dado para alimentar um modelo clássico de aprendizado de máquina constitui uma grande limitação de tais modelos. Tendo em vista a vasta quantidade de sinais que podem ser utilizados e as diversas características que cada sinal possui, é possível que a extração características obtenha informações que são de pouca relevância para a classificação de estresse e não obtenha características mais importantes. Assim, utilizar modelos que exigem uma escolha manual de características do dado, o que constitui uma grande limitação desses modelos e por isso, as redes neurais são mais adequadas para classificação de estresse.

Ao analisar os modelos mais complexos da lista, é possível observar algumas poucas ocorrências do uso de redes neurais. Tais modelos são encontrados em (SAEED et al., 2020), (ALHITARY et al., 2018), (KESHAN; PARIMI; BICHINDARITZ, 2015) e (LI; LIU, 2020) na forma de uma *Multilayer Perceptron* (MLP), em (NATH; THAPLIYAL; CABAN-HOLT, 2021), (DI MARTINO; DELMASTRO, 2020), (SEO et al., 2019) e (SOUZA et al., 2022) como uma Rede Neural Recorrente (LSTM ou GRU) e em (SEO et al., 2019) e (LI; LIU, 2020) como uma Rede Neural Convolucional (CNN).

É importante notar que todos os modelos aqui citados são modelos de aprendizagem supervisionada. Tendo em vista que os experimentos que envolvem estresse necessitam de um protocolo de indução de estresse, é natural que para esses casos existam profissionais da saúde envolvidos, de maneira que o experimento possa ocorrer de maneira apropriada e também as classes possam ser rotuladas de maneira correta. Portanto é importante mencionar o trabalho de (HUYSMANS et al., 2018), que utilizou *Self-Organizing Maps* (SOM) para a classificação de estresse, ou seja, um algoritmo não supervisionado.

Em (HUYSMANS et al., 2018), foram realizados experimentos com induções alternadas de estresse e relaxamento para que fossem coletadas da condutância da pele e do eletrocardiograma dos participantes. Após essa etapa, foi montada a estrutura do algoritmo SOM para que fosse treinado em características específicas dos sinais de condutância e eletrocardiograma, em seguida a essa etapa, foi utilizado um modelo *Gaussian Mixture* para agrupar as regiões de características similares obtidas pelo treinamento do SOM e por fim foi feita uma comparação de principais características de cada grupo para determinar se tal grupo correspondia a fase de estresse ou de relaxamento.

Este experimento obteve uma performance de 79.0% com uma sensitividade de 75.6%, ou seja, valores bons para classificação e este trabalho tem a vantagem de apenas classificar agrupamentos gerados pelos sinais fisiológicos, ou seja, em um experimento em ambiente menos controlado, é possível que possam ser detectados diversos agrupamentos diferentes, não só estresse e relaxamento e tais agrupamentos podem ajudar a entender melhor como os sinais fisiológicos se comportam ou se organizam em diferentes situações das pessoas.

Portanto, observando os modelos e as técnicas de aprendizado de máquina utilizados na literatura, podemos citar duas principais lacunas:

1. A falta de estudos com algoritmos não supervisionados;
2. A falta de algoritmos de aprendizado de máquina que representem o estado da arte na utilização da tarefa de detecção de estresse.

Este trabalho visa preencher a segunda lacuna, e para isso esta dissertação propõe um novo modelo de aprendizagem profunda, o MoStress, descrito em detalhes na Seção 4. Este novo modelo é composto por uma etapa de pré-processamento e pode ser utilizado com diversos tipos de redes neurais. Nos experimentos desse trabalho, as redes utilizadas foram as Redes Neurais Recorrentes, as *Echo State Networks* e uma adaptação do NBeats para a classificação de múltiplas séries temporais, ou seja, esta dissertação utiliza modelos que representam o estado da arte em redes neurais para a classificação de estresse e por isso preenche a segunda lacuna.

3.3 Sinais Fisiológicos para Classificação de Estresse

Além do panorama geral do estado da arte de aprendizado de máquina na classificação de estresse a partir de sinais fisiológicos, em (MELCHIADES et al., forthcoming) podemos en-

contrar os diversos sinais que já foram utilizados para executar tal tarefa. Assim, nas terceiras e quartas colunas do Quadro 2, é possível observar os principais sinais fisiológicos utilizados pelos algoritmos de aprendizado de máquina e também a maneira pela qual o estresse foi induzido.

Observando a lista de protocolos de indução de estresse na última linha do Quadro 2, podemos observar diversos tipos de protocolos diferentes. Alguns deles utilizam imagens para indução de estresse, como em (GJORESKI et al., 2016b), outros utilizam tarefas aritméticas, como (SUBHANI et al., 2017) e (MCDUFF; GONTAREK; PICARD, 2014) e alguns utilizaram métodos menos comuns como testes de direção, utilizado em (KESHAN; PARIMI; BICHINDARITZ, 2015) e (PEDROTTI et al., 2014).

Apesar de existirem diversos métodos de indução de estresse, também podemos observar que a maioria dos estudos se baseia em protocolos já consolidados para essa tarefa como o *Trier Social Stress Test* (TSST) (KIRSCHBAUM; PIRKE; HELLHAMMER, 1993), o *Positive and Negative Affect Scale* (PANAS) (WATSON; CLARK; TELLEGEN, 1988) ou o*State-Trait Anxietry Inventory* (STAI) (SPIELBERGER, 1970). Estes protocolos são considerados "padrão ouro"na indução de estresse, o que significa que os resultados da indução por tais testes são de grande exatidão e conseguem levar os voluntários verdadeiramente a estados de estresse.

Dessa maneira, este trabalho também busca utilizar dados que tenham sido obtidos por protocolos "padrão ouro"de indução de estresse. Aliado a este fato, essa dissertação também procura usar dados fisiológicos que possam ser medidos por sensores vestíveis, sendo essa uma condição não primordial para a escolha do dados, porém que deve ser levada consideração. Pois como explanado na Seção 1.1, sensores vestíveis estão se tornando cada vez mais populares, de maneira que utilizar os dados coletados por tais sensores pode ajudar também a popularizar a detecção de estresse na vida cotidiana.

Portanto, este trabalho utiliza os conjunto de dados público WESAD (SCHMIDT et al., 2018), que também é encontrado no Quadro 2. Este conjunto de dados é constituído por dados fisiológicos coletados por sensores vestíveis posicionados no tórax e no pulso de cada participante, de maneira que os principais protocolos de indução de estresse foram utilizados em cada voluntário, gerando assim um conjunto de dados com os diversos sinais temporais e com as classes de *baseline*, *stress* e *amusement*. Este conjunto de dados será descrito em detalhes na Seção 5.1.1.1.

Observando os trabalhos que utilizaram o conjunto de dados do WESAD, vemos três trabalhos diferentes. O primeiro deles é o (SOUZA et al., 2022), que faz parte dessa dissertação e será explicado em detalhes nas próximas seções, o segundo é o (LI; LIU, 2020), que utilizou CNNs para classificar estresse e o terceiro é o (SIIRTOLA, 2019), que utilizou modelos estatísticos e algoritmos clássicos de aprendizado de máquina para classificação de estresse.

Dessa forma, concluir que existem poucos trabalhos com redes neurais utilizando os dados do WESAD. Também é interessante notar que apesar desse conjunto de dados ser constituído por séries temporais, o trabalho de (LI; LIU, 2020) utilizou CNNs, redes comumente utilizadas

para tratamento de imagens. Dessa maneira, esse trabalho é importante de ser aprofundado e isso será feito na próxima seção.

3.4 Classificação de Estresse com o Uso de CNNs

As *Convolutional Neural Networks* são comumente utilizadas para classificação de imagens. Estas redes são constituídas de matrizes que são aplicadas à matriz de pixeis de uma imagem de maneira a extrair as principais características da imagem. Uma vez que as principais características foram obtidas, utiliza-se essas informações em uma *MultiLayer Perceptron* (MLP) para que assim possa se fazer a classificação da imagem.

No estudo que publicou WESAD, os pesquisadores também trabalharam na classificação de estresse, e para isso tentaram extrair as principais características dos sinais coletados e utilizar em modelos clássicos de aprendizado de máquina. Porém, como explicado na Seção 3.2, essa extração de características de um sinal constitui uma grande limitação quando se trata de problemas de classificação, pois é possível que as características extraídas não sejam as mais relevantes para o problema proposto.

Assim, a principal característica do trabalho de (LI; LIU, 2020) é utilizar uma CNNs nos dados do WESAD para extrair as principais características de cada sinal e assim utilizar uma MLP para fazer a classificação de estresse. A Figura 4 mostra a rede utilizada por (LI; LIU, 2020) para os dados do WESAD coletados pelo sensor do tórax.

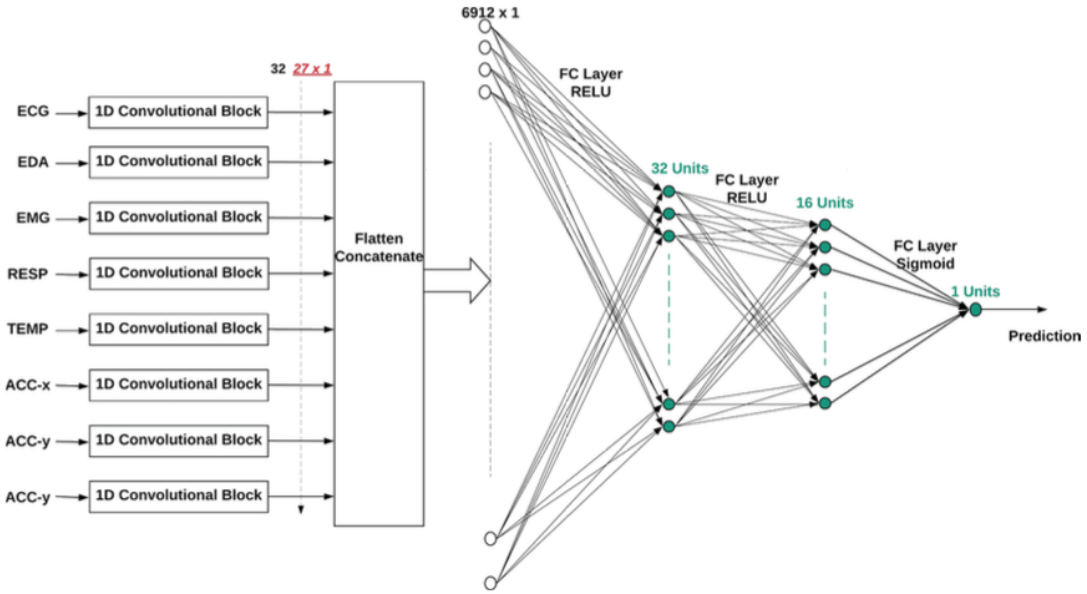


Figura 4: CNN Utilizada nos Dados do WESAD. Fonte: (LI; LIU, 2020)

Como podemos ver na Figura 4, essa rede utilizou um bloco convolucional de uma dimensão para cada sinal do conjunto de dados. Assim, cada bloco convolucional extraiu a principal característica de cada sinal e essas informações foram concatenadas e utilizadas por uma MLP, para assim fazer a classificação.

Essa abordagem gerou bons resultados de classificação, com 99.80% e 99.55% de acurácia para classificação binária e para a classificação de 3 classes, respectivamente. Um resultado melhor do que o obtido pelos próprios pesquisadores do WESAD.

O trabalho de (LI; LIU, 2020) alcançou bons resultados de classificação com os dados do WESAD, porém, CNNs não são as redes mais comuns de se utilizar ao se trabalhar com séries temporais. Portanto, existe uma lacuna no uso de redes neurais próprias para se trabalhar com séries temporais com os dados do WESAD.

Essa lacuna é preenchida pelo modelo proposto nessa dissertação com MoStress, modelo esse que será descrito em detalhes nas próximas seções.

4 MOSTRESS

MoStress é o nome do modelo sequencial criado para a classificação de estresse. Esse modelo é considerado sequencial pelo fato de não ser constituído apenas de um algoritmo de aprendizado de máquina, mas também uma série de etapas que visam limpar e melhorar a qualidade dos dados para assim facilitar o aprendizado do modelo.

Tendo em vista que séries temporais podem gerar quantidade de dados muito grandes, é de extrema importância que elas sejam tratadas e transformadas da melhora maneira, visando uma melhor qualidade dos dados e uma melhor performance do modelo. Como descrito em (GUDIVADA; APON; DING, 2017), esta etapa é de crucial importância para a maioria dos modelos de aprendizado de máquina, e nesse aspecto, o MoStress não constitui exceção.

Portanto, consideramos que a etapa de pré-processamento de dados deve sempre acontecer antes de alimentar um modelo de aprendizado de máquina. Essa etapa é basicamente a única constante em todos os experimentos que serão descritos na Seção 5 e por isso constitui parte do MoStress, tornando-o um modelo sequencial.

4.1 Visão Geral

A arquitetura do MoStress se divide em duas partes: a parte de pré-processamento e a parte de rede neural profunda. O pré-processamento do MoStres visa melhorar a qualidade dos dados aplicando quatro técnicas diferentes:

- Transformada de Fourier, descrito na Seção 4.2.1;
- Normalização *Rolling Z-Score*, descrito na Seção 4.2.2;
- Classificação de Janelas, descrito na Seção 4.2.3;;
- Cálculo de Pesos para Classes, descrito na Seção 4.2.4.

Cada uma das 4 técnicas mencionas, tem por objetivo melhorar os dados e calcular parâmetros de maneira que a arquitetura possa desempenhar melhor na tarefa de classificação, essas técnicas serão exploradas em detalhes na Seção 4.2

A segunda parte do MoStress consiste em uma arquitetura de rede neural profunda, responsável por obter os resultados de classificação. Atualmente foram implementadas 3 arquiteturas diferentes:

- Rede Neural Recorrente, descrito na Seção 4.3.1;
- *Echo State Network*, descrito na Seção 4.3.2;
- NBeats com a adição de uma *Multilayer Perceptron* (MLP), descrito na Seção 4.3.3.

Tais redes recebem como entrada os dados pré-processados e um vetor de pesos calculados para cada classe, de maneira que esses dados de entrada são os dados de saída da etapa de pré-processamento. As redes neurais implementadas serão exploradas em mais detalhes na Seção 4.3.

É importante notar que para o NBeats também foi adicionada uma segunda rede MLP, pois este modelo foi projetado para fazer predição de uma série temporal e a MLP foi utilizada como complemento da arquitetura para que pudesse ser feita a classificação de múltiplas séries temporais.

A Figura 5 mostra uma visão geral do Mostress. Nesta figura é possível ver as todas as etapas de pré-processamento citadas acima, bem como os modelos de redes neurais que foram utilizados na implementação do MoStress.

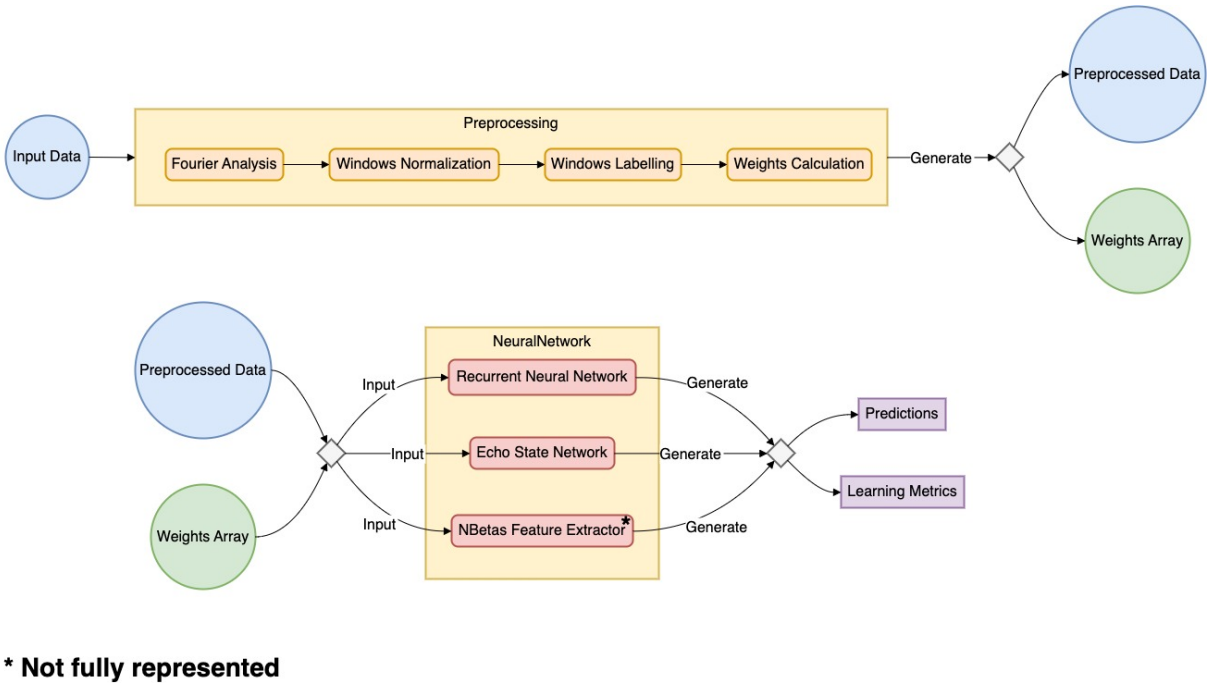


Figura 5: MoStress, Visão Geral

Na Figura 5 também é possível notar que a etapa de pré-processamento não gera apenas os dados pré-processados, mas também um vetor de pesos referente a cada classe referente ao aos dados de entrada. Esse vetor também pode ser utilizado como entrada dos modelos de redes neurais para assim poder ajudar no desbalanceamento de dados, como vai ser detalhado na Seção 4.2.4.

Outro aspecto dessa figura se encontra no fato de que o *NBeats Feature Extractor* não está completamente representado neste diagrama. Isso porque neste modelo, existem ainda etapas intermediárias antes de poder ser gerado as predições e as métricas de aprendizado.

Mais detalhamentos do experimentos do NBeats, bem como detalhamento dos experimentos feitos com outros modelos serão descritos na Seção 5. Assim, as próximas seções buscam entrar em detalhes do MoStress, tanto sobre a etapa de pré-processamento como as arquiteturas de

redes neurais implementadas.

4.2 Pré-Processamento de Dados

O objetivo da etapa de pré-processamento do MoStress é de limpar os dados de entrada e fornecer melhores informações à rede neural, de maneira que se possa obter bons resultados de classificação. Além disso, o MoStress foi projetado de maneira que uma janela de dados fosse utilizada como entrada da rede neural. Essa característica permite que o tamanho das janelas sejam ajustados de maneira que sempre exista uma quantidade suficiente para garantir bons resultados de aprendizagem da rede neural. Assim, para treinar uma instância de MoStress, foi necessário utilizar um método para lidar com as janelas de entrada das redes.

O método escolhido para gerar as entradas de dados da rede neural e que será usado em todo o pipeline de pré-processamento é o de janelas móveis. Neste método adota-se uma quantidade de amostras de sinais dentro de uma janela temporal e, a partir disso, move-se esta janela com um passo determinado até o próximo janelamento. O tamanho da janela e o passo utilizado vão ser descritos em detalhes na Seção 5.

Assim, a etapa de pré-processamento do MoStress, que é mostrado na Figura 6, tem por objetivo endereçar 4 problemas que podem impactar o desempenho de aprendizado da rede neural:

- Sinal de entrada ruidoso: problema endereçado pela Transformada de Fourier;
- Dados em diferentes escalas: problema endereçado pela Normalização *Rolling Z-Score*;
- Classificação das janelas pré-processadas: problema endereçado pela Classificação de Janelas por Frequência de Classes;
- Dados altamente desbalanceados: problema endereçado pelo Cálculos de Pesos por Classe.

