

AI Driving Classification

Licenciatura em Engenharia Informática

Alberto Manuel de Matos Pingo, nº 2202145

João Pedro Quintela de Castro, nº 2201781

Leiria, julho de 2024

AI Driving Classification

Licenciatura em Engenharia Informática

Alberto Manuel de Matos Pingo, nº 2202145

João Pedro Quintela de Castro, nº 2201781

Trabalho de Projeto da unidade curricular de Projeto Informático realizado sob a orientação do Professor Sílvio Priem Mendes, da Professora Doutora Anabela Moreira Bernardino e do Professor Doutor Paulo Jorge Gonçalves Loureiro.

Leiria, julho de 2024

Dedicatória

Inserir aqui a dedicatória. Trata-se de um elemento **facultativo**.

Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória.
Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto
da dedicatória. Texto da dedicatória. Texto da dedicatória.

Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória.
Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto da dedicatória. Texto
da dedicatória. Texto da dedicatória. Texto da dedicatória.

Agradecimientos

Inserir aqui os agradecimentos. Trata-se de um elemento **facultativo**.

[illegible]

Texto dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto
dos agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos
agradecimentos. Texto dos agradecimentos. Texto dos agradecimentos. Texto dos
agradecimentos. Texto dos agradecimentos.

Resumo

Este trabalho aborda a utilização de Inteligência Artificial, especificamente Redes Neurais do tipo LSTM (Long Short-Term Memory), para a classificação de comportamentos de condução. O objetivo principal é desenvolver um modelo capaz de identificar padrões de condução agressiva e não agressiva utilizando dados de sensores de dispositivos móveis.

A metodologia empregue inclui a aquisição e pré-processamento de dados de sensores, o desenvolvimento e treinamento de um modelo de Redes Neurais Recorrentes LSTM, e a validação e avaliação do desempenho do modelo.

Os resultados mostram que o modelo desenvolvido consegue classificar de forma eficaz os diferentes tipos de comportamento de condução, apresentando uma alta taxa de acurácia. As conclusões indicam que a aplicação de IA na classificação de comportamentos de condução pode contribuir significativamente para a segurança viária e a otimização de sistemas de condução autônoma.

Palavras-chave: Inteligência Artificial, Redes Neurais, LSTM, RNN, Condução, Classificação

##PÁGINA IMPAR##

Abstract

This work addresses the use of Artificial Intelligence, specifically Long Short-Term Memory (LSTM) Neural Networks, for the classification of driving behaviors. The main objective is to develop a model capable of identifying aggressive and non-aggressive driving patterns using data from mobile device sensors.

The methodology employed includes the acquisition and preprocessing of sensor data, the development and training of an LSTM Recurrent Neural Network model, and the validation and evaluation of the model's performance.

The results show that the developed model can effectively classify different types of driving behavior, presenting a high accuracy rate. The conclusions indicate that the application of AI in driving behavior classification can significantly contribute to road safety and the optimization of autonomous driving systems.

Keywords: Artificial Intelligence, Neural Networks, LSTM, RNN, Driving, Classification

##PÁGINA IMPAR##

Índice

Dedicatória	ii
Agradecimentos.....	iii
Resumo.....	iv
Abstract	v
Lista de Figuras	x
Lista de tabelas.....	xi
Lista de siglas e acrónimos.....	xii
1. Introdução.....	1
1.1. Enquadramento do tema	1
1.2. Justificação e Pertinência do Tema	1
1.3. Objetivos	1
1.4. Métodos e técnicas utilizados	1
1.4.1. Aquisição de Dados.....	1
1.4.2. Pré-processamento de Dados	2
1.4.3. Desenvolvimento do Modelo	2
1.4.4. Treino e Validação	2
1.4.5. Avaliação do Desempenho.....	2
1.5. Estrutura do trabalho	2
2. Metodologia.....	3
2.1. Reuniões	3
3. Fundamentação científica.....	4
3.1. Inteligência Artificial	4
3.2. Redes Neurais (RN).....	5
3.3. Redes Neurais Recorrentes (RNN)	6

3.4.	Long Short-Term Memory (LSTM)	8
3.4.1.	Arquitetura de uma LSTM: Estrutura e Funcionamento	9
3.4.2.	Forget Gate	9
3.4.3.	Input Gate	10
3.4.4.	Output Gate	11
3.5.	LSTM VS RNN	12
3.6.	Parâmetros comuns de LSTM	13
3.7.	Arquitetura Bidirecional	15
3.7.1.	Estrutura e Funcionamento	15
3.7.2.	Vantagens e Desvantagens	15
3.8.	Redes Neurais Convolucionais	16
3.9.	Redes Convolucionais Unidimensionais	16
3.9.1.	Estrutura e Funcionamento	16
3.9.2.	Vantagens e Desvantagens	18
3.10.	----Teorias e Conceitos principais	Error! Bookmark not defined.
3.11.	----Pesquisas e Estudos Relevantes	Error! Bookmark not defined.
3.12.	----Identificação de Lacunas na Literatura	Error! Bookmark not defined.
4.	Estado de arte	19
4.1.1.	Tecnologias e Ferramentas Atuais	19
4.2.	Tecnologias e ferramentas utilizadas	19
4.2.1.	Linguagens	19
4.2.2.	Bibliotecas	19
4.2.3.	Plataformas	22
5.	Trabalhos Relacionados	23
5.1.	Driving Behavior Classification Based on Sensor Data Fusion Using LSTM Recurrent Neural Networks	23

5.2. Driving Behavior Classification Based on Oversampled Signals of Smartphone Embedded Sensors Using an Optimized Stacked-LSTM Neural Networks 26

6. Análise	30
6.1. Análise dos sensores de dispositivos móveis.....	30
6.1.1. Definição e Função.....	30
6.1.2. Tipos de Sensores Utilizados	30
6.2. Análise dos datasets.....	31
6.2.1. Estrutura dos Dados	31
7. Arquiteturas propostas	32
7.1. Stacked LSTM.....	32
7.1.1. Introdução ao Stacked LSTM	32
7.1.2. Estrutura da Stacked LSTM	32
7.2. Bidirectional LSTM	33
7.2.1. Introdução ao Bidirectional LSTM	33
7.2.2. Estrutura da Bidirectional LSTM.....	33
7.3. Convolutional LSTM	33
7.3.1. Introdução ao Convolutional LSTM	33
7.3.2. Estrutura da Convolutional LSTM.....	34
8. Desenvolvimento.....	35
8.1. Implementação	35
8.2. Descrição e Caracterização dos Dados	36
8.2.1. Estrutura dos Dados	36
8.2.2. Sensores Utilizados	36
8.3. Tratamento dos Dados	36
8.3.1. Processo de Tratamento de Dados	37
8.4. Classificação dos Dados	39
8.4.1. Descrição dos Parâmetros de Entrada	39

8.4.2.	Processo de Classificação dos Dados	39
8.5.	Normalização dos Dados	41
8.5.1.	Processo de Normalização dos Dados	41
8.6.	Representação Visual dos Dados.....	43
8.7.	Separação dos Dados em Treino e Teste	43
8.7.1.	Processo de Separação dos Dados	43
8.8.	Criação do Modelo	44
8.8.1.	Estrutura do Modelo	45
8.9.	Compilação e Treino	45
8.10.	Resultados	Error! Bookmark not defined.
9.	Testes	46
9.1.	Objetivos dos Testes	46
9.2.	Estratégia.....	Error! Bookmark not defined.
9.3.	Métricas de Avaliação	46
9.4.	Resultados dos Testes	48
9.5.