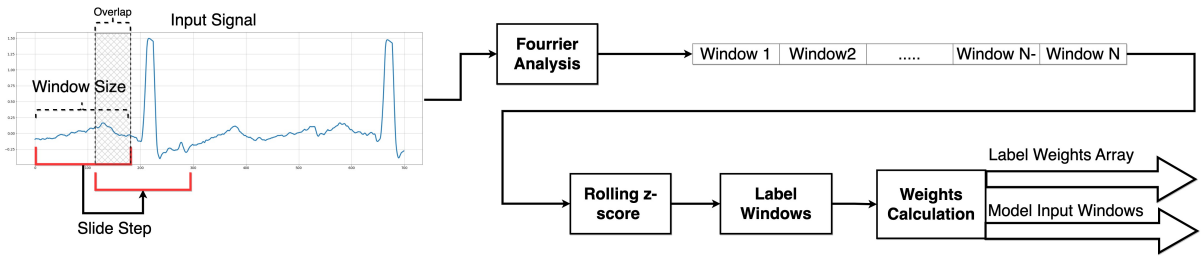


Figura 6: MoStress Etapa de Pré-processamento

Todos esses quatro problemas citados acima, podem ter um grande impacto no aprendizado final do modelo. Ruídos e dados com diferentes escalas tornam mais difícil os ajustes dos parâmetros da rede. Rotular incorretamente uma janela, especialmente se for uma janela

de transição, pode gerar classificações erradas do modelo. Por último, dados desbalanceados tornam mais difícil para o modelo fazer boas classificações das classes com menos dados. Cada um desses problemas vai ser discutido em mais detalhes nas próximas seções.

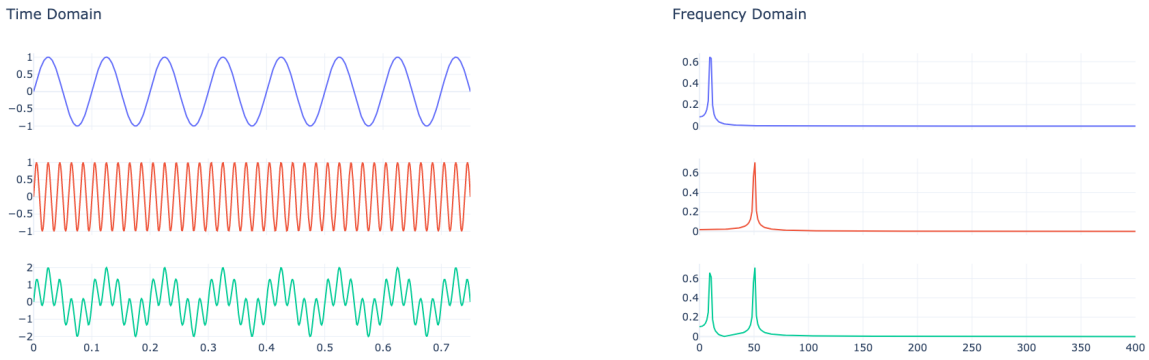
4.2.1 Análise de Fourier

A Transformada de Fourier é uma técnica utilizada para decompor sinais temporais encontrando os componentes temporais que compõem o sinal original. Esta transformada leva o sinal temporal analisado para o domínio da frequência onde é possível observar os diversos espectros de frequência que o compõem, como explica (SNEDDON, 1995).

Assim, se considerarmos um sinal temporal descrito por $f(x)$, e com o período P, podemos descrever a amplitude e a fase de tal sinal na frequência $n/P, n\epsilon\mathbb{Z}$ como na Equação 4.1

$$c_n = \frac{1}{P}\int_{-\infty}^{\infty} f(x)e^{-i2\pi\frac{n}{P}x}dx \tag{4.1}$$

Portanto, se pegarmos dois sinais diferentes e somarmos, como na Figura 7a, podemos aplicar a Transformada de Fourier em cada um deles e perceber que no espectro de frequência do sinal resultante, temos a soma das frequências dos sinais que compõem o sinal resultante, como podemos observar na Figura 7b.



(a) Sinais no Domínio do Tempo

(b) Sinais no Domínio da Frequência

Figura 7: Aplicação da Transformada de Fourier

A Figura 8 mostra a limpeza de um sinal de Temperatura utilizando a Transformada de Fourier.

Dessa maneira, é possível utilizar a Transformada de Fourier no espectro da frequência, entender quais são as frequências dominantes que compõem o sinal e assim passar um filtro que remova as frequências indesejadas do sinal original. Retirando assim os ruídos e gerando um dado melhor de ser processado por qualquer modelo.

(a) Sinal sem Limpeza de Frequência (b) Sinal com Limpeza de Frequência

Figura 8: Aplicação da Limpeza do Sinal Utilizando Transformada de Fourier

4.2.2 Normalização de Dados

A segunda etapa no pré-processamento dos dados é a normalização. Uma vez que foi escolhido uso de janelas móveis, a normalização também vai acontecer com os dados dentro de cada janela.

Sinais fisiológicos podem ter diferentes padrões entre si e por isso podem ter grande variação de escalar. Podemos citar como exemplo da temperatura corporal, que possui um nível médio de 35°Celsius, enquanto o sinal de eletrocardiograma possui um nível médio centrado no zero.

Por isso, é importante buscar uma normalização que deixe os dados em uma escala padrão, uma escala normalizada, porém não mude o padrão das séries temporais e principalmente não altere as estatísticas da série analisada.

Dessa maneira, tendo em vista que sinais fisiológicos muitas vezes possuem caraterísticas não estacionárias, ou seja, estes sinais estão alterando suas estatísticas no decorrer do tempo, a escolha de um método de normalização apropriado se torna ainda mais importante.

Portanto, como temos de preservar os padrões dos sinais temporais e tais sinais podem ainda ser não estacionários, foi escolhida uma normalização baseada na média e no desvio padrão, que é a normalização z-score (DEVORE, 2017) aplicada em cada janela. A normalização *Z-Score* é mostrada na Equação 4.2:

$$z = \frac{x - \mu}{\sigma} \tag{4.2}$$

Onde x é um ponto dentro da janela, μ é a média dos pontos na janela, e σ é o desvio padrão.

A Figura 9 mostra um sinal de temperatura após a aplicação da normalização *Z-Score* por janelas. É importante notar que a principal diferença entre as Figuras 9a e 9b é a escala do sinal.

(a) Sinal sem Normalização (b) Sinal com Normalização

Figura 9: Aplicação da Normalização *Z-Score*

Com a normalização, os *outliers* de cada sinal continuam sendo considerados no sinal resultante, não alterando o padrão de sinal temporal de entrada. Após levar cada janela a média zero e desvio padrão um, todos os sinais vão estar dentro de limite de valores centrado em zero, portanto, normalizados.

4.2.3 Classificação de Janelas

Após a normalização das janelas utilizando *Rolling Z-Score*, é necessário que seja atribuída uma classe para cada janela e isso foi feito a partir de uma contagem de frequências de cada classe dentro de cada janela. Para exemplificar, considere a situação demonstrada na Figura 10, onde vemos o exemplo de um sinal temporal em que estão presentes 3 classes diferentes: *amusement*, *baseline* e *stress*.

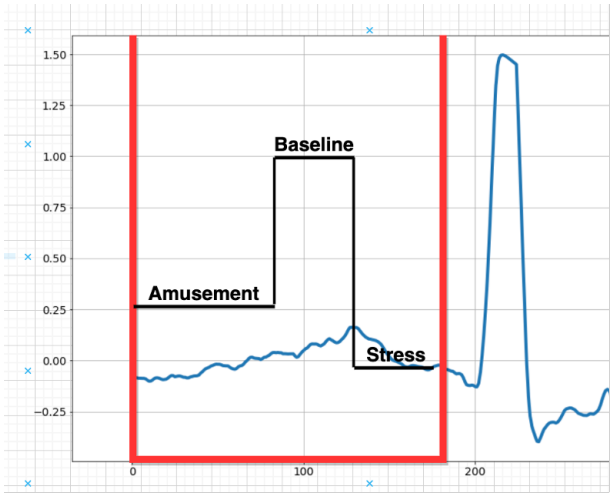


Figura 10: Três Classes em Uma Janela que Representa o Estado de *amusement*

Para determinar qual das classes vai representar a janela da Figura 10, foram realizadas as etapas abaixo:

- Calcular a frequência de cada classe presente dentro da janela;
- Determinar que a classe com maior frequência será a classe atribuída à janela;
- Descartar as janelas em que a frequência é menor que um valor χ.

O último dos 3 passos foi feito com o intuito de evitar que sejam entregues à rede neural artificial estados transitórios. Tendo em vista que neste exemplo o modelo quer classificar entre *baseline*, *stress* e *amusement*, a rede neural precisa ter seus parâmetros ajustados para estes estados específicos. Desta maneira, como estados de transição não podem ser bem classificados em uma dessas classes, optou-se por não utilizar janelas de transição e descartá-las, para que assim não fosse comprometido o resultado do modelo.

Assim, para o exemplo aqui mencionado, a janela seria classificada como *amusement*, desde que a frequência dessa classe fosse maior que o limiar χ, pois, caso contrário, essa janela seria descartada.

4.2.4 Dados Desbalanceados

Por fim, o pré-processamento de dados do MoStress utiliza o cálculo de pesos mostrado na Equação 4.3 para gerar um peso para cada classe e assim ajudar o modelo a lidar com classes que possuem menos ocorrências. Dessa forma, classes com menos ocorrências os maiores pesos serão atribuídos.

$$weight(x) = \frac{\#Samples}{\#Classes \times \#OccurrencesOfClassX} \tag{4.3}$$

Existem diversas outras maneiras de se lidar com o desbalanceamento de dados como por exemplo: utilizar *oversampling* ou *undersampling* com diferentes métodos de interpolação de para gerar um conjunto de dados mais balanceado. Não obstante tais técnicas, muitas delas alteram os dados originais ou dependem de sua estrutura para serem utilizadas. Sendo assim, a maneira com que o MoStress lida com o desbalanceamento de dados é mais geral e pode ser utilizada na maioria dos problemas de classificação de séries temporais.

Assim esta técnica encerra a etapa de pré-processamento de dados do MoStress, que terá sempre como saída os dados tratados e um vetor de pesos para cada classe.

4.3 Rede Neural

Além da etapa de pré-processamento, o MoStress também possui uma rede neural associada. Tal arquitetura deve ser adaptada de modo à receber as janelas de séries temporais que são geradas na etapa anterior. Dessa maneira, atualmente foram implementadas 3 tipos diferentes de redes neurais no MoStress, que são as seguintes:

- Redes Neuras Recorrentes - RNN
- *Echo State Neural Networks* - ESN
- NBeats

Como mencionado na Seção 3, existe uma lacuna na utilização de RNNs com os dados do WESAD (SCHMIDT et al., 2018), e a primeira rede visa preencher essa primeira lacuna. Somado a esse aspecto, também foram implementadas as duas outras redes acima mencionadas, para que possamos estudar como os modelos do estado da arte na área de aprendizado de máquina se comportam no problema de classificação de estresse.

As ESN's foram escolhidas como um representante de algoritmo do estado da arte na área de aprendizado de máquinas por representar uma diferente configuração das redes neurais. Como descrito na Seção 2.2.2, estas redes representam um paradigma diferente das redes neurais regulares. Já o NBeats foi escolhido por ser um modelo já consagrado na predição de séries temporais e por isso este trabalho tenta fazer a adaptação de tal modelo para classificação de séries temporais que representam o estresse.

É importante notra que cada uma dessas redes possuem hiper-parâmetros próprios que devem ser configurados de acordo com o problema de classificação a ser resolvido. Portanto, essa seção se propõe apenas a especificar a arquitetura implementada no MoStress. Assim, a otimização de hiper-parâmetros da RNN, da ECN e do NBeats serão detalhadas nos experimentos feitos para cada rede e que são descritos em detalhes nas Seções 5.1.2.3, 5.1.2.4 e 5.1.2.5, respectivamente

4.3.1 Rede Neural Recorrente

As RNNs são comumente utilizadas para lidar com séries temporais. Os neurônios que compõem esta rede, como as GRU e as LSTM, no momento da aprendizagem levam em conta um pedaço determinado da série de dados e isso ajuda com o problema da diminuição do valor do gradiente, como bem explica (GOODFELLOW; BENGIO; COURVILLE, 2016).

Dessa maneira, a RNN utilizada no MoStress consiste de uma camada de neurônios RNN de entrada, juntamente de uma camada de saída com neurônios Dense, assim como mostra a Figura 11.

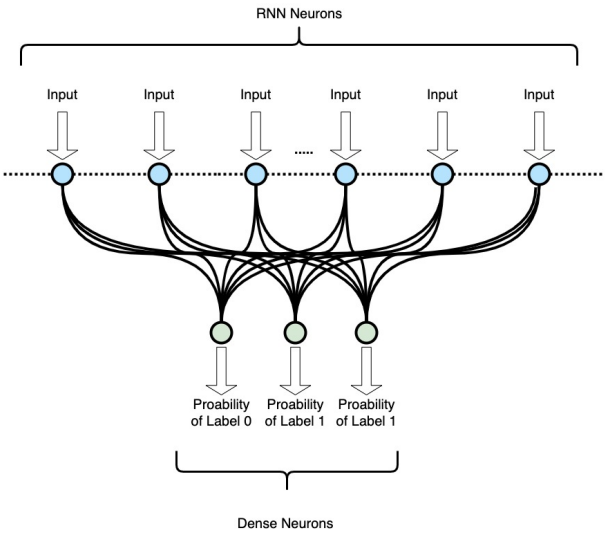


Figura 11: Rede Neural do MoStress: RNN

É importante notar que essa arquitetura deve ser adaptada ao problema de classificação no qual será usada. A Figura 11, por exemplo, mostra uma arquitetura adaptada para um problema de classificação com três classes diferentes. No caso de um número diferente de classes, a

camada de saída terá uma quantidade diferente de neurônios e assim a arquitetura da rede será alterada.

4.3.2 *Echo State Neural Networks*

As ESN's são utilizadas da mesma maneira que as RNNs, porém, os neurônios internos da rede são dispostos e interligados de maneira aleatória e as ligações entre todos os neurônios, tanto de entrada, saída ou os internos, é inicializada também de maneira aleatória. Nesse caso, o algoritmo de *backpropagation* atua para ajustar apenas os pesos de saída da rede para assim fazer a classificação.

Portanto, a arquitetura da ESN no MoStress é muito semelhante à da RNN anteriormente descrita, ou seja, temos os neurônios de entrada, os neurônios de RNN internos e os neurônios de saída correspondentes ao número de classes no problema de classificação, assim como mostra a Figura 12.

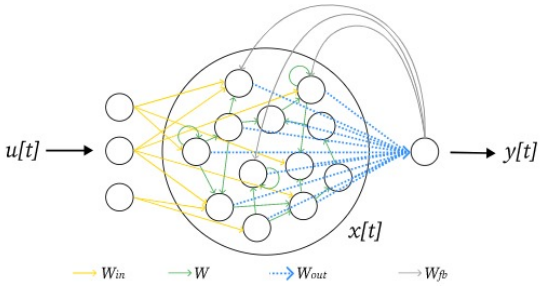


Figura 12: Rede Neural do MoStress: ESN. Fonte: (TROUVAIN et al., 2020)

E no caso da Figura 12, estamos falando de um problema de classificação binária, tendo em vista o único neurônio de saída e evidentemente, a rede deve ser adaptada para diferentes problemas de classificação.

4.3.3 NBeats

O caso do NBeats para o MoStress é diferente em relação às duas outras arquiteturas descritas. O que fez com que fossem necessárias certas adaptações no funcionamento do MoStress para que fosse implementado esta arquitetura.

Primeiramente é importante comentar que o Nbeats foi desenvolvido para ter apenas uma série temporal como dados de entrada, já os outros modelos implementados podem utilizar diversas séries temporais ao mesmo tempo. Segundo que o Nbeats é um modelo pensado para predição de séries temporais e não classificação.

Dessa maneira, para adaptar tal arquitetura para trabalhar com classificação de múltiplas séries temporais, no MoStress o NBeats aparece sempre em conjunto de uma segunda rede neural que possa ser capaz de fazer classificação de múltiplas séries temporais.

Assim, uma instância de NBeats é treinada para cada série temporal envolvida no problema e depois essas séries temporais são mais uma vez introduzidas no modelo do NBeats de maneira que possam ser coletados os resíduos de cada bloco. Assim, após a coleta de resíduos, esses dados serão utilizados como entrada da segunda rede neural para que assim possa fazer a classificação.

A Figura 13 mostra como o NBeats é utilizado dentro do MoStress.

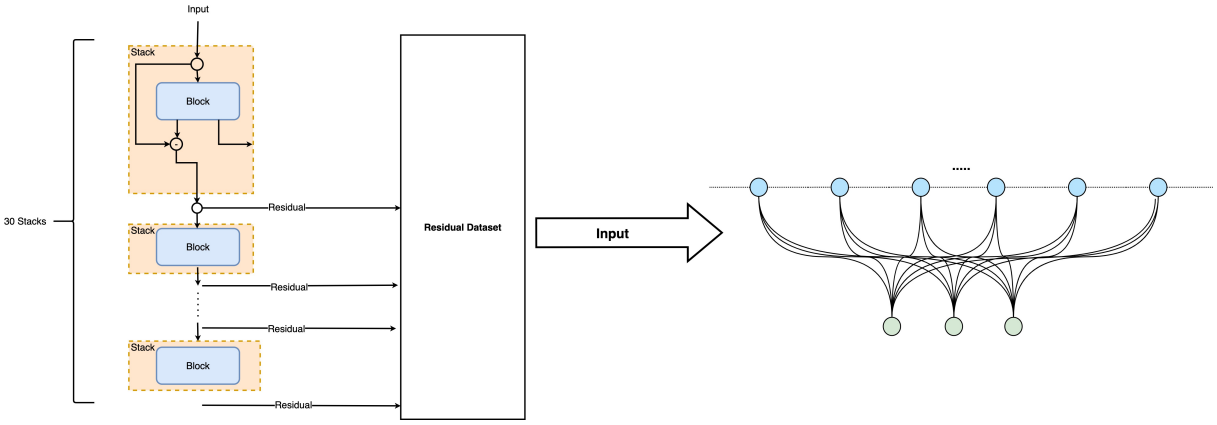


Figura 13: Rede Neural do MoStress: NBeats

Aqui é importante notar que dentro de cada pilha do Nbeats existe apenas um bloco e ao total existem 30 blocos diferentes. Esta configuração foi escolhida pois para o MoStress está sendo usado o N-BEATS-G na configuração com os melhores resultados obtidos por (ORESHKIN et al., 2019). Para ajustar os diferentes problemas de classificação dentro do MoStress é preciso ajustar a segunda rede neural de acordo com as características do problema estudado.

4.4 Avaliação de Resultados

A validação e resultados apresentados pelo MoStress são coletados a partir de dados nunca antes utilizados pela rede neural. O diagrama da Figura 5 mostra que os dados de entrada, após serem pré-processados na primeira etapa do MoStress, são divididos em dados de treinamento e de validação.

Como o próprio nome diz, os dados de treinamento são utilizados para treinar a rede neural, e desse treinamento podemos obter as curvas de aprendizado e também o modelo com os pesos atualizados. Os dados de validação são utilizados para que a rede neural tente fazer previsão acerca desses dados e assim possamos colher a matriz de confusão da classificação e assim derivar as métricas de avaliação do modelo.

A maneira como os dados originais são divididos em treino e validação é feita de maneira aleatória e depende da estrutura de cada conjunto de dados, no caso dos experimentos com o MoStress realizados nesse trabalho, foram retirados os dados completos de um dos pacientes para serem usados como validação e os dados dos outros paciente foram unificados em um só e

utilizados para o treinamento.

Portanto, o MoStress foi implementado de maneira que o pesquisador ou desenvolvedor que tiver interesse de utilizá-lo possa implementar seu próprio esquema de separação de dados para treino e validação, devendo apenas garantir que esses dados sejam janelas de séries temporais para que possam ser utilizadas nas redes neurais.

5 AVALIAÇÃO EXPERIMENTAL

Para validar o MoStress e sua arquitetura, diversos experimentos foram executados de maneira a validar algumas hipóteses e alcançar os objetivos propostos por esse trabalho. Assim, as próximas seções vão descrever em detalhes todos os experimentos feitos, discutir seus resultados e suas limitações, de maneira que se possa chegar mais perto da resposta de nossa pergunta de pesquisa. Para isso, os experimentos aqui descritos tem por objetivo validar a seguinte hipótese: é possível utilizar séries temporais que representam sinais fisiológicos associados ao estresse agudo para alimentar uma rede neural e assim obter uma detecção de estresse do indivíduo.

5.1 Metodologia

A implementação do MoStress foi feita na linguagem *Python* na versão 3.9 e o *framework* de aprendizado de máquinas utilizado foi o *Tensorflow* na versão 2.10. Para a implementação das ESN's foi utilizado o *framework ReservoirPy* (TROUVAIN et al., 2020), tendo em vista que até o momento tais redes não existem implementadas no *Tensorflow*. Além disso, diversas outras bibliotecas mais comuns foram utilizadas para a implementação MoStress, a lista completa de bibliotecas utilizadas, juntamente com a implementação do modelo, podem ser encontradas em detalhes no repositório do MoStress[1] do grupo RamosAI.

Os experimentos foram executados no *cluster* da Unisinos, onde foi utilizado um *node* que possui as seguintes especificações:

- Processador AMD Epyc com 16 cores e 32 threads;
- 64 GB de memória RAM;
- GPU Tesla T4 com 16 GB de memória.

Os dados utilizados em todos os experimentos foram os dados do WESAD (SCHMIDT et al., 2018), que serão descritos em detalhe na Seção 5.1.1.1, que ao todo ocupam aproximadamente 20 GB de memória, caracterizando, assim, uma grande quantidade de dados a ser processada e tendo o risco de extrapolar a memória ou a GPU do *cluster* utilizado. Dessa maneira, foi necessário implementar um bom gerenciamento de memória no código do MoStress e em um experimentos até fazer uma divisão dos dados, porém, tais aspectos serão discutidos em detalhes nas próximas seções.

[1] github.com/ramos-ai/MoStress

5.1.1 Descrição do Trabalho do WESAD

A principal fonte de dados para a realização dos experimentos foi o banco de dados público do WESAD (SCHMIDT et al., 2018), que consiste em dados fisiológicos coletados de estudantes da Universidade de Siegen, na Alemanha, e que foram submetidos a diversos protocolos de indução de estresse. O critério de exclusão de candidatos utilizado no estudo foi: gravidez, tabagismo, pessoas que sofrem de transtornos mentais e doenças cardiovasculares. O número total de indivíduos no experimento foi de 17 pessoas com idades entre 25 e 29 anos de idade. Todavia, devido ao mau funcionamento de um dos sensores, apenas os dados de 15 indivíduos são próprios para uso.

Tendo em vista que esse foi o principal trabalho utilizado como base e como fonte de dados para esse estudo, as próximas seções vão descrever como os dados foram coletados e também vão descrever os modelos que os pesquisadores utilizaram para fazer classificação de estresse.

5.1.1.1 Dados Utilizados

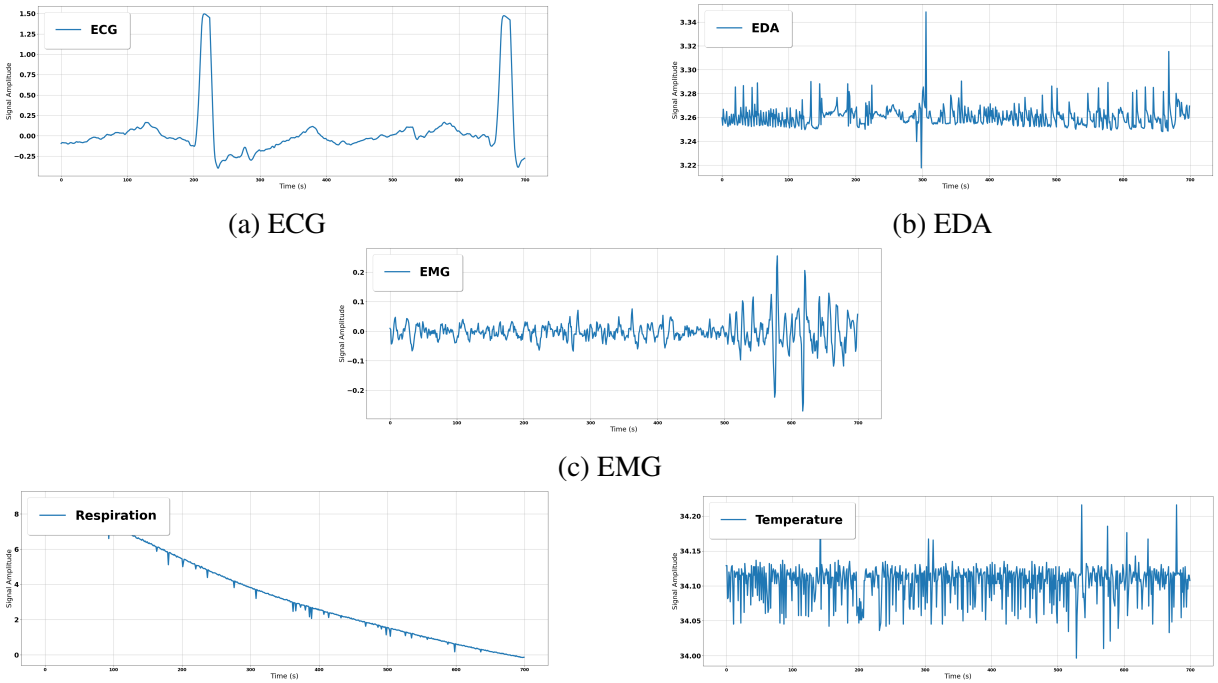
O estudo do (SCHMIDT et al., 2018) usou dois dispositivos. O primeiro dispositivo foi alocado no tórax do paciente e este coletava o movimento através de um acelerômetro em 3 eixos (ACC), respiração (RESP), eletrocardiograma (ECG), atividade eletrodermal (EDA), eletromiograma (EMG) e temperatura (TEMP), todos coletados com uma frequência de 700Hz. O outro dispositivo foi colocado no punho dos pacientes, e este coletou pressão da corrente sanguínea (BVP) a uma frequência de 64 Hz, EDA a uma frequência de 4Hz, TEMP a uma frequência de 4Hz e um ACC a uma frequência d 32 Hz. Ambos os dispositivos foram manualmente sincronizados através de um gesto de duplo tapa no início do experimento. Uma amostra coletada de um paciente e referente à um segundo de coleta pode ser visto na Figura 14.

O protocolo de estudo do (SCHMIDT et al., 2018) tem por objetivo evocar um estado neutro (*baseline*), um estado de estresse (*stress*) e um estado de animação (*amusement*), em cada participante. Para isso, após os participantes estarem devidamente equipados com os sensores, cada condição citada acima foi alcançada com os seguintes procedimentos:

- ***Baseline*:** Os indivíduos eram convidados e se sentar ou ficar de pé durante 30 minutos e revistas foram disponibilizadas.

- ***Amusement*:** Os indivíduos assistiram vídeos engraçados durante 392 segundos $\approx$ 6 minutos e 30 segundos.

- ***Stress*:** Os indivíduos foram expostos ao *Trier Social Stress Test* TSST (KIRSCHBAUM; PIRKE; HELLHAMMER, 1993) por 10 minutos, onde os primeiros os primeiros 5 minutos eles fizeram um discurso em público sobre suas características pessoais, os outros 5 minutos eles tiveram de realizar tarefas aritméticas de cabeça.



(a) ECG (b) EDA

(c) EMG

(d) RESP (e) TEMP

Figura 14: Amostra de Um Segundo de Sinal

O estado de *amusement* e de estresse tinham como objetivo de tirar os participantes do estado de *baseline*, portanto, após cada um dessas sessões, uma sessão de meditação era ministrada aos participantes para os acalmar. O experimento gerou uma serie temporal de mais de 3 milhões de pontos para cada um dos indivíduos no e para cada sinal coletado pelos sensores.

No intuito de validar o protocolo aplicado e obter o real estado de cada participante durante o experimento, os pesquisadores do (SCHMIDT et al., 2018) utilizaram uma combinação de questionários foi fornecida aos indivíduos após a conclusão de cada indução de estado. Os questionários usado foram *Positive and Negative Affect Schedule* (PANAS), algumas perguntas tiradas do *State-Trait Axiety Inventory* (STAI), *Self-Assessment Manikins* (SAM), e *Short Stress State Questionnaire* (SSSQ).

Todo o experimento foi feito com a adição de estados de meditação e e descanso, que tinham o objetivo de normalizar os sinais fisiológicos do pacientes e assim prepará-los para uma nova etapa do experimento. A Figura 15 mostra nas duas versões do experimento, a ordem em que cada estado foi induzido e as faixas vermelhas correspondem ao momento em que os questionários de validação foram aplicados. Os dois tipos de experimento foram aplicados visando evitar efeitos de ordem em que cada estimulo foi induzido.

Após a coleta de dados e obtenção do real estado de cada participante, os pesquisadores do (SCHMIDT et al., 2018) também trabalharam num processo de extração de *features* dos sinais coletados e utilizaram diferentes algoritmos de aprendizado de máquina para o problema de classificação em 3 classes (*baseline* x *stress* x *amusement*) e o problema de classificação binária (*stress* x *non-stress*).

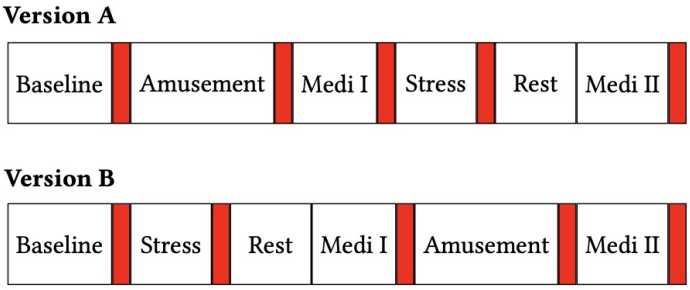


Figura 15: Versões do Experimento de Coleta de Dados. Fonte: (SCHMIDT et al., 2018)

Nos experimentos realizados pelos pesquisadores do WESAD melhores modelos obtidos utilizando somente os dados coletados pelo sensor colocado no tórax e também usou apenas os dados fisiológicos (os dados de ACC foram excluídos). Dessa maneira, nossos experimentos vão se basear nos mesmos dados que os utilizado pelos pesquisadores e também vão trabalhar com o problema de multi classificação utilizando as 3 classes: *baseline* x *stress* x *amusement* e as séries temporais correspondentes a dados fisiológicos coletados pelo sensor do tórax: ECG, EDA, EMG, RESP e TEMP.

5.1.1.2 Modelo Utilizados

O trabalho que publicou os dados do WESAD (SCHMIDT et al., 2018) além de apresentar os dados também fez análises de classificação de estresse nos dados coletados. Neste trabalho, os pesquisadores utilizaram apenas algoritmos clássicos de aprendizado de máquina, não usando técnicas avançadas como redes neurais profundas, aprendizado por reforço e outras. Dessa maneira, para que os dados coletados pudessem ser utilizados foi preciso retirar características específicas de cada dado para que se pudesse alimentar os modelos.

Os modelos que receberam tais dados foram:

- *Decision Tree* - DT
- *Random Forest* - RF
- *AdaBoost DT* - AB
- *Linear Discriminant Analyss* - LDA
- *k-nearst Neighbour* - kNN

Todos estes modelos foram testados utilizando os dados coletados e de várias maneiras diferentes, porém, no problema de classificação de 3 classes, o melhor modelo foi o *AdaBoost DT* e utilizou apenas os dados fisiológicos coletados a partir do sensor do tórax.

Podemos ver seu relatório de classificação no Quadro 3, onde de fato é possível notar uma boa acurácia do modelo para o problema geral.

Quadro 3: Relatório de Classificação para o *AdaBoost DT*

	Precison	Recall	F1-Score	Support
Baseline	0.8002	0.9169	0.8546	70429
Stress	0.9025	0.8779	0.8900	39864
Amusement	0.5926	0.3486	0.4390	22300
Accuracy			0.8096	132593
Macro AVG	0.7651	0.7145	0.7279	132593
Weighted AVG	0.7961	0.8096	0.7953	132593

E se observarmos a matriz de confusão deste modelo, mostrada na Figura 16, também é possível perceber como o resultado de classificação para cada uma das classes também é satisfatório.

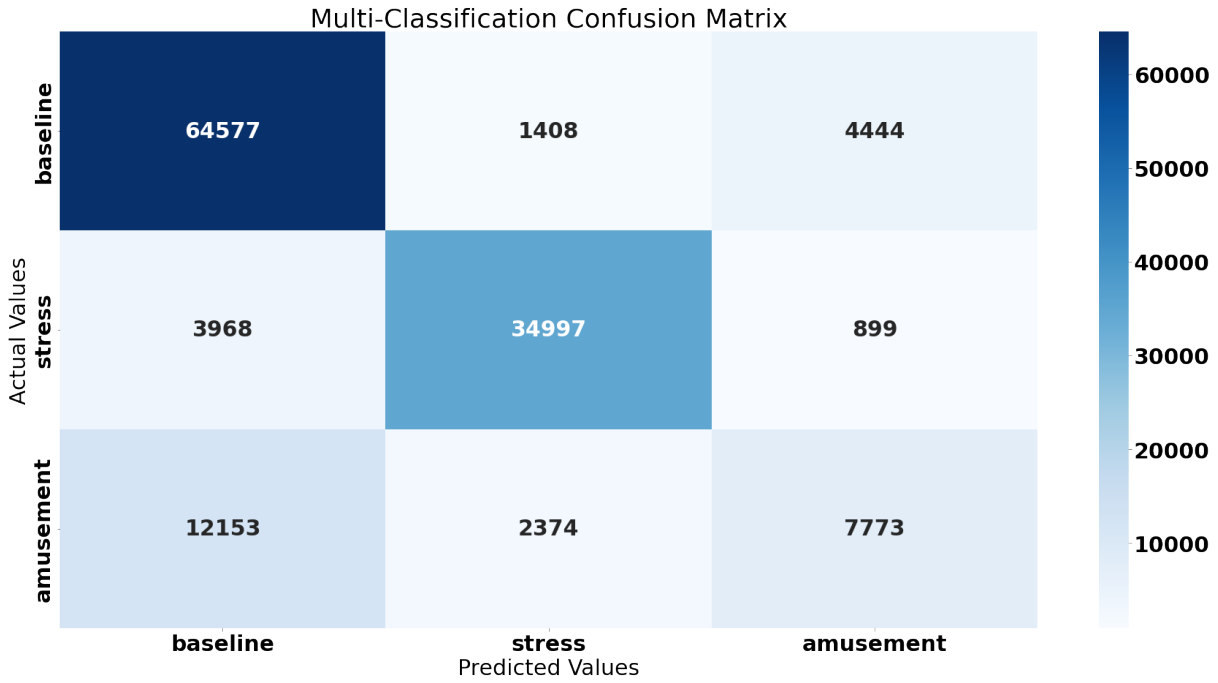


Figura 16: Matriz de Confusão para o *AdaBoost DT*

Sendo assim, é interessante notar como um modelo clássico de aprendizado de máquinas pode chegar a bons resultados, mesmo que exista a limitação de se ter de extrair as principais características do sinal de entrada.

5.1.2 Configuração dos Elementos do MoStress

Como descrito na Seção 4, o MoStress possuí duas fases: a fase de pré-processamento dos dados e a fase de redes neurais. Na Seção 5.1.2.1 serão descritos quais foram os parâmetros de

pré-processamento utilizados nos experimentos com o MoStress. Em seguida serão descritos em detalhes os aspectos das redes neurais utilizadas no experimento do MoStress

5.1.2.1 Parâmetros de Pré-Processamento

Antes dos dados do WESAD serem usados como dados de entrada nos modelos de aprendizado de máquina, tais dados devem ser pré-processados, como mencionado na Seção 4.2. No MoStress, esta etapa consiste em 4 passos: Limpeza de Ruído, Normalização dos Dados, Classificação de Janelas e Cálculo de Pesos, porém, de todos esses passos, apenas a normalização e o calculo de pesos não necessitam da escolha de parâmetros, tendo em vista que no primeiro basta utilizar as séries temporais de entrada e no segundo basta utilizar as janelas para se calcular os pesos.

Para o caso da Limpeza de Ruído utilizando Transformada de Fourier, se olharmos para os dados fisiológicos na Figura 14, nós podemos perceber que RESP é o sinal que possui o maior período entre todos os sinais, enquanto o ECG é o sinal de maior frequência. Assim, aplicamos a Transformada Discreta de Fourier (FFT) no sinal ECG, tendo em vista que esse é o de maior frequência e pode ser o mais afetado por uma re-amostragem.

Apos a FFT, foi possível observar o espectro de frequência do ECG na Figura 17, que consiste em representação no domínio da frequência de todas as ondas que forma o sinal.

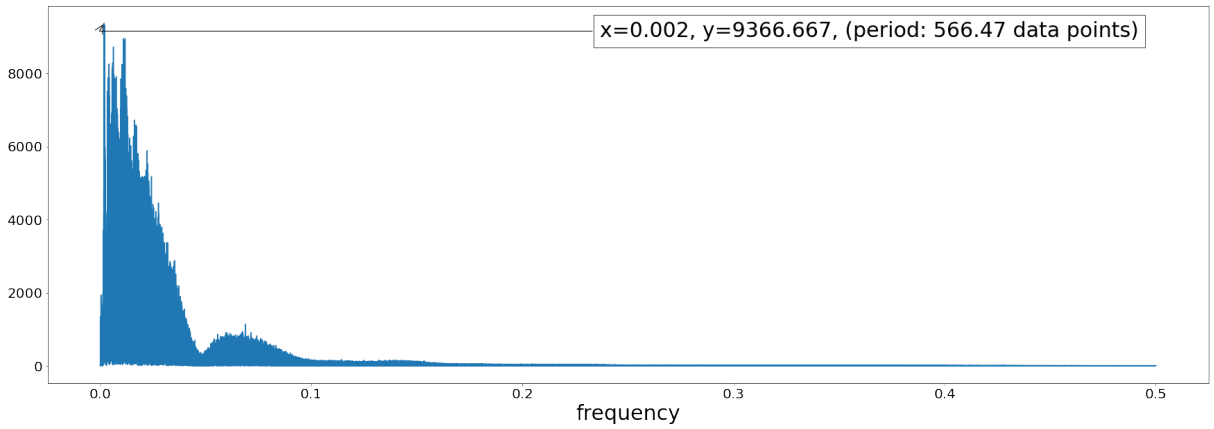


Figura 17: Espectro de Frequência do ECG

Para escolher uma frequência de re-amostragem que limpe o sinal, é importante que esta frequência seja no mínimo duas vezes maior que a frequência natural do sinal, como diz o Teorema de Nyquist (FARROW et al., 2011). Caso a frequência de amostragem esteja abaixo desse limiar de duas vezes a frequência natural, iremos incorrer no problema de "aliasing", que consiste numa falha na possibilidade de reconstrução do sinal original devido a frequência escolhida ser muito baixa, e assim não há informação no sinal re-amostrado para voltar ao sinal original.

A análise do espectro do ECG nos mostra que este sinal tem a componente de maior frequên-

cia como sendo $\frac{1}{566.47} \approx 0.002Hz$, por isso, vamos considerar essa a frequência natural do sinal, que corresponde a aproximadamente 566 pontos em um período. Assim, de maneira a limpar o sinal e evitar o problema de aliasing, vamos usar apenas um ponto a cada 100 pontos. Este valor corresponde a uma frequência de re-amostragem de $0.1428Hz$, que é maior do que o dobro da frequência natural do sinal e por isso, este valor está de acordo com o teorema de Nyquist e assim não vamos gerar o problema de aliasing.

Após ter o sinal limpo, ainda existem três parâmetros a serem definidos:

- Tamanho da janela;
- Passo de movimentação da janela;
- Limiar de contagem de classificação de janelas χ.

Tais parâmetros foram escolhidos de maneira empírica onde foram testados janelas de 0.5, 1, 1.5 e 2 minutos, sempre com um passo de 1 segundo e também foram testados limiares χ de 0.6, 0.7 e 0.8. Os testes destes parâmetros foram feitos utilizando as RNN *baselines*, que serão descritas na Seção 5.1.2.2. Após os testes, a Quadro 4 resume a escolha dos melhores parâmetros de pré-processamento.

Quadro 4: Conjunto de Parâmetros Pré-Processamento Testados

		Valor Testado		
Parâmetro	Tamanho de Janela	0.5 min	1 min	**1.5 min**
	Passo de Janela	1 sec	1 sec	**1 sec**
	Limiar χ	0.5	0.6	**0.7**

Com estes parâmetros definidos foi possível normalizar os dados dentro de cada janela e também calcular o peso referente a cada classe. Como mencionado na Seção 5.1.1.1, os dados de *baseline* foram coletados durante 30 minutos, os de *stress* durante 10 minutos e os de *amusement* durante aproximadamente 6.5 minutos, isso gerou um grande desbalanceamento de dados, onde temos muitos registros de *baseline*, alguns de *stress* e poucos de *amusement*.

Tal desbalanceamento se refletiu no cálculo dos pesos. Ao utilizar a Equação 4.3, os pesos das classes de *baseline*, *stress* e *amusement* foram de 0.62, 1.11, e 1.98, respectivamente, ou seja, inversamente proporcional ao tempo de coleta de cada dado, portanto, fazendo com que a rede neural de mais importância ao aprendizado das classes com o pesos menores.

É importante mencionar que os pesos calculados foram utilizados apenas nos modelos de *baseline* e no experimento da RNN (Seções 5.1.2.2 e 5.1.2.3), pois no experimento das ESN (Seção 5.1.2.4), tais pesos não foram utilizados pois o *framework* utilizado na implementação ainda não possuía uma opção de adicionar pesos na aprendizagem da rede e no experimento com o Nbeats, Seção 5.1.2.5 os pesos foram recalculados diversas vezes. Tais detalhes de cada experimento serão descritos nas seções a seguir.

5.1.2.2 Modelos de *Baseline*

Os modelos *baseline* utilizados consistem em uma RNN simples, com apenas uma camada de entrada e uma camada de saída, ou seja, sem camadas intermediárias.

Nas camada de entrada existem 128 neurônios LSTM ou GRU, com o número de neurônios escolhidos de modo empírico e utilizando função de ativação *Leaky ReLU* com coeficiente alfa de 0.5. A camada de saída consiste em 3 neurônios Dense, tendo em vista que temos 3 classes diferentes, com função de ativação *softmax*, assim como mostra a Figura 18.

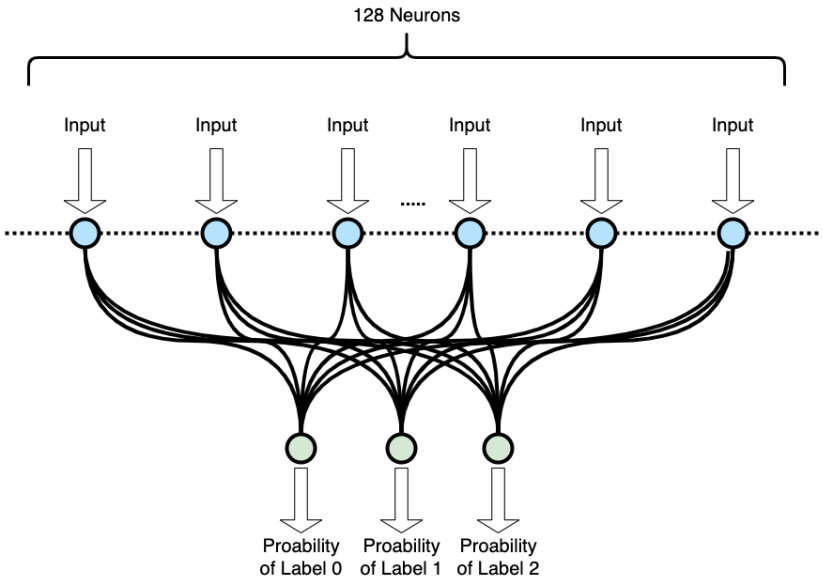


Figura 18: Arquitetura dos Modelos de *Baseline*

As amostras das classes que foram utilizadas para treino e validação do modelo não passaram por nenhum processo de codificação ou decodificação, ou seja, cada classe era representada por um número natural começando do 0. Desta maneira, os modelos *baseline* foram compilados com a função custo *sparse categorical crossentroy* e uma métrica de avaliação de *sparse categorical accuracy*.

Além disso, o modelo foi treinado com 3 otimizadores diferentes: Adam, Nadam e RMSProp, por conta disso, temos ao total 6 modelos de *baseline* que consistem na combinação de modelos com 2 tipos de neurônio de RNN e os 3 otimizadores.

Resultados Numéricos

Ao avaliarmos os resultados numéricos dos modelos de *baselinde*, podemos observar que o treino de tais modelos com os dados pré-processados do WESAD gerou os resultados do Quadro 5.

Quadro 5: Relatório de Classificação para os Modelo de *Baseline*

		Model Accuracy	Stress Precision	Stress Recall	Stress F1-Score
LSTM	Adam	0.81	0.77	0.88	0.82
	Nadam	**0.83**	**0.96**	**0.89**	**0.92**
	RMSProp	0.76	0.88	0.87	0.88
GRU	Adam	0.72	0.92	0.63	0.75
	Nadam	0.59	0.95	0.66	0.78
	RMSProp	**0.73**	**0.99**	**0.67**	**0.80**

De acordo com o Quadro 5, os melhores resultados obtidos para a rede com neurônio LSTM foram obtidos utilizando o otimizador Nadam com os seguintes resultados: 83% de acurácia, 96%, 89% e 92% de precisão, *recall*, *f1-score*, respectivamente, para a classe que representa o estresse. Já para a rede com o neurônio GRU o melhor resultado obtido foi alcançado utilizando o otimizador RMSProp, com os seguintes resultados: 73% de acurácia, 99%, 67% e 80% de precisão, *recall*, *f1-score*, respectivamente, para a classe que representa o estresse.

O otimizador RMSProp não utiliza *momentum* para acelerar o algoritmo de *backpropagation* que acontece no processo de aprendizagem, assim como explicado na Seção 2.2.4.1. Também é importante considerar que este otimizador não possui mecanismos que ajudam a corrigir a direção do gradiente do algoritmo de *backpropagation*, como possuem os otimizadores Adam e Nadam, como explicado nas Seções 2.2.4.2 e 2.2.4.2. Dessa forma, podemos observar que o mecanismo de *gates* das GRU não utilizou dos mecanismos de correção de direção, e por isso, seu melhor resultado foi obtido apenas com o RMSProp.

Ao analisarmos os resultados da LSTM, podemos ver que estes foram melhores quase todos os parâmetros, sendo pior apenas na precisão do estresse e por isso, podemos caracterizar esse modelo como melhor que o modelo de GRU. Observando também que o melhor otimizador para essa rede foi o Nadam, podemos concluir que a implementação de memória das LSTM conseguiu utilizar os mecanismos de ajuste de direção do gradiente do algoritmo de *backpropagation*, assim resultando no melhor resultado dos modelos de *baseline*.

Assim, baseados nesses resultados obtidos pelos modelos de *baseline*, os próximos experimentos serão executados no intuito de superar esses resultados.

5.1.2.3 Rede Neural Recorrente

O primeiro experimento realizado tem por objetivo esclarecer a seguinte pergunta: quão bons podem ser os resultados de classificação de estresse dos dados pré-processados do WESAD utilizando apenas RNNs?

Para tentar alcançar a resposta de tal pergunta, foram alterados diversos parâmetros às RNNs de *baseline* visando a melhora dos resultados.

Também é importante mencionar que esse experimento juntamente com parte do conteúdo dessa dissertação foi publicado em (SOUZA et al., 2022).

Neste experimento, primeiramente foi necessário fazer uma otimização dos hiper-parâmetros de maneira a tentar obter o melhor resultado que a rede pode oferecer.

Para as nova arquitetura de RNN foram alterados alguns parâmetros que ajudam na prevenção de *overfitting* para evitar que a rede incorra nesse problema e assim buscar não mudar a quantidade de neurônios presentes nos *baselines*. Dessa maneira, entre a camada de entrada e a camada de saída foram adicionados um Ruído Gaussiano com um desvio padrão de 0.5, valor escolhido empiricamente, seguido de uma *dropout* com uma frequência de 0.3 (SRIVASTAVA et al., 2014).

Além disso, nos neurônios de saída foram adicionados um regularizador L2 - *Ridge Regression* com λ de 10^{-6} e também impusemos que a norma máxima dos neurônios de saída fosse 3.

Com esses ajustes feitos, foram rodados os experimentos novamente com a combinação dos dois neurônios de RNN e os três otimizadores utilizados no experimento dos modelos de *baseline*.

Ao avaliarmos os resultados numéricos dos modelos modelos de RNN, podemos observar que o treino de tais modelos com os dados pré-processados do WESAD gerou os resultados do Quadro 6.

Quadro 6: Relatório de Classificação para os Modelo das RNNs

		Model Accuracy	Stress Precision	Stress Recall	Stress F1-Score
LSTM	Adam	0.5	0.43	0.74	0.54
	Nadam	0.81	0.91	0.89	0.90
	RMSProp	**0.86**	**0.96**	**0.93**	**0.94**
GRU	Adam	0.65	0.61	0.97	0.75
	Nadam	**0.82**	**0.99**	**0.92**	**0.96**
	RMSProp	0.6	0.79	0.43	0.55

Ao comparamos tais resultados com os obtidos com os *baselines* podemos ver que a utilização de um neurônio de LSTM juntamente com o otimizador RMSPropo alcançou resultados melhores. Todavia, também é interessante notar como ao escolher tais otimizadores, alguns resultados pioram bastante, como por exemplo, se compararmos as redes LSTM Adam dos *baselines* com as novas RNNs, vemos uma queda brusca de acurácia, precisão e F1-Score, o que mostra como o modelo fez muitas classificações equivocadas do estresse confundindo com outras classes.

É importante notar também que se olharmos a matriz de confusão da Figura 19, que corresponde a matriz da melhor das RNNs, ou seja, a RNNs com neurônios de LSTM e otimizador RMSProp, podemos ver que apesar dos bons resultados, o modelo ainda confundiu bastante a

classe de *baseline* com *amusement*, fato esse que pode ser atribuído ao desbalanceamento dos dados. Porém, ainda precisamos investigar de forma mais aprofundada para entender o motivo de tal imprecisão do modelo.

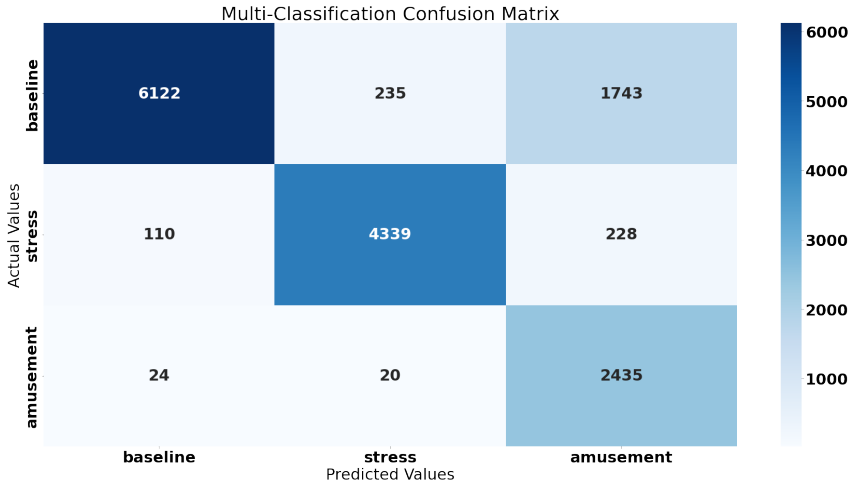


Figura 19: Matriz de Confusão da Melhor RNN

Assim, esse experimento mostrou que é possível sim melhorar os resultados de classificação de estresse utilizando apenas redes neurais recorrentes. Não obstante os bons resultados, ainda se faz necessária uma investigação de como modelos que compõem o estado da arte desempenham na classificação de estresse. Os próximos experimentos visam responder essa pergunta.

5.1.2.4 *Echo State Neural Networks*

O segundo experimento tem por objetivo esclarecer a seguinte pergunta: uma rede ESN, que consiste na disposição aleatória de neurônios RNN, consegue obter resultados melhores que os modelos de *baseline*, utilizando os dados pré-processados do WESAD?

Para responder tal pergunta, foi criada uma rede ESN utilizando a biblioteca *ReservoirPy* (TROUVAIN et al., 2020), por conta disso, alguns parâmetros ou estratégias para a implementação de redes neurais não puderam ser utilizados, como por exemplo: esta biblioteca não aceita treinar as redes com uma métrica de *sparse categorical crossentropy*, dessa maneira, para treinar a rede foi necessário fazer uma codificação nas classes de entrada para números binários e só depois treinar e validar a rede.

Outro aspecto importante de ser mencionado é que no *ReservoirPy* não é possível utilizar pesos para cada classe para ajudar no desbalanceamento de dados, o que faz que nesse caso, os cálculos de peso realizados pelo pré-processamento do MoStress não foram úteis e a não utilização de tais pesos pode afetar os resultados.

Para realizar uma otimização dos hiper-parâmetros referentes às ESN's, é importante notar que tais redes são compostas de três partes: a entrada, a parte interna e a saída.

Para a entrada foi utilizado apenas um neurônio RNN que recebe as janelas e é responsável

por fazer os dados percorrem os neurônios internos. A parte interna da rede possuí 64 neurônios com o maior módulo dos auto valores da matriz W (também conhecido como raio espectral da matriz) com o valor de 0.9 e com o fator de multiplicação dos estados x de cada neurônio com o valor de 0.1, tal coeficiente também é conhecido como *leakingRate*.

Na camada de saída foram escolhidos um neurônio regularizado com o regularizador L2 - *Ridge Regression* com λ de 10^{-6}, é importante notar que como as classes foram codificadas, podemos usar apenas um neurônio de saída para trabalhar com o problema de multi classificação.

Fazendo uma avaliação dos resultados numéricos obtidos, podemos observar que após treinar a rede de ESN e fazer classificações com os dados de validação, foram obtidos os resultados mostrados no Quadro 7

Quadro 7: Relatório de Classificação para os Modelo da ESN

	Precison	Recall	F1-Score	Support
Baseline	0.6220	**0.9129**	**0.7399**	8016
Stress	**0.7642**	0.5043	0.6076	4523
Amusement	0.0047	0.0004	0.0008	2423
Accuracy			0.6416	14962
Macro AVG	0.4636	0.4725	0.4494	14962
Weighted AVG	0.5650	0.6416	0.5802	14962

Aqui podemos ver que o modelo não alcançou resultados expressivos, este modelo obteve apenas 64% de acurácia, o que é muito inferior ao melhor modelo *baseline*. Também podemos ver na Figura 20 como o modelo errou muito na classificação de *stress* e *amusement*, acertando praticamente somente a classe de *baseline*.

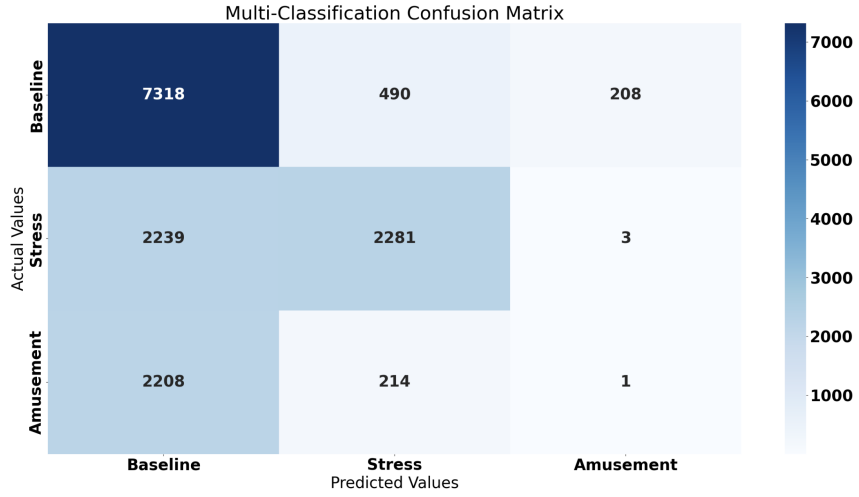


Figura 20: Matriz de Confusão das ESN

O resultado final para este experimento não foi satisfatório. Porém, algumas limitações impostas por parte da implementação podem ser responsáveis por tal resultado, além disso, ainda existem diversas outras arquiteturas de ESN que podem ser elaboradas e podem obter um resultado mais satisfatório, assim, podemos afirmar que um modelo simples de ESN ainda não é capaz de classificar estresse de maneira satisfatória, utilizando os dados pré-processados do WESAD.

5.1.2.5 Nbeats *Feature Extractor*

O último experimento tem por objetivo responder duas perguntas:

1. Como usar o Nbeats, um modelo uni-variável para predição de séries temporais, para fazer a classificação de séries temporais?

2. Uma vez que tal modelo foi utilizado, a classificação com os dados pré-processados do WESAD, será satisfatória?

Existem diversas maneiras de responder a primeira pergunta, tendo em vista a complexidade do modelo, várias configurações seriam possíveis para tentar usar o Nbeats para classificação, porém, como descrito na Seção 4.3.3, foram coletados os resíduos gerados pelo Nbeats para serem os dados de entrada de um rede neural para assim fazer a classificação.

Este experimento se baseia na teoria das Redes Neurais Convolucionais (CNNs), em tais redes, muito utilizadas no tratamento de imagens, são utilizadas matrizes de convolução nos dados de maneira a retirar as suas principais características. Após essa extração, outra rede neural é alimentadas com as características extraídas para assim fazer a classificação dos dados.

No caso desse experimento, o Nbeats faria o papel das matrizes de convolução das CNNs, pois como este modelo tenta sempre diminuir o valor dos resíduos para fazer uma boa predição,

podemos pensar que cada resíduo possui uma característica sobre o sinal temporal e que tais características podem ser bem utilizadas e gerar uma boa classificação.

A Figura 13 demonstra como NBeats é utilizado para a extração de características de cada série temporal e mostra a utilização desses dados dentro de uma MLP. Porém, para realizar tal experimento, existem detalhes de implementação importantes a serem discutidos, que podem ter efeito nos resultados do experimento e que estão ilustrados no diagrama UML da Figura 21.

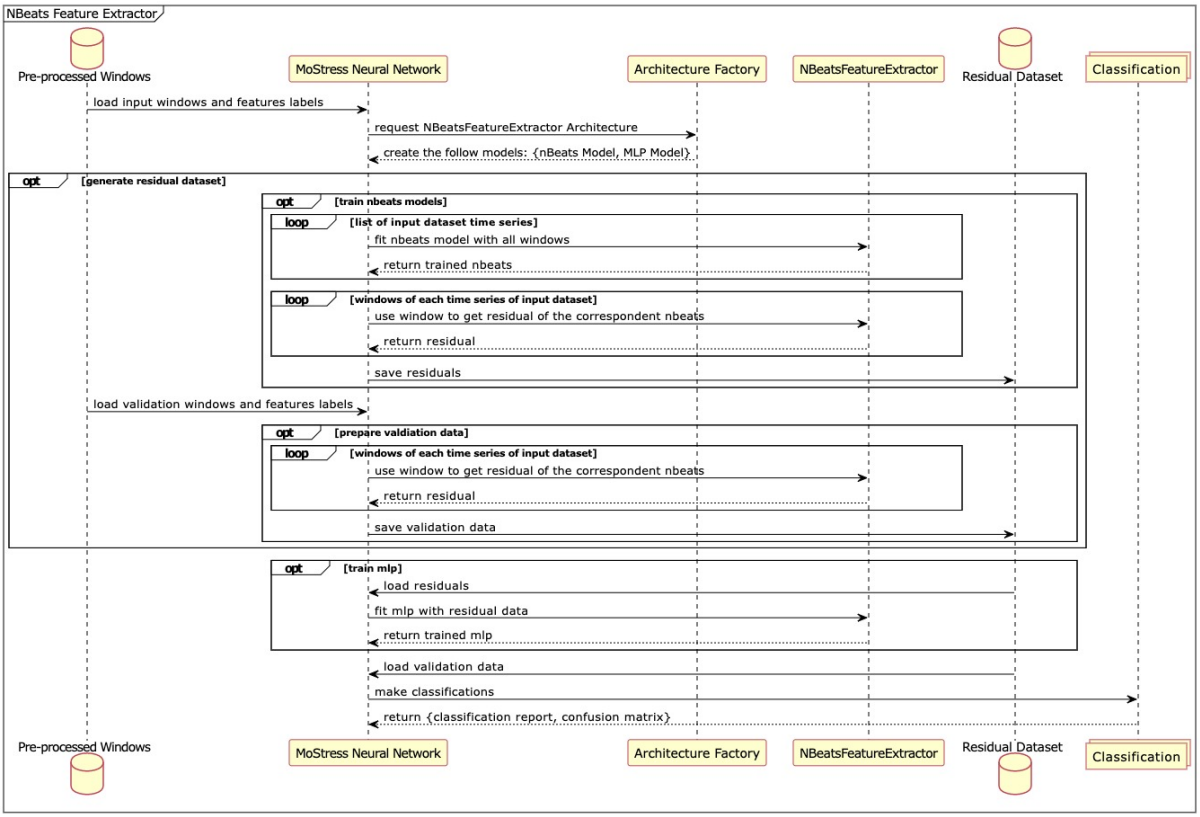


Figura 21: Diagrama de Sequência do Experimento do *NBeats Feature Extractor*

Como descrito no diagrama da Figura 21 a implementação seguiu os seguintes passos:

1. Foram carregados os dados pré-processados do WESAD utilizados para treinamento;
2. Foram geradas 5 arquiteturas do Nbeats, uma para cada série temporal, e também foi gerada a arquitetura da MLP de classificação'
3. Para cada uma das 5 séries temporais foi treinado uma instância de Nbeats;
4. Cada janela de cada série temporal dos dados de treino foi utilizada por sua correspondente instância de Nbeats para fazer uma predição e assim coletar os resíduos da janela correspondente;
5. Foram salvos na memória os resíduos a serem utilizados para treinamento da MLP;
6. Foram carregados os dados de pré-processamento do WESAD utilizados para validação;

7. Cada janela de cada série temporal dos dados de validação foi utilizada por sua correspondente instância de Nbeats para fazer uma predição e assim coletar os resíduos da janela correspondente;

8. Foram salvos na memória os resíduos a serem utilizados para validação da MLP.

Sobre a implementação, o principal ponto a se notar é que o dados do WESAD ocupam aproximadamente 20GB de memória, ao passar uma janela com 420 pontos por uma instância de Nbeats, teremos 30 resíduos para uma janela, assim, passamos de um dado com uma estrutura de 420 pontos para outro com uma estrutura de 30x420 pontos. Fazendo essa operação de coleta de resíduos para todo os dados pré-processados do WESAD, os dados de resíduo acabaram por ocupar aproximadamente 70GB de memória, ou seja, um valor 6 GB maior do que a memória do cluster onde os experimentos foram executados.

Para contornar este problema, a coleta de resídios além de ser feita a nível de série temporal, também foi feita em 3 etapas diferentes, ou seja, era coletada uma etapa, salva na memória de disco, feita uma limpeza da memória RAM e começava a coleta da próxima etapa. Este procedimento possibilitou a coleta de todos os resíduos, entretanto ainda foi necessário lidar com uma segunda limitação.

Para que comparações pudessem ser feitas entre este experimentos e os outros, é importante que o resídio de cada série temporal fosse consolidado em um bloco de resíduo que correspondesse a todas as séries temporais juntas, tendo em vista que nos outros experimentos as séries temporais não foram utilizadas separadamente. Dessa forma, foi feita a união dos resíduos de cada série temporal para alimentar modelo MLP, portanto, era necessário que fosse feita a conversão para Tensor.

Como o *framework* de aprendizado de máquinas utilizado foi o *Tensorflow* configurado para o uso de GPU, ao fazer operações com tensores temos a limitação de memória da GPU, que no nosso caso é de 16GB, somado a isso, a conversão de dados para Tensor do *Tensorflow* utiliza duas vezes o espaço da memória da GPU, um espaço de memória referente ao dado que está sendo convertido e outro espaço de memória referente ao Tensor que vai ser gerado, por isso, assumindo que o Tensor não ocupa mais espaço que o dado que o gerou, o limite de tamanho para o resíduo de das 5 séries temporais era de 8GB e por isso o resíduo das 5 séries temporais foi repartido em 16 pedaços diferentes.

O problema que pode ser gerado de treinar a MLP com 16 pedaços de dado diferentes é que ao final do treino, a rede ficará mais especializada no último pedaço utilizado para treino, por isso, o treinamento de 16 pedaços foi realizado 5 vezes, de maneira a chegar na convergência de aprendizado da MLP.

No quesito de otimização de hiper-Parâmetros, para este experimento existem dois modelos a serem otimizados, o Nbeats e a MLP. Para o NBeats, foi utilizada a arquitetura com um bloco por pilhas e 30 pilhas, como descrito na Seção 4.3.3, porém, para rodar o experimento ainda é necessário escolher o valor do *horizon* e do período de *lookback*.

Após o pré-processamento de dados do WESAD, cada linha da matriz que representa as janelas corresponde a um segundo, assim, de maneira empírica, foi escolhido o *horizon* como 1, pois assim, o Nbeats está sempre prevendo um segundo no futuro da série temporal.

A escolha do período de *lookback* foi feita baseada no próprio trabalho que introduziu o Nbeats (ORESHKIN et al., 2019). Os criadores do Nbeats afirmam que para um *horizon* H, os melhores períodos de *lookback* são: 2H, 3H, 4H, 5H, 6H e 7H, sendo que o 7H foi o período utilizado que gerou os melhores resultados, portanto, o período de *lookback* para o NBeats do MoStress também foi de 7H.

As instâncias de Nbeats foram treinadas com uma *EarlyStopping* como função de *callback*, configurada com uma paciência de 30 épocas, visando evitar o *overfitting*, e as instâncias utilizaram Adam como otimizador e a MAE como métrica de *loss*, assim como é descrito em (ORESHKIN et al., 2019) e foram configuradas 100 épocas de treinamento.

No caso da MLP foi escolhida uma arquitetura simples que consistia em uma camada de neurônios Dense com 32 neurônios, uma Dropout com frequência de 0.3 e por fim uma camada de saída com 3 neurônios Dense e com função de ativação *softmax* e com regularizador L2 - *Ridge Regression* com λ de 10^{-6} e também impusemos que a norma máxima dos neurônios de saída fosse 3.

A MLP foi compilada utilizando Adam como otimizador, e também com uma *EarlyStopping* como função de *callback*, configurada com uma paciência de 30 épocas, a métrica de *loss* utilizada foi a *sparse categorical crossentroy* e também foi medida a *sparse categorical accuracy*.

Para avaliar os resultados numéricos deste experimentos, é necessário ter em vista que o Nbeats foi utilizado apenas com um extrator de características das séries temporais, não existem resultados de classificação vindo desse modelo, porém, ao analisarmos a curva de aprendizado na Figura 22, existem pontos interessantes a serem comentados.

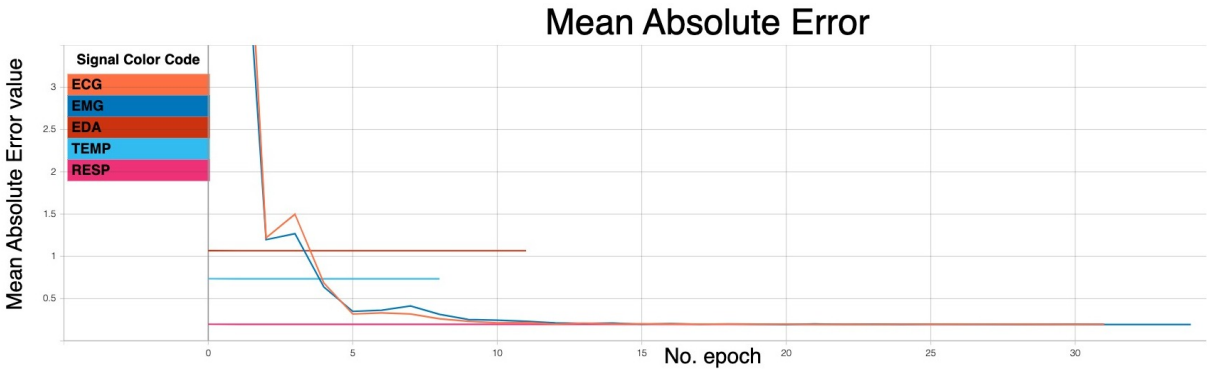


Figura 22: Curva de Aprendizado do Nbeats para cada Série Temporal

Primeiramente é notável o pequeno erro observado na convergência dos sinais de ECG, EMG e RESP. Isso significa que tais sinais podem ser facilmente preditos pela arquitetura do Nbeats, o que traz a hipótese que tais sinais talvez também possam obter resultados satisfatórios na tarefa da classificação das 3 classes.

Ao juntar os resíduos desses 3 sinais e utilizar a MLP para fazer a classificação de estresse, obteve-se o relatório de classificação do Quadro 8.

Quadro 8: Relatório de Classificação para os Modelo MLP com os Sinais ECG, EMG e RESP combinados

	Precison	Recall	F1-Score	Support
Baseline	0.7615	0.3150	0.4456	8016
Stress	0.5039	0.7984	0.6178	4523
Amusement	0.1462	0.2703	0.1898	2423
Accuracy			0.4539	14962
Macro AVG	0.4705	0.4612	0.4178	14962
Weighted AVG	0.5840	0.4539	0.4563	14962

Para estes sinais, podemos observar que o resultado da classificação foi muito insatisfatório, ficando com a acurácia abaixo do ESN e também abaixo dos *baselines*. Ao observarmos a matriz de confusão deste modelo na Figura 23, podemos observar como as classificações foram de fato equivocadas.

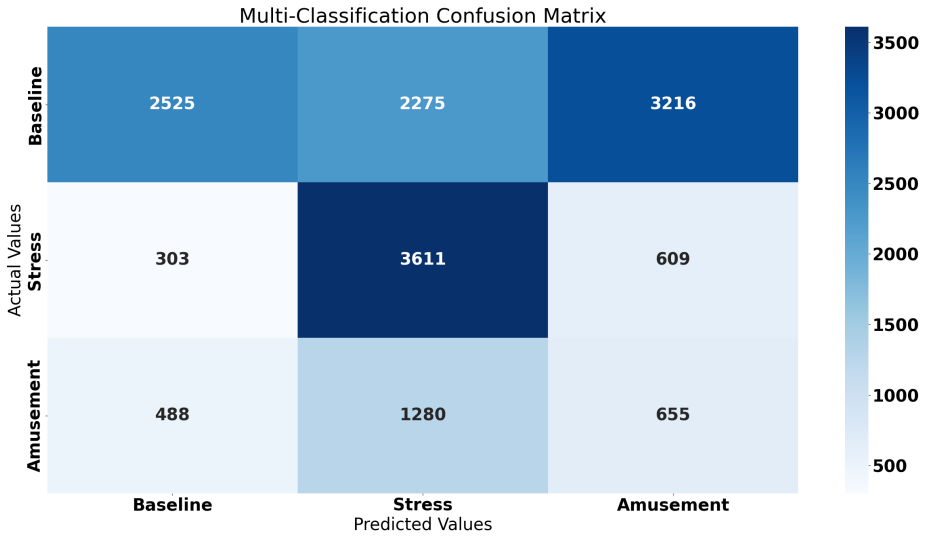


Figura 23: Matriz de Confusão para os Sinais ECG, EMG e RESP combinados

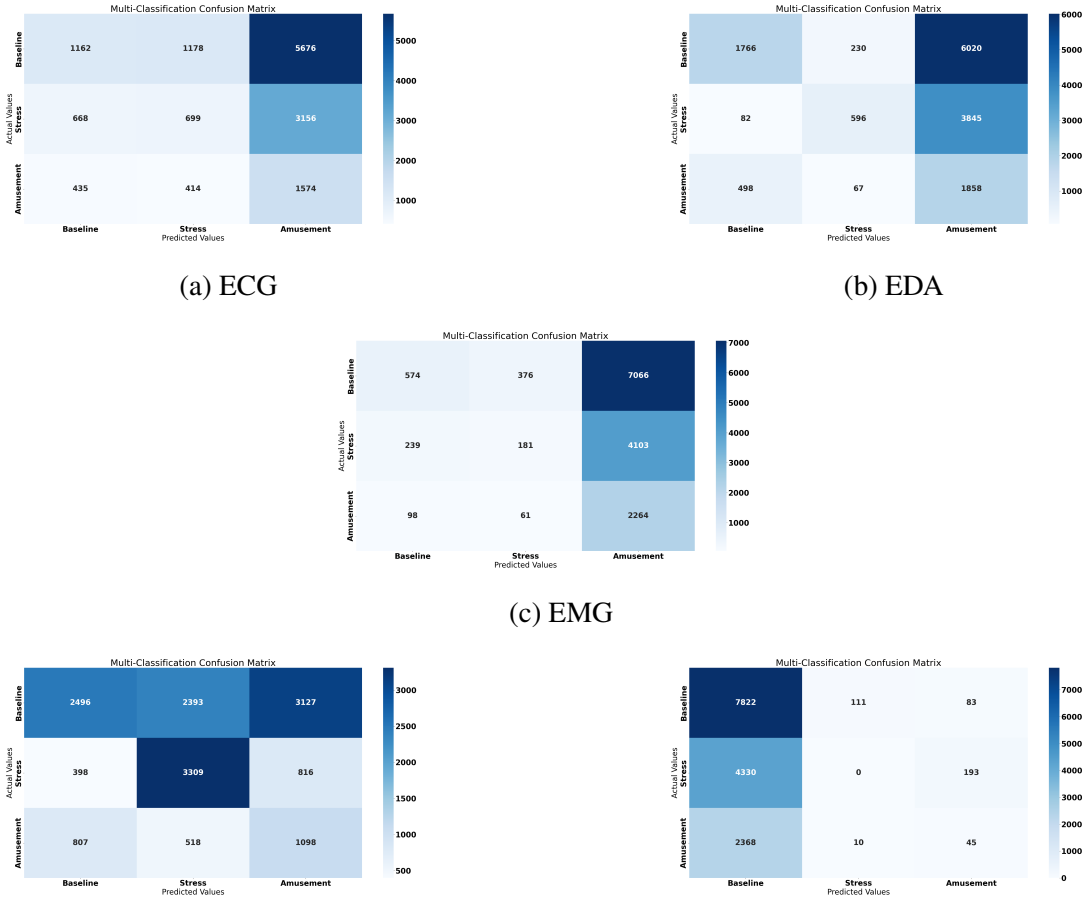
Não obstante o resultado pobre no problema de multi classificação, é interessante notar que das 4523 janelas de estresse, o modelo conseguiu prever corretamente 3611, ou seja, aproximadamente 80% das classificações de estresse foram corretas.

Portanto, para a tarefa de classificação de 3 classes, a junção dos 3 sinais obteve um resultado

insatisfatório, porém, foi avaliada a possibilidade de que cada sinal, individualmente possa atingir a bons resultados de classificação ao tentar classificar *baseline*, *stress* e *amusement*.

O Quadro 9 mostra o relatório de classificação por série temporal e ao observarmos tal tabela podemos perceber que nenhuma série temporal isolada conseguiu um resultado satisfatório, e também é interessante observar nos Quadros 9a e 9b, que os sinais ECG e EMG alcançaram acurácias um pouco melhores que um classificador aleatório poderia conseguir.

Se analisarmos as matrizes de confusão de cada série temporal, mostradas na Figura 25, podemos ratificar a baixa performance de classificação de cada série temporal isolada, entretanto, ao observarmos as matrizes das Figuras 24a, 24b e 24c, referentes aos sinais ECG, EDA e EMG, podemos notar que tais sinais obtiveram boa performance na classificação do *amusement*, tendo em vista as 2423 janelas dessa classe, os sinais de ECG, EDA e EMG, respectivamente, classificaram corretamente 64.96%, 76.68% e 93.33% dessas janelas.



(a) ECG

(b) EDA

(c) EMG

(d) RESP

(e) TEMP

Figura 24: Matriz de Confusão por Série Temporal

Observando o sinal de RESP podemos ver que este sinal classificou corretamente 3309 janelas de *stress*, o que corresponde a 73.15% das classificações corretas e o sinal de TEMP classificou correntemente 7822 janelas de *baseline* das 8016, o que corresponde a 97.75% das classificações corretas.

Ao observar esses resultados, podemos levantar a hipótese que cada sinal fisiológico é melhor para classificar uma classe do que outras, o que nos levar a pensar que todos esses sinais combinados podem gerar uma resultado melhor na tarefa de multi classificação, assim como nos experimentos anteriores. Portanto, foi feita a concatenação dos resíduos de cada série temporal para que fosse feita a classificação de estresse.

Se observarmos o relatório de classificação de todas as séries temporais combinadas, no Quadro 10, podemos também observar que a combinação das séries temporais também obteve um resultado pobre no problema de classificação proposto.

Quadro 10: Relatório de Classificação de Todas as Séries Temporais Combinadas

	Precison	Recall	F1-Score	Support
Baseline	0.5218	0.5334	0.5275	8016
Stress	0.2596	0.3495	0.2980	4523
Amusement	0.1785	0.0499	0.0780	2423

Accuracy			0.3995	14962
Macro AVG	0.3200	0.3110	0.3012	14962
Weighted AVG	0.3869	0.3995	0.3853	14962

Ao observarmos na Figura 25 a matriz de confusão gerada por todas as séries temporais combinadas, vemos que o resultado geral foi insatisfatório e para cada classe individual também.

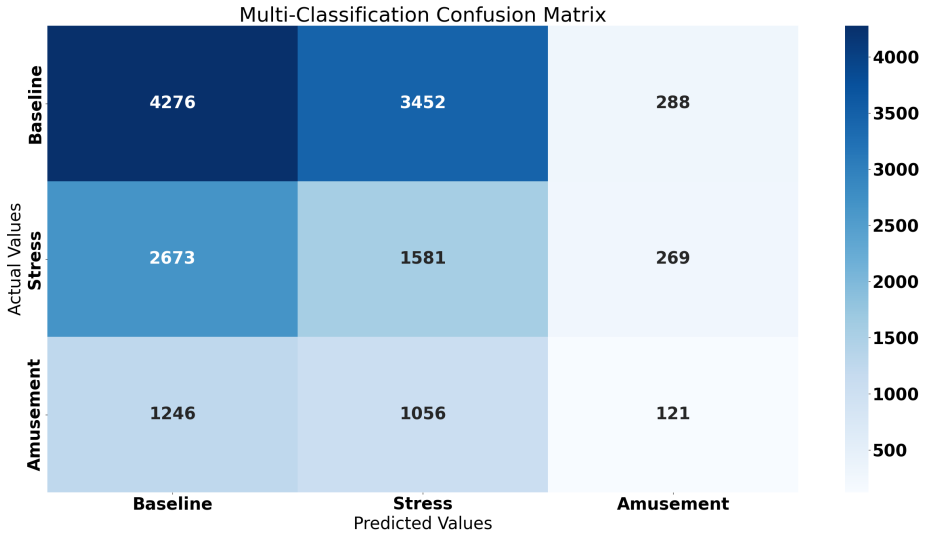


Figura 25: Matriz de Confusão Todas as Séries Temporais Combinadas

Com isso, podemos concluir que a utilização dos resíduos gerados pelos dados pré-processados

do WESAD gera resultados insatisfatórios ao passarem pelo modelo do Nbeats e serem utilizados por uma MLP para fazer classificação de *baseline*, *stress* e *amusement*.

É possível que existam outras maneiras de utilizar a arquitetura do Nbeats, ou até mesmo uma maneira diferente de tratar os resíduos que possa gerar resultados melhores, porém os resultados deste experimento mostram que tal procedimento gera resultados insatisfatórios e não conseguiu ultrapassar os modelos de *baseline*. Todavia, um dos trabalhos futuros, descritos na Seção 6.2, consiste em explorar mais possibilidades com o NBeats de maneira a obter melhroes resultados de classificação.

5.2 Comparação Entre Modelos

Tendo em foco a detecção de estresse, o Quadro 11 resume a comparação de todos os modelos estudados nesse trabalho, focando principalmente nas métricas gerais dos modelos e também nas métricas de estresse.

Quadro 11: Comparação Entre Todos os Modelos

Model Name	Model Accuracy	Stress Precision	Stress Recall	Stress F1-Score
AdaBoost DT (SCHMIDT et al., 2018)	0.80	0.90	0.87	0.89
Baseline	0.83	0.96	0.89	0.92
Recurrent Neural Network	**0.86**	**0.96**	**0.93**	**0.94**
Echo State Network	0.64	0.76	0.50	0.60
NBeats Feature Extractor	0.39	0.25	0.34	0.29

Se observar os resultados de cada modelo é interessante notar como o experimento mais simples foi capaz de gerar o melhor resultado. Evidente que isso não significa que os outros modelos explorados não possam gerar resultados melhores, como descrito em cada experimento da Seção 5, ainda existem diversas escolhas e ajustes que podem ser feitos de maneira a possivelmente melhorar os resultados.

Ao compararmos o resultado do experimento da RNN, descrito na Seção 5.1.2.3, com o melhor resultado obtido pelos pesquisadores do WESAD, podemos notar que a diferença entre estes modelos não é muito grande, o que dificulta encontrar a resposta da pergunta de qual modelo é melhor.

O treinamento de cada RNN descrita no experimento da Seção 5.1.2.3 demorou por volta de 6h, porém, basta adicionar os dados em tal modelo e já poderemos treiná-lo e fazer sua avaliação. Já se consideramos o *AdaBoost DT*, apesar de não ter essa informação no trabalho de (SCHMIDT et al., 2018), possivelmente este modelo demorou menos tempo para ser treinado. Todavia foi necessário fazer uma extração de características dos sinais de entrada para que se pudesse treinar e avaliar o modelo.

Além do mais, todos os experimentos feitos neste trabalho utilizaram os dados pré-processados do WESAD, já em (SCHMIDT et al., 2018), é apenas descrita a extração de características dos sinas e não é mencionado nenhum outro tratamento de dados, dessa maneira, podemos concluir que mais experimentos são necessários para que se possa traçar um bom paralelo entre os dois modelos, porém é seguro afirmar que os modelos com as RNNs são muito promissores.

5.3 Discussão de Resultados

Um dos trabalhos que também utilizou redes neurais nos dados do WESAD foi (LI; LIU, 2020). Neste trabalho os pesquisadores utilizaram redes CNN e conseguiram uma acurácia 97.48% e f1-score de 96.82% para o problema de classificação com três classes utilizando somente os dados fisiológicos do sensor do tórax, e para esse tipo de problema, esse ainda não foi o melhor resultado obtido por tal estudo. Ao utilizar os dados fisiológicos e também os dados de acelerômetro do sensor do tórax, o estudo mostrou um resultado com acurácia 99.55% e f1-score de 99.46%.

Apesar dos ótimos resultados descritos nestes estudo, podemos considerar que existe uma falta de informações para que possamos replicar e atestar esse resultado. No artigo de (LI; LIU, 2020) não existe descrição de pré-processamento dos dados, bem como não existe um detalhamento dos resultados a partir de uma matriz de confusão. Dessa maneira, a falta dessas informações acaba por dificultar a reprodução dos resultados e a validação do modelo.

De toda forma, o estudo de (LI; LIU, 2020) conseguiu utilizar as CNN's de forma a realizar uma extração de características de cada sinal mais eficiente do que a descrita no próprio trabalho do (SCHMIDT et al., 2018). A evidência disso é mostrada ao avaliarmos o melhor modelo para classificação de três classes gerado pelos pesquisadores do (SCHMIDT et al., 2018), o *AdaBoost*, que foi descrito em detalhes na Seção 5.1.1.2 e obteve uma acurácia de 80.96% e f1-score de 89%, ou seja, um pior resultado em relação a CNN de (LI; LIU, 2020).

Ao comparar as RNN's descritas nesse trabalho, nós também obtivemos resultados piores em comparação a (LI; LIU, 2020) e melhores que (SCHMIDT et al., 2018). Podemos atribuir o fato de que as redes neurais obtiveram resultados melhores que os algoritmos usados pelos pesquisadores do (SCHMIDT et al., 2018) pelo fato de que as redes não necessitam de uma extração de características do sinal feita previamente por um especialista. Assim, ao utilizar tais redes é possível mitigar erro humano e focar em utilizar características do sinal que possam ser mais relevantes.

Para tirarmos conclusões do motivo das nossas RNN's terem resultados inferiores as CNN's de (LI; LIU, 2020) ainda é necessário mais estudo sobre como as CNN's foram criadas e utilizadas. Também se faz necessário estudar diferentes formas de se utilizar os dados com as RNN's e também diferentes configurações desse tipo de rede.

Assim, os experimentos descritos nesta seção demonstraram as seguintes contribuições deste trabalho:

1. A utilização de uma Rede Neural Recorrente com os dados do WESAD: como descrito na Seção 3, não existiam trabalho que utilizassem este tipo de rede para os dados do WESAD, mesmo essa rede sendo um dos principais mecanismos para tratar séries temporais. Apesar dos resultados serem inferiores aos obtidos por (LI; LIU, 2020), este trabalho mostrou que as RNN são capazes de obter bons resultados de classificação com os dados do WESAD;

2. A utilização de modelos que compõem o estado da arte na área de redes neurais: também na Seção 3 foi observado que não existiam modelos mais complexos que redes neurais como RNNs ou CNNs na área de classificação de estresse. Nesse sentido este trabalho trouxa a utilização das ESNs e do NBeats para classificação de estresse, e com isso, adicionou modelos que representam o estado da arte na área de classificação de estresse.

3. Uma utilização possível do NBeats para classificação de séries temporais: o NBeats é um modelo criado para previsão de uma série temporal. Esse trabalho utilizou os resíduos do NBeats para fazer classificação de diversas séries temporais, e com isso descreveu uma nova maneira de utilizar este modelo.

Tendo em vista os pontos destacados acima, este trabalho mostrou como se pode usar técnicas de aprendizado profundo para detectar estresse agudo a partir de séries temporais que caracterizam sinais fisiológicos. Dessa maneira, podemos concluir que os experimentos realizados nessa dissertação foram capazes de responder a pergunta de pesquisa.

Quadro 9: Relatório de Classificação por Série Temporal

(a) ECG

	Precison	Recall	F1-Score	Support
Baseline	0.5501	0.4420	0.4901	8016
Stress	0.3138	0.3668	0.3382	4523
Amusement	0.1670	0.2229	0.1909	2423
Accuracy			0.3838	14962
Macro AVG	0.3436	0.3439	0.3398	14962
Weighted AVG	0.4166	0.3838	0.3958	14962

(b) EMG

	Precison	Recall	F1-Score	Support
Baseline	0.5468	0.4406	0.4880	8016
Stress	0.3054	0.3197	0.3124	4523
Amusement	0.1860	0.2893	0.2264	2423
Accuracy			0.3796	14962
Macro AVG	0.3461	0.3499	0.3423	14962
Weighted AVG	0.4154	0.3796	0.3926	14962

(c) EDA

	Precison	Recall	F1-Score	Support
Baseline	0.6981	0.4800	0.5689	8016
Stress	0.7743	0.3657	0.4968	4523
Amusement	0.1520	0.4589	0.2284	2423
Accuracy			0.4421	14962
Macro AVG	0.5415	0.4349	0.4314	14962
Weighted AVG	0.6327	0.4421	0.4920	14962

(d) TEMP

	Precison	Recall	F1-Score	Support
Baseline	0.5800	0.7486	0.6536	8016
Stress	0.2525	0.1548	0.1919	4523
Amusement	0.2013	0.1531	0.1739	2423
Accuracy			0.4727	14962
Macro AVG	0.3446	0.3522	0.3398	14962
Weighted AVG	0.4197	0.4727	0.4364	14962

(e) RESP

	Precison	Recall	F1-Score	Support
Baseline	0.6166	0.2378	0.3432	8016
Stress	0.5205	0.7787	0.6239	4523
Amusement	0.2040	0.4296	0.2766	2423
Accuracy			0.4324	14962
Macro AVG	0.4470	0.4820	0.4146	14962
Weighted AVG	0.5207	0.4324	0.4173	14962

6 CONCLUSÃO

Este trabalho buscou responder a seguinte pergunta de pesquisa: "Como detectar estresse agudo a partir de sinais fisiólogos associados utilizando técnicas de aprendizado profundo?"e para isso, tentou cumprir os seguintes objetivos:

1. Entender como sinais fisiológicos (que podem ser considerados como séries temporais) podem ser tratados e utilizados em algoritmos de aprendizado de máquina.

2. Criar uma arquitetura que possa fazer a classificação de estresse de maneira satisfatória.

3. Utilizar modelos de aprendizado de máquina no estado da arte na classificação de estresse.

Na Seção 3 podemos entender melhor quais foram os trabalhos realizados na área de classificação de estresse e pode ser visto diversos trabalhos diferentes que utilizaram sinais fisiológicos diferentes para classificação de estresse. Assim foi possível entender como sinais fisiológicos característicos do estresse podem ser utilizado em aprendizado de máquina, assim completando o primeiro dos objetivos.

A Seção 4 descreve a arquitetura criada por este trabalho e que conseguiu classificar estresse de maneira satisfatória. A Seção 5 descreve os experimentos feitos de discute os resultados obtidos, assim completando o segundo objetivo.

E os experimentos descritos na Seção 5, em particular as Seções 5.1.2.4 e 5.1.2.5, mostram como alguns modelos no estado da arte podem desempenhar na classificação de estresse, assim completando o terceiro objetivo.

Portanto, podemos afirmar que este trabalho descreveu como detectar estresse agudo a partir de sinais fisiológicos associados e como o estado da arte está posto em tal tema.

6.1 Limitações

Uma das limitações desse trabalho é o fato de o MoStress ter sido testado apenas em um conjunto de dados, dessa forma não podemos ter clareza como o pré-processamento afeta o aprendizado de alguns modelos e também como outras séries temporais afetam o aprendizado e o desempenho da arquitetura proposta.

Além disso, outra limitação a ser explorada é a maneira que os resultados de avaliação do modelo são obtidos e como os dados são utilizados para aprendizagem. A abordagem aqui utilizada juntou os dados de todos os pacientes para treinamento, separando apenas um para a validação do modelo, porém esta divisão pode ser feita de diversas maneiras diferentes e isso pode afetar a maneira que os experimentos são feitos e os resultados que podem ser obtidos.

Por fim, uma outra limitação que pode ser observada é a falta de um estudo de generalização do MoStress, pois ainda é necessário entender se será possível que uma instância pré-treinada do MoStress consegue virar especialista em classificar estresse agudo em um indivíduo específico.

6.2 Trabalhos Futuros

Os principais trabalhos futuros consistem em:

- Testar o MoStress com dados diferentes: para que se possa encontrar melhorias no MoStress e buscar fazer com que o modelo possa ser generalizado, é preciso que este modelo seja testado com uma diversidade maior de dados. Por isso, é necessário utilizar o MoStress com outros tipos de dados públicos sobre estresse e buscar parcerias para que se possa obter mais dados para validar o modelo;
- Estudar outras maneiras de usar o Nbeats para a classificação de estresse: uma das principais características do NBeats é a diminuição do erro de predição da série temporal. Sendo assim, de maneira a utilizar essa principal característica, é possível ajustar as camadas de saída de cada bloco do NBeats de maneira que cada bloco faça uma classificação. Assim, a arquitetura do NBeats vai diminuir o erro de classificação, e assim, poderá chegar em um bom resultado para a classificação da série temporal;
- Estudar como utilizar técnicas de desbalanceamento de dados nas ESN's: como descrito neste trabalho, as ESN's não obtiveram bons resultados de classificação, porém também não puderam utilizar os pesos de cada classe para ajudar na classificação. Dessa maneira, utilizar outras técnicas de desbalanceamento de dados se faz importante para que uma gama maior de modelos possa ser incorporada ao MoStress;
- Entender como modelos no estado da arte de classificação de séries temporais podem ser utilizados para classificar estresse: neste trabalho foi mostrado quais são os modelos que compõem o estado da arte na área de classificação de estresse a partir de sinais fisiológicos. Porém, como sinais fisiológicos são representados por séries temporais, é necessário fazer uma busca na literatura para entender quais são os modelos que compõem o estado da arte na área de classificação de séries temporais e depois aplicar tais modelos na classificação de estresse.

Por isso ainda existe muito trabalho a ser feito e mutias contribuições a serem concluídas.

6.3 Artigos Publicados

Como resultado desta pesquisa, foi possível elaborar um artigos científicos que foi publicados em um veículo qualificado da área. Os referido artigo está listado abaixo e também se encontra anexado no Apêndice A:

- SOUZA, A. de et al. Mostress: a sequence model for stress classification. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2022., 2022, Padova, Italy. Anais. . . IEEE, 2022. p. 1–8.

REFERÊNCIAS

ALHITARY, A. E. et al. Objective detection of chronic stress using physiological parameters. **Medical & biological eng. & comp.**, [S.l.], v. 56, n. 12, p. 2273–2286, 2018.

ATES, H. C. et al. Wearable devices for the detection of covid-19. **Nature Electronics**, [S.l.], v. 4, n. 1, p. 13–14, 2021.

BATTALIO, S. L. et al. Sense2stop: a micro-randomized trial using wearable sensors to optimize a just-in-time-adaptive stress management intervention for smoking relapse prevention. **Contemporary Clinical Trials**, [S.l.], v. 109, p. 106534, 2021.

BROWN, T. B. et al. Language models are few-shot learners. **CoRR**, [S.l.], v. abs/2005.14165, 2020.

DELMASTRO, F.; DI MARTINO, F.; DOLCIOTTI, C. Cognitive training and stress detection in mci frail older people through wearable sensors and machine learning. **IEEE Access**, [S.l.], v. 8, p. 65573–65590, 2020.

DEVORE, G. R. Computing the z score and centiles for cross-sectional analysis: a practical approach. **Journal of Ultrasound in Medicine**, [S.l.], v. 36, n. 3, p. 459–473, 2017.

DI MARTINO, F.; DELMASTRO, F. High-resolution physiological stress prediction models based on ensemble learning and recurrent neural networks. In: IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS (ISCC), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p. 1–6.

ESLAMI, T. et al. Asd-diagnet: a hybrid learning approach for detection of autism spectrum disorder using fmri data. **Frontiers in neuroinformatics**, [S.l.], v. 13, p. 70, 2019.

FARROW, C. L. et al. Nyquist-shannon sampling theorem applied to refinements of the atomic pair distribution function. **Physical Review B**, [S.l.], v. 84, n. 13, p. 134105, 2011.

FERRARI, A. J. et al. The burden attributable to mental and substance use disorders as risk factors for suicide: findings from the global burden of disease study 2010. **PloS one**, [S.l.], v. 9, n. 4, p. e91936, 2014.

GJORESKI, M. et al. Continuous stress detection using a wrist device: in laboratory and real life. In: ACM INTER. JOINT CONF. ON PERVASIVE AND UBIQUITOUS COMPUTING: ADJUNCT, 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p. 1185–1193.

GJORESKI, M. et al. Continuous stress detection using a wrist device: in laboratory and real life. In: ACM INTERNATIONAL JOINT CONFERENCE ON PERVASIVE AND UBIQUITOUS COMPUTING: ADJUNCT, 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p. 1185–1193.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT Press, 2016. `http://www.deeplearningbook.org`.

GREDEN, J. F. The burden of recurrent depression: causes, consequences, and future prospects. **Journal of Clinical Psychiatry**, [S.l.], v. 62, p. 5–9, 2001.

GUDIVADA, V.; APON, A.; DING, J. Data quality considerations for big data and machine learning: going beyond data cleaning and transformations. **International Journal on Advances in Software**, [S.l.], v. 10, n. 1, p. 1–20, 2017.

HEALEY, J. A.; PICARD, R. W. Detecting stress during real-world driving tasks using physiological sensors. **IEEE Transactions on intelligent transportation systems**, [S.l.], v. 6, n. 2, p. 156–166, 2005.

HOVSEPIAN, K. et al. cstress: towards a gold standard for continuous stress assessment in the mobile environment. In: ACM INTER. JOINT CONF. ON PERVASIVE AND UBIQUITOUS COMPUTING, 2015., 2015. **Proceedings...** [S.l.: s.n.], 2015. p. 493–504.

HUYSMANS, D. et al. Unsupervised learning for mental stress detection-exploration of self-organizing maps. **Proc. of Biosignals 2018**, [S.l.], v. 4, p. 26–35, 2018.

JAEGER, H. Tutorial on training recurrent neural networks, covering bppt, rtrl, ekf and the"echo state network"approach. , [S.l.], 2002.

KESHAN, N.; PARIMI, P.; BICHINDARITZ, I. Machine learning for stress detection from ecg signals in automobile drivers. In: IEEE INTERNATIONAL CONFERENCE ON BIG DATA (BIG DATA), 2015., 2015. **Anais...** [S.l.: s.n.], 2015. p. 2661–2669.

KIRSCHBAUM, C.; PIRKE, K.-M.; HELLHAMMER, D. H. The 'trier social stress test'–a tool for investigating psychobiological stress responses in a laboratory setting. **Neuropsychobiology**, [S.l.], v. 28, n. 1-2, p. 76–81, 1993.

LI, R.; LIU, Z. Stress detection using deep neural networks. **BMC Medical Informatics and Decision Making**, [S.l.], v. 20, n. 11, p. 1–10, 2020.

LOTFAN, S. et al. Support vector machine classification of brain states exposed to social stress test using eeg-based brain network measures. **Biocybernetics and Biomedical Engineering**, [S.l.], v. 39, n. 1, p. 199–213, 2019.

MCDUFF, D.; GONTAREK, S.; PICARD, R. Remote measurement of cognitive stress via heart rate variability. In: IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, 2014., 2014. **Anais...** [S.l.: s.n.], 2014. p. 2957–2960.

MELANDRI, L. Introduction to reservoir computing methods. , [S.l.], 2014.

MELCHIADES, M. B. et al. A survey on the use of machine learning for detecting stress patterns in human beings. , [S.l.], forthcoming.

NAPOLETANO, P.; ROSSI, S. Combining heart and breathing rate for car driver stress recognition. In: IEEE 8TH INTERNATIONAL CONFERENCE ON CONSUMER ELECTRONICS-BERLIN (ICCE-BERLIN), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p. 1–5.

NATH, R. K.; THAPLIYAL, H.; CABAN-HOLT, A. Machine learning based stress monitoring in older adults using wearable sensors and cortisol as stress biomarker. **Journal of Signal Processing Systems**, [S.l.], p. 1–13, 2021.

ORABI, A. H. et al. Deep learning for depression detection of twitter users. In: WORKSHOP ON COMPUTATIONAL LINGUISTICS AND CLINICAL PSYCHOLOGY: FROM KEYBOARD TO CLINIC, 5., 2018. **Proceedings...** [S.l.: s.n.], 2018. p. 88–97.

ORESHKIN, B. N. et al. N-beats: neural basis expansion analysis for interpretable time series forecasting. **arXiv preprint arXiv:1905.10437**, [S.l.], 2019.

PEDROTTI, M. et al. Automatic stress classification with pupil diameter analysis. **International Journal of Human-Computer Interaction**, [S.l.], v. 30, n. 3, p. 220–236, 2014.

SAEED, S. M. U. et al. Eeg based classification of long-term stress using psychological labeling. **Sensors**, [S.l.], v. 20, n. 7, p. 1886, 2020.

SALONI DATTANI, H. R.; ROSER, M. Mental health. **Our World in Data**, [S.l.], 2021. https://ourworldindata.org/mental-health.

SANO, A.; PICARD, R. W. Stress recognition using wearable sensors and mobile phones. In: HUMAINE ASSOCIATION CONFERENCE ON AFFECTIVE COMPUTING AND INTELLIGENT INTERACTION, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p. 671–676.

SANTAMARIA-GRANADOS, L. et al. Using deep convolutional neural network for emotion detection on a physiological signals dataset (amigos). **IEEE Access**, [S.l.], v. 7, p. 57–67, 2018.

SANTOS SIERRA, A. de et al. A stress-detection system based on physiological signals and fuzzy logic. **IEEE Transactions on Industrial Electronics**, [S.l.], v. 58, n. 10, p. 4857–4865, 2011.

SCHMIDT, P. et al. Introducing wesad, a multimodal dataset for wearable stress and affect detection. In: ACM INTER. CONF. ON MULTIMODAL INTERACTION, 20., 2018. **Proceedings...** [S.l.: s.n.], 2018. p. 400–408.

SCHNEIDERMAN, N.; IRONSON, G.; SIEGEL, S. D. Stress and health: psychological, behavioral, and biological determinants. **Annu. Rev. Clin. Psychol.**, [S.l.], v. 1, p. 607–628, 2005.

SELYE, H. The evolution of the stress concept: the originator of the concept traces its development from the discovery in 1936 of the alarm reaction to modern therapeutic applications of syntoxic and catatoxic hormones. **American scientist**, [S.l.], v. 61, n. 6, p. 692–699, 1973.

SEN, R.; YU, H.-F.; DHILLON, I. Think globally, act locally: a deep neural network approach to high-dimensional time series forecasting. **arXiv preprint arXiv:1905.03806**, [S.l.], 2019.

SEO, W. et al. Deep ecg-respiration network (deeper net) for recognizing mental stress. **Sensors**, [S.l.], v. 19, n. 13, p. 3021, 2019.

SIIRTOLA, P. Continuous stress detection using the sensors of commercial smartwatch. In: ADJUNCT PROCEEDINGS OF THE 2019 ACM INTERNATIONAL JOINT CONFERENCE ON PERVASIVE AND UBIQUITOUS COMPUTING AND PROCEEDINGS OF THE 2019 ACM INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, 2019. **Anais...** [S.l.: s.n.], 2019. p. 1198–1201.

SNEDDON, I. N. **Fourier transforms**. [S.l.]: Courier Corporation, 1995.

SOUZA, A. de et al. Mostress: a sequence model for stress classification. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN), 2022., 2022, Padova, Italy. **Anais...** IEEE, 2022. p. 1–8.

SPIELBERGER, C. D. Manual for the state-trait anxietry, inventory. **Consulting Psychologist**, [S.l.], 1970.

SRIVASTAVA, N. et al. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, [S.l.], v. 15, n. 1, p. 1929–1958, 2014.

SUBHANI, A. R. et al. Machine learning framework for the detection of mental stress at multiple levels. **IEEE Access**, [S.l.], v. 5, p. 13545–13556, 2017.

SULTAN, H. H.; SALEM, N. M.; AL-ATABANY, W. Multi-classification of brain tumor images using deep neural network. **IEEE Access**, [S.l.], v. 7, p. 69215–69225, 2019.

TROUVAIN, N. et al. ReservoirPy: an efficient and user-friendly library to design echo state networks. In: **Artificial neural networks and machine learning – ICANN 2020**. [S.l.]: Springer International Publishing, 2020. p. 494–505.

WATSON, D.; CLARK, L. A.; TELLEGEN, A. Development and validation of brief measures of positive and negative affect: the panas scales. **Journal of personality and social psychology**, [S.l.], v. 54, n. 6, p. 1063, 1988.

ZHAI, J.; BARRETO, A. Stress detection in computer users based on digital signal processing of noninvasive physiological variables. In: IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, 2006., 2006. **Anais...** [S.l.: s.n.], 2006. p. 1355–1358.

ANEXO A – ARTIGOS PUBLICADOS

MoStress: a Sequence Model for Stress Classification

Arturo de Souza*†, Mateus B. Melchiades*, Sandro J. Rigo*, Gabriel de O. Ramos*
* Graduate Program in Applied Computing, Universidade do Vale do Rio dos Sinos, São Leopoldo, RS, Brazil
† SAP Labs Latin America, São Leopoldo, RS, Brazil
{arturossouza@edu.,mateusbme@edu.,rigo@,gdoramos@}unisinos.br

***Abstract*—Mental disorders affect a large number of people worldwide. In response to the increasing number of people affected by such illnesses, there has been an increased interest in the use of state-of-the-art technologies to mitigate its effects. This paper presents a Sequence Model for Stress Classification (MoStress), which is a novel pipeline for pre-processing physiological data collected from wearable devices and for identifying stress sequences using a recurrent neural network (RNN). Using the WESAD dataset, the RNN model achieved accuracy of 86% in a three-class classification problem (baseline vs. stress vs. amusement). When only considering the presence of stress or not, we achieved an accuracy of 96.5% as well as precision, recall, and f1-score of 96%, 93%, and 94%, respectively. Those results are close to other papers using the same dataset, however, the neural network used on MoStress, is considerable simpler.**

***Index Terms*—stress detection, wearable devices, RNN, multi-class classifications, mental health, deep learning, sliding window, unbalanced data**

I. Introduction

Psychological disorders such as depression, stress, and anxiety, have become common in modern society. Recent data suggests that around 10.7% of the global population suffers from at least one mental health disorder [1]. If we look at anxiety, for example, there has been an increase of over 94 million people affected by it from 1990 to 2017, which corresponds to an over 50% increase [1].

These conditions have shown to be extremely dangerous for those affected by them. The relationship between mental health problems and suicide, presented by [2], indicates that an individual suffering from anxiety is 3 times more likely to commit suicide than someone who is not. When we consider depression, the risk increases to 20 times. With this in mind, detecting such diseases is a crucial task, which can be assisted by different state-of-the-art technologies.

An increasingly popular technology for detecting psychological illnesses are wearable devices. These devices are becoming more accessible by the day, and its diversity in vital signs measurement capabilities has proven useful for numerous tasks. For instance, during the COVID-19 pandemic, several works in the literature used wearable devices and their data to assist in the battle against SARS-COV-2 [3]. The effective use of these devices helps expand the use of wearable devices to monitor a wider diversity of physiologic signs [3]. Therefore, these devices are bound to become even more relevant in the future.

Another popular technology that can help reduce the increase in people affected by mental disorders are machine learning algorithms. These algorithms are able to provide insight for virtually any type of data. Some examples of machine learning in mental health are using social media posts to detect depression [4], brain images to detect Autism Spectrum Disorder [5], and physiologic signs to classify an individual's emotions [6].

Stress detection is another popular application of machine learning to health data. Important studies were able to apply ML models to physiological measurements from wearable devices such as Blood Volume Pulse, Heart Rate, and Galvanic Skin Response [7]. Other works also used wearable data to predict stress using supervised learning algorithms such as k-NN [8], and Naive Bayes classifier [9].

Besides the variety of studies on stress detection problem with machine learning, we map that most of those works lack of clearly specified pre-processing pipeline and the dearth of RNNs models for detecting stress from physiological signals. On Section III we unravel those two gaps with more details and this paper aims to fill them with MoStress pipeline.

In this paper we present the MoStress pipeline for data pre-processing, which helps machine learning models improve stress detection tasks when using physiologic data collected from wearable devices. This pipeline, described in details in Section V, is based on the sliding window technique and employs a Fourrier transformation for cleaning noise signals, and rolling-z score normalization in each window. The pipeline also labels windows based on a counting threshold of the labels conditions present on each window and weights calculation for each class of data, in order to deal with an unbalanced dataset. On the pipeline we also propose a Recurrent Neural Network model, described in details in Section V-E, which uses the pre-processed data as input.

In the multi-class classification problem proposed by [10], MoStress achieved an accuracy of 86% for the three-class classification problem (baseline vs. stress vs. amusement), as well as an accuracy of 96.5% with precision, recall, and f1-score of 96%, 93%, and 94%, respectively, for the stress condition in the multi-class problem.

Based on the results obtained, the proposed pipeline helps on the training step of a neural network, thus making it possible for simpler deep learning models to achieve insightful results. Our pipeline also enables the reduction of larger

model's without compromising performance, leaving the usage of more complex models to harder problems, such as when dealing with more them one kind of data.

The contributions of this work can be summarized as follows:

- MoStress pipeline for stress classification;
- A full description of the methodology used on data preprocessing;
- A Recurrent Neural Network model for stress detection based on time series data.

To the best of our knowledge, this is the first sequential model for stress classification with a full description of the methods and a RNN architecture for dealing with time series, which is able to achieved insightful results with low complexity models.

II. Background

On this section we provide a description of important concepts which we use to idealize the RNN used on MoStress and also to understand how stress affects the body. It is important to notice that to create MoStress we used concepts which will be described on further sections, and here we will focus only on the deep learning aspect.

A. Stress Manifestation

Stress is a natural response from the human body to any nonspecific demand imposed upon it, leading to the necessity to adapt [11]. This demand can present itself in several and unique ways; for example, when exposed to heat, our bodies sweat as a cooling mechanism, which differs from the increased heart rate in response to greater muscular demand, such as running a marathon. In short, stress can be defined as the body's response to change in demand.

The same logic can be applied to situations of psychological stress such as work or school. For instance, a student performing an exam has an increased demand for attention, which causes the body to deliver more energy to the brain. When the body detects an event that demands mental adaptation, the sympathetic nervous system (SNS) and the hypothalamic-pituitary adrenocortical (HPA) axis increase the available energy by activating various chemical processes such as breaking down fats or converting glycogen to usable energy [12].

As the body adapts by increasing the available energy, there is an increase in cortisol in the blood, urine, and saliva. Such increase can be clinically measured and is sometimes used as ground-truth in some related work presented in Section III. Other side-effects of stress can be found in increased heart rate, blood pressure, stroke volume pressure, and immune response [13].

B. Neural Networks

Modern deep learning techniques provide a powerful framework for several kinds of tasks [14]. Supervised learning plays one of the main roles in the state-of-the-art techniques, such as the recent GPT-3 model [15] which can be applied to natural language processing, time series forecasting [16], and image recognition [17] tasks.

An important type of neural networks used in deep learning models are Recurrent Neural Networks. RNNs are widely used on time series analysis tasks, since its main characteristic is to take into account a series of past data to calculate the output, which will determine the kind of problem to solve, such as classification problems or time series forecast. A prediction from an RNN is described in Equation 1:

$$\mathbf{s}^{(t)} = f(\mathbf{s}^{(t-1)}, \mathbf{x}^{(t)}; \theta), \tag{1}$$

where the current state $\mathbf{s}^{(t)}$ is calculated based on the past state $\mathbf{s}^{(t-1)}$, input $\mathbf{x}^{(t)}$, and neuron parameters θ. Long Short-Term Memory (LSTM) and Gate Recurrent Unit (GRU) are examples of RNNs. These types of network have different gates which can learn new information, update them, forget them, and output a prediction based on new sequence input. Such mechanism helps prevent the vanish gradient problem, where after several learning interactions of the back-propagation procedure on the time series, the value of gradient is too small and does not perform update network parameters [14].

In order to have better results on machine learning problems, it is important to create a proper neural network architecture; to do so, one must carefully choose the layers, optimizers and hyper-parameters. Therefore, the following two subsections discuss about topics of model architecture that are responsible for the results described in this paper.

1) Activation Function for Multi-class Classification: Supervised learning also embraces the challenge of multi-class classification problems. These problems consist on classifying data in one of three or more different classes. In problems like that, the activation function of the last layer usually must calculate the probability of the output of the network to correspond to each class. In this work, we use the softmax activation function, which calculates the probability g the output z belongs to the class i of K classes as the Equation 2:

$$g(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

The softmax activation function is the generalization of the sigmoid function, shown in Equation 3, used for binary classification problems.

$$g(\mathbf{x}) = \frac{e^{\mathbf{x}}}{1 + e^{\mathbf{x}}} \tag{3}$$

With the choice of activation function for the last layers in place, it is important to study the metrics available to evaluate models on multi-classification problems and understand how we can use it to compare MoStress with other models.

2) Performance evaluation: There are numerous metrics one can use to assess the performance of a machine learning algorithm; however, it is worth highlighting the four most common, which shall be referenced in future sections. These metrics are Accuracy, Precision, Recall, and F1 Score. Accuracy is primarily used with categorical and discrete outputs and

is determined by Equation 4, where TP represents the number of true positives, TN true negatives, FP false positives, and lastly FN the number of false negatives. In short, it is the number of correct inferences divided by the total number of predictions.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

The second metric, Precision, as shown in Equation 5, is an indicative of how exact the predictions are using all positive cases; a low precision means a large presence of false positives. This approach is particularly useful in situations in which mistakes in the positive class are more costly than those in the negative, such as disease treatment. Recall, demonstrated in Equation 6, complements Precision by representing a model's completeness, or the presence of false negatives, also often described as the model's sensitivity. This method is particularly useful in situations where retrieving positive results proves to be more important than its negative counterparts.

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

Lastly, F1 score combines both precision and recall into a metric which captures the trade-off between recall's completeness and precision's exactness. This approach is often used beside accuracy in classification problems, and is demonstrated in Equation 7.

$$F1Score = \frac{2 \times precision \times recall}{precision + recall} \tag{7}$$

III. Related Work

Over the past years, there have been numerous publications which aim to predict stress by applying machine learning algorithms to physiological data collected through wearable devices. The rampant advances in wearable technology enabled researchers to obtain measurements such as Heart Rate (HR), Electrocardiogram (ECG), Blood Volume Pulse (BVP), Respiration Rate (RR), and Galvanic Skin Response (GSR) in a non-intrusive manner with decent accuracy [18] [19] [20], as well as build extensive datasets with such data [10]. While some articles developed improved models from pre-existing datasets, others collected their own data through stress inducing techniques such as Trier Social Stress Test (TSST) [21] and Stroop Color-Word test (SCWT) [22].

The study of [7] used an WEKA Random Forest algorithm to classify HR, BVP, GSR, temperature and wrist acceleration into three stress categories (No Stress, Low Stress and High Stress). On [23] also applied Random Forest as classifier for categorizing HR, HRV, and EDA into stress and no stress. Unlike the two aforementioned articles, on [24] they focused primarily on ECG and Respiratory Inductive Plethysmograph (RIP) measurements from a chest-based device to classify between stressed and not stressed patients using an instance of Multiple Kernel Learning (MKL). Although the authors obtain satisfying results, datasets containing measurements spanning over a long time period may result in inferior performance when using the algorithms mentioned above due to their worse recall capabilities when compared to models based on deep learning techniques.

Other authors employed more advanced algorithms based on Neural Networks, which often provide higher accuracy when working with a high-dimension feature space. Li and Liu applied a deep Convolutional Neural Network (CNN) to WESAD dataset's raw data [10] and obtained accuracy >95% on binary stress classification tasks [25]. The study of [26] also applied CNN to WESAD, but with a normalized dataset which resulted in an accuracy of 87.7% when classifying between baseline, stress, and amusement. As we can see, Neural Networks resulted in a much higher accuracy, but these values could still be improved. By feeding raw data to the model, there is a chance that it may learn incorrect patterns, which can be mitigated by normalizing all features.

Based on what has been said so far, we identified and aim to fill two main gaps in current research: 1) the lack of a clearly specified pre-processing pipeline capable of improving accuracy when training stress detection models, and 2) the dearth of models using Recurrent Neural Networks for detecting stress from physiological measurements.

IV. WESAD Dataset

The main data source for validating the proposed pipeline, as well as training and validating our model comes from the WESAD dataset [10]. The paper published alongside with the dataset also brings results of different machine learning models used on a three-class problem for classifying between baseline, stress, and amusement conditions, as well as one versus all problem for classifying between stress and non-stress conditions. In Section VI we will compare our pipeline and model with the ones presented in the dataset article.

The WESAD dataset consists of data collected from students at the University of Siegen, Germany. The exclusion criteria stated in the study invitation were pregnancy, heavy smokers, and people suffering from mental disorders, chronic, and cardiovascular diseases. The total number of subjects in the experiment were 17 people with ages between 25 and 29 years although, due to sensor malfunction, only 15 subjects have their data available.

The study used two sensor arrays. The first device was located in the chest, which collected motion with a 3 axes accelerometer (ACC), respiration (RESP), electrocardiogram (ECG), electrodermal activity (EDA), electromiogram (EMG), and body temperature (TEMP), all collected with a sample rate of 700Hz. The other device was located on the wrist, which collected blood volume pressure (BVP) at a 64Hz rate, EDA at 4Hz, TEMP at 4Hz, and ACC at 32Hz. Both sensors were manually synchronized through a double tap gesture at the start of the test.

The study protocol aims to evoke neutral (baseline), stress and the amusement conditions in the participants. To do so, after the participants were equipped with both sensing devices,

each aforementioned condition was achieved by the following procedures:

- **Baseline Condition:** Subjects were sitting or standing for 30 minutes and magazines were provided.
- **Amusement Condition:** Subjects watched funny videos for 392 seconds.
- **Stress Condition:** Subjects were exposed to Trier Social Stress Test (TSST) for 10 minutes, where in the first 5 minutes they had to make a public speech about their personal traits, and in the other 5 minutes they had to perform mental arithmetic tasks.

The amusement and stress conditions aim to excite the the subjects. Because of that, after each of these sessions a meditation session took place in order to calm down the subjects. The whole experiment generated a time series with more than 3 million points for each signal collected for each subject in the experiment.

In order to validate the study protocol and obtain the ground truth for each subject's condition during the experiment, a combination of questionnaires was given to the subjects after each induced condition. The questionnaires used were the Positive and Negative Affect Schedule (PANAS), some questions from the State-Trait Axiety Inventory (STAI), Self-Assessment Manikins (SAM), and Short Stress State Questionnaire (SSSQ).

V. MoStress Pipeline

After collecting the data and obtaining the ground truth, the WESAD authors also described the feature extraction process for the collected signals, as well as different supervised machine learning models used to classify both between baseline, stress, and amusement and between stress and non-stress. Since the best model in the 3 class classification problem used only physiological data (excluded ACC data) from the chest sensor, our experiments only used them as well.

The labels for the subject's condition in the dataset correspond to the emotion being stimulated at the moment and later validated by obtaining the ground truth the questionnaires. Because of that, the condition labels are sampled at 700Hz as well.

The method chosen to generate the input data shape and apply all pre-processing step of the pipeline is the sliding window method. This method consists in taking a sample of the available signals inside a time window, and then sliding it with a chosen step to the next time range. The size of the window and the time step used will be described in the following sections.

The pre-processing steps of MoStress pipeline, which are shown in Figure 1, aim to address four problems that can hinder model learning, which are:

- Noise input signal
- Data with different scales
- Transitional states, where the window label is not clear
- Unbalanced data (where labels do not have the same amount of points)

All these three issues may have a huge impact on MoStress model learning. Data with different scales makes it harder for the model to adjust the network parameters. Incorrect window labeling, specially in state transitions, may mislead model classification. Finally, unbalanced data makes it harder for the model to classify the conditions with less data. Each of this problems is discussed in details in the next sessions.

A. Fourier Analysis

If we look at the physiological signals in Figure 3, we can notice that Respiration has the largest period and that ECG has a high frequency. We can also see that the rest of the signals have their main information located in the middle level due their variation frequency being highly noisy. Thus, we applied Discrete Fourier Transform on the ECG signal since it has the highest natural frequency and may be more affected by resampling.

After we applied the Discrete Fourier Transform, we generated the ECG frequency spectrum which can be seen in Figure 2.

The ECG frequency spectrum analysis shows us that the ECG signal has the high frequency component at $0.002Hz$, so we considered it the natural frequency of the data, which corresponds to approximately 566 points of the period. In order to clean the data and avoid aliasing issues, we used only 1 point of data for each 100 points. This value corresponds to a resampling frequency of $0.1428Hz$ which is more than twice the natural frequency and follows the Nyquist rule to avoid aliasing.

B. Rolling Z-Score Normalization

The second step for preparing the data was normalization. Since we chose sliding window as our method, this normalization is also performed on each window. When observing physiological signals, like the ones in Figure 3, we can notice how outliers carry important information about signal itself; for instance, if we look at the ECG signal, Figure 3a, the main source of information comes from the spikes and their frequency, while the rest of the signal can be considered noise. In short, outliers play the main role in signal characterization for this type of data.

Also, since all signals change their mean, or variance or other statistics parameters during the experiment period, we concluded that we were dealing with non-stationary data, hence we need to choose an appropriate method to normalizing this data.

Based on the fact that we need the normalize the data but still preserve the information brought by the outliers, a centralization parameter sensitive to our data is required. With this in mind, we chose a normalization method based on the mean and standard deviation, which is the z-score normalization. We applied the algorithm in each window, which consists of Equation 8:

$$z = \frac{x - \mu}{\sigma} \tag{8}$$

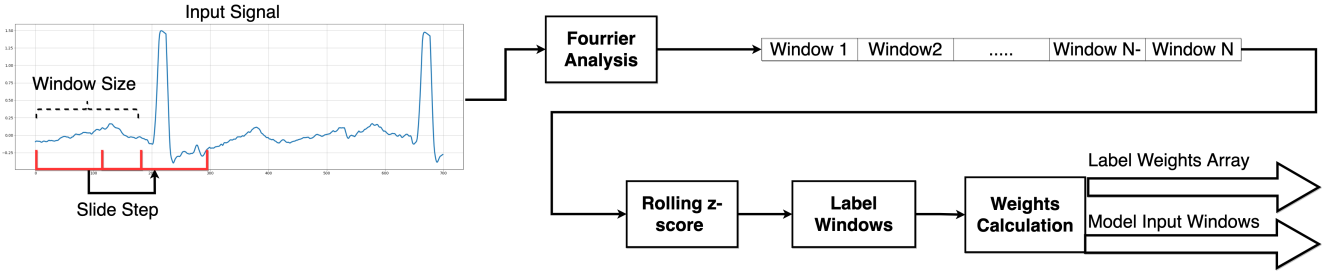


Fig. 1: MoStress Pre-processing Step

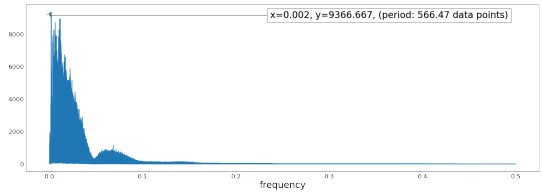
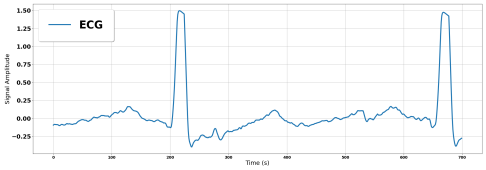
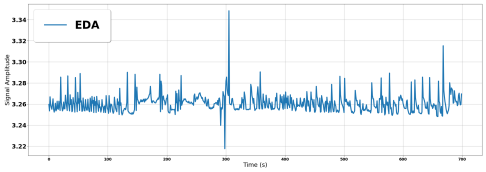


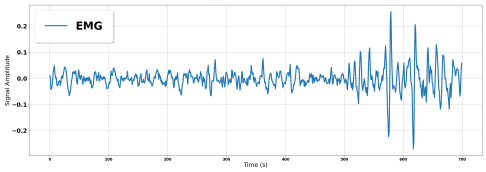
Fig. 2: Frequency Spectrum of ECG Signal



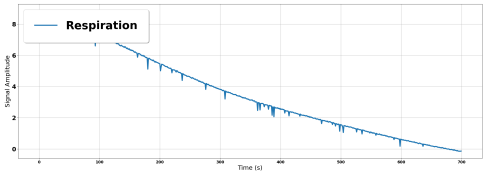
(a) ECG



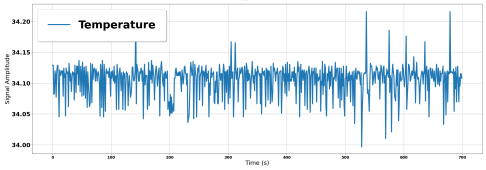
(b) EDA



(c) EMG



(d) Respiration



(e) Temperature

Fig. 3: One-Second Sample

where x are the window's data points, μ the the window's mean, and σ the window's variance.

With this normalization, the outliers of each signal are still in place and the resultant signal has a mean value close to zero and standard deviation close to one, placing all data into a similar range of values.

C. Window's Labels

After normalizing the data based on the sliding window method, it is necessary to attribute a condition label for each window, this was done on three steps:

- Compute the frequency of each label present in the window's time range.
- Take the most frequent label of the time range as the window label.
- Discard windows that have a chosen label frequency smaller than a pre-determined threshold μ.

The last step was done in order to avoid using transition states to train the model. Since the model aims to classify between baseline, stress, and amusement, the neural network needs to have the parameters fitted to those specific states. Knowing that transition states cannot be classified with one specific condition label, the windows which capture state transitions are not suitable to train the model.

After attributing a single label for all available windows, we used all windows and their respective labels from 14 subjects and used as input for the model. The remaining subject was used as validation data for the trained model to predict.

D. Unbalanced Data

As described in IV, the period of inducing for baseline, stress, and amusement conditions was not homogeneous. Thus, the dataset was highly unbalanced, containing more data for baseline than both stress and amusement.

After the windowing step, about 53.18% of the windows were labeled as baseline, 30% as stress and only 16.8% as amusement. In order to increase the model's carefulness with

TABLE I: Models Results for the Three-Class Classification

		LSTM				GRU			
		Model Accuracy	Stress Precision	Stress Recall	Stress F1-Score	Model Accuracy	Stress Precision	Stress Recall	Stress F1-Score
Baseline	Adam	0.81	0.77	0.88	0.82	0.72	0.92	0.63	0.75
	Nadam	0.83	0.96	0.89	0.92	0.59	0.95	0.66	0.78
	RMSProp	0.76	0.88	0.87	0.88	0.73	0.99	0.67	0.08
Non - Baseline	Adam	0.5	0.43	0.74	0.54	0.65	0.61	0.97	0.75
	Nadam	0.81	0.91	0.89	0.90	0.82	0.99	0.92	0.96
	RMSProp	**0.86**	**0.96**	**0.93**	**0.94**	0.6	0.79	0.43	0.55

respect to less frequent labels, we attributed weights to each class. The weight of a class x is calculated by the Equation 9 [27].

$$weight(x) = \frac{\#Samples}{\#Classes \times \#OccurrencesOfClassX} \quad (9)$$

After we calculated the weights for each class, we obtained values for baseline, stress, and amusement of 0.62, 1.11, and 1.98, respectively. These values were used as parameters to train the model.

E. RNN Architecture

The final step of MoStress Pipeline is an RNN model which take as input the pre-processed data from the previous steps and perform the classification task, on this paper, the three-class classification problem. In order to achieve a better architecture or the RNN model, we fist started with a baseline model.

The baseline model created for the three-class classification problem is shown in Figure 4. This model is consisted of 128 LSTM or GRU neurons (both types were tested) fully connected with three dense neurons, one for each class in the problem. In addition to the two types of neurons tested. We also tested the model with the three different optimization functions. In total, six models were trained for each experiment.

All models in all experiments used a softmax activation function in the last dense layer in order to output the probability of each class. In the RNN neurons, we used LeakyReLU as activation function. We split 60% of the subject's data for training and 40% for evaluation. The metric used as loss function for the model was the Sparse Categorical Entropy, which is a measure of entropy for multi-class classification problems.

After running the pipeline and training all six models we obtained the results shown in Figure 5. As we can observe, some of the models were having difficulties to converge, we can see that fact on Table I, where the baseline model, with Adam optimization function, and GRU as neuron, only achieved 0.59 of accuracy. We concluded that the model complexity lies in the number of neurons in the input layer, so we added a Gaussian Noise with 0.5 of standard deviation and a Dropout with 30% of probability of turn off an RNN neuron. In the last three dense neurons, we added a Ridge Regression regularizer with λ of 10^{-6} and 3 as maximum norm for bias.

After these adjustments, all models stopped improved their convergence and we validated them with the pre-processed

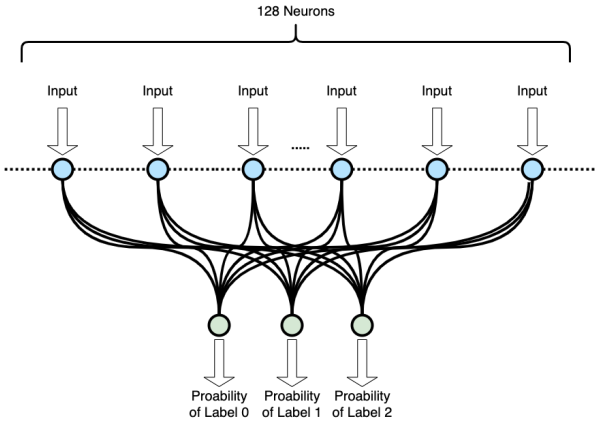


Fig. 4: MoStress RNN Model Architecture

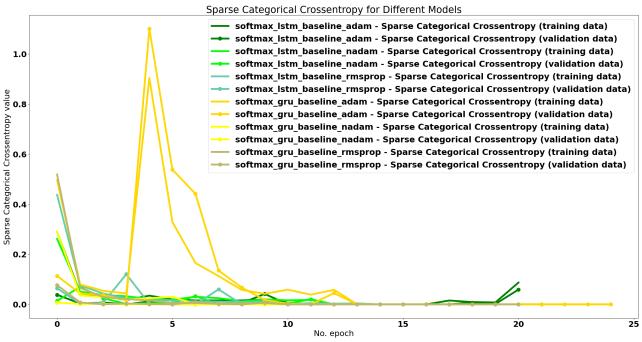


Fig. 5: First Training of Six Models

dataset stored for testing proposes, as described on section VI. The results of this validation are available on table I.

The results in table I show that the model using LSTM neurons in the input layer and RMSProp as optimization function had the best result among all models. Therefore, we chose this model as contribution and to take a closer look in the results section.

VI. Experimental Evaluation

In order to assess our pipeline, we performed several experiments with different implementations of the pipeline, using different parameters for window size, sliding window step, and counter threshold for label windows. We also sanitized the input signals to remove noise and insert on the pipeline only the main information given by them. We also did tests on different RNN architectures and selected the model with better performance.

It is important to mention that we applied the pipeline for all data available on the WESAD dataset. However, we only used the data collected from the chest sensor in our models and also discarded the accelerometer data in order to compare the proposed model with the work that use this dataset.

We also used data from 14 of the 15 subjects to train and test our models and used the remaining subject as unseen data for the model to make new predictions. Our results are based on this new data given to the model.

A. Methodology

All the steps of the pipeline were executed on a machine learning-oriented cluster, which consists of a server with an AMD Epyc processor, with 16 cores and 32 threads, 64 GB of memory and a Tesla T4 GPU.

After we applied the first step of MoStress pre-processing, the Fourrier analysis, we need to determine several parameters for the pipeline pre-processing step, such as:

- Window length;
- Sliding window step;
- Counting threshold mu.

For choosing the first two parameters, we need to consider that the ideal model would be able to use only a few data points on the window, a small window length, and should be able to detect the signal pattern with no past points, which means a infinity sliding window step. Based on that and on other papers which used sliding window technique, as the ones described on Section III, we tested MoStress with a window size of 30 seconds, 1 minute and 1.5 minutes, and we use the sliding window step of 1 second, and the biggest window achieved the better results.

To determine the threshold mu, we can analyze its objective which is: remove transitional windows from the input data in order to help the RNN model to better identify the time series pattern. If mu is equal to 1, this means that we will only accept windows with exclusive one label, and if mu is equal to zero, only the number of labels on the window will determine the window label, meaning that, for instance, if we have a window with 10 labels stress and 11 labels amusement, the window label will be amusement, despite this might be a window on a transitional state. Base on that, we tested the mu equal to 0.5, 0.6 and 0.7 with different windows length and the we achieved the best results with the value of 0.7.

B. Numerical Results

The best model on the three-classification problem has the architecture shown in Figure 4 with the addition of a Dropout and a Gaussian Noise layers. The model uses RMSProp as optimization function and softmax in the last layers as activation function. With this model and all the data pre-processed on the pipeline described in Section V, the full confusion matrix generated by the final model on new data in shown in Table II.

With the model's confusion matrix, the results obtained for each class of the multi classification problem are shown in Table III:

TABLE II: Best Model Confusion Matrix

		Predicted Values		
		Baseline	Stress	Amusement
True Values	Baseline	6122	235	1743
	Stress	110	4339	228
	Amusement	24	20	2435

TABLE III: Best Model Results for Each Class

		Metric		
		Precison	Recall	F1-Score
Label	Baseline	0.98	0.76	0.85
	Stress	0.96	0.93	0.94
	Amusement	0.55	0.98	0.77

In Section III, we can notice that the majority of the work done in stress detection used traditional machine learning algorithms or neural networks. In order to better compare the pipeline efficiency and the proposed model, we will compare our results with other work that also used the WESAD dataset.

Analyzing the results in the paper published by the WESAD dataset authors [10], they achieved an accuracy of 80.34% on the three-class problem, using all chest-based physiological modalities with AdaBoost classifier and an accuracy of 93.12% on the binary case using all chest-based physiological modalities with the Linear Discriminant Analysis classifier. Our model achieved and overall accuracy on the three-class problem with the same chest based data of 86% and, even on the three-class problem, the accuracy for the stress prediction task is of 96.5%, which is higher than the binary case present in the original article [10].

If we look the results of the amusement classification of our best model on Table III, we have poor results for all metrics. We attribute that behavior due the low quantity of amusement data available. As described on Section IV, the amusement condition was imposed on the subjects only 392 seconds, 6 minutes and 30 seconds, while baseline condition had 30 minutes duration and stress condition has 10 minutes.

It is important to mention that the authors pre-processed the dataset and extracted several important features of each signal, and these values were used as input for all different models tested. Therefore, it is safe to claim that either the proposed pipeline had a better performance than the author's feature extraction, or that the proposed RNN model performs better than the classic machine learning models or both.

VII. Conclusion

In this paper we proposed a pipeline for data pre-processing in order to facilitate training models and improving stress classification performance. We also proposed a model based on RNNs which is simple and capable of achieving good results on multi-class classification problems.

As shown in Section VI, it is remarkable how a simple neural network based on RNN neurons featuring a proper data pre-processing can achieve great results on stress prediction tasks. As mentioned on [14], the addition of more layers or units within a layer can represent more complex functions; with this is mind, the combination of a good

data pre-processing and more complex models could be used for processing more complex data and gratly improve stress detection using wearable devices.

Although good results were achieved, each person's body is different, so their body reaction to stress is also different. Based on that, it is hard to create a neural network which adjusts to all kinds of people. In order to do that, the model needs to make good classifications with even smaller windows size.

Future work in this area could consist of fine-tuning the model's hyper parameters to have even better classification performance. The model and pre-processing data pipeline could also be tested on other datasets and despite the lack of different studies using the state of art on the stress classification field, we also will compare our results with the best achieved by now, which consists of some classical machine learning models.

We also started to use the state of art model N-BEATS [28] to classify stress and achieve a better understanding on how this state affects the body through the different physiologic signals.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. This research was partially supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (grant 402679/2021-0), and Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul - FAPERGS (grants 19/2551-0001658-1 and 21/2551-0001944-1). We also thank the support of SAP Labs Latin America for the travel grant to the first author by means of the DevX program.

REFERENCES

[1] H. R. Saloni Dattani and M. Roser, "Mental health," *Our World in Data*, 2021, https://ourworldindata.org/mental-health.

[2] A. J. Ferrari, R. E. Norman, G. Freedman, A. J. Baxter, J. E. Pirkis, M. G. Harris, A. Page, E. Carnahan, L. Degenhardt, T. Vos *et al.*, "The burden attributable to mental and substance use disorders as risk factors for suicide: findings from the global burden of disease study 2010," *PloS one*, vol. 9, no. 4, p. e91936, 2014.

[3] H. C. Ates, A. K. Yetisen, F. Güder, and C. Dincer, "Wearable devices for the detection of covid-19," *Nature Electronics*, vol. 4, no. 1, pp. 13–14, 2021.

[4] A. H. Orabi, P. Buddhitha, M. H. Orabi, and D. Inkpen, "Deep learning for depression detection of twitter users," in *Proceedings of the 5th Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, 2018, pp. 88–97.

[5] T. Eslami, V. Mirjalili, A. Fong, A. R. Laird, and F. Saeed, "Asd-diagnet: a hybrid learning approach for detection of autism spectrum disorder using fmri data," *Frontiers in neuroinformatics*, vol. 13, p. 70, 2019.

[6] L. Santamaria-Granados, M. Munoz-Organero, G. Ramirez-Gonzalez, E. Abdulhay, and N. Arunkumar, "Using deep convolutional neural network for emotion detection on a physiological signals dataset (amigos)," *IEEE Access*, vol. 7, pp. 57–67, 2018.

[7] M. Gjoreski, H. Gjoreski, M. Luštrek, and M. Gams, "Continuous stress detection using a wrist device: in laboratory and real life," in *proceedings of the 2016 ACM inter. joint conf. on pervasive and ubiquitous computing: Adjunct*, 2016, pp. 1185–1193.

[8] S. M. U. Saeed, S. M. Anwar, H. Khalid, M. Majid, and U. Bagci, "Eeg based classification of long-term stress using psychological labeling," *Sensors*, vol. 20, no. 7, p. 1886, 2020.

[9] A. E. Alhitary, E. W. A. Hay, A. K. Al-bashir *et al.*, "Objective detection of chronic stress using physiological parameters," *Medical & biological eng. & comp.*, vol. 56, no. 12, pp. 2273–2286, 2018.

[10] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. Van Laerhoven, "Introducing wesad, a multimodal dataset for wearable stress and affect detection," in *Proceedings of the 20th ACM inter. conf. on multimodal interaction*, 2018, pp. 400–408.

[11] H. Selye, "The evolution of the stress concept: The originator of the concept traces its development from the discovery in 1936 of the alarm reaction to modern therapeutic applications of syntoxic and catatoxic hormones," *American scientist*, vol. 61, no. 6, pp. 692–699, 1973.

[12] E. Charmandari, C. Tsigos, and G. Chrousos, "Endocrinology of the stress response," *Annu. Rev. Physiol.*, vol. 67, pp. 259–284, 2005.

[13] N. Schneiderman, G. Ironson, and S. D. Siegel, "Stress and health: psychological, behavioral, and biological determinants," *Annu. Rev. Clin. Psychol.*, vol. 1, pp. 607–628, 2005.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020.

[16] R. Sen, H.-F. Yu, and I. Dhillon, "Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting," *arXiv preprint arXiv:1905.03806*, 2019.

[17] H. H. Sultan, N. M. Salem, and W. Al-Atabany, "Multi-classification of brain tumor images using deep neural network," *IEEE Access*, vol. 7, pp. 69 215–69 225, 2019.

[18] M. Garbarino, M. Lai, D. Bender, R. W. Picard, and S. Tognetti, "Empatica e3—a wearable wireless multi-sensor device for real-time computerized biofeedback and data acquisition," in *2014 4th Inter. Conf. on Wireless Mobile Communication and Healthcare-Transforming Healthcare Through Innovations in Mobile and Wireless Tech.* IEEE, 2014, pp. 39–42.

[19] J. Hailstone and A. E. Kilding, "Reliability and validity of the zephyr™ bioharness™ to measure respiratory responses to exercise," *Measurement in Physical Education and Exercise Science*, vol. 15, no. 4, pp. 293–300, 2011.

[20] E. Ertin, N. Stohs, S. Kumar, A. Raij, M. Al'Absi, and S. Shah, "Autosense: unobtrusively wearable sensor suite for inferring the onset, causality, and consequences of stress in the field," in *Proceedings of the 9th ACM conference on embedded networked sensor systems*, 2011, pp. 274–287.

[21] C. Kirschbaum, K.-M. Pirke, and D. H. Hellhammer, "The 'trier social stress test'–a tool for investigating psychobiological stress responses in a laboratory setting," *Neuropsychobiology*, vol. 28, no. 1-2, pp. 76–81, 1993.

[22] A. R. Jensen and W. D. Rohwer Jr, "The stroop color-word test: a review," *Acta psychologica*, vol. 25, pp. 36–93, 1966.

[23] F. Delmastro, F. Di Martino, and C. Dolciotti, "Cognitive training and stress detection in mci frail older people through wearable sensors and machine learning," *IEEE Access*, vol. 8, pp. 65 573–65 590, 2020.

[24] K. Hovsepian, M. Al'Absi, E. Ertin, T. Kamarck, M. Nakajima, and S. Kumar, "cstress: towards a gold standard for continuous stress assessment in the mobile environment," in *Proceedings of the 2015 ACM inter. joint conf. on pervasive and ubiquitous computing*, 2015, pp. 493–504.

[25] R. Li and Z. Liu, "Stress detection using deep neural networks," *BMC Medical Informatics and Decision Making*, vol. 20, no. 11, pp. 1–10, 2020.

[26] A. Kumar, K. Sharma, and A. Sharma, "Hierarchical deep neural network for mental stress state detection using iot based biomarkers," *Pattern Recognition Letters*, vol. 145, pp. 81–87, 2021.

[27] G. King and L. Zeng, "Logistic regression in rare events data," *Political analysis*, vol. 9, no. 2, pp. 137–163, 2001.

[28] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-beats: Neural basis expansion analysis for interpretable time series forecasting," *arXiv preprint arXiv:1905.10437*, 2019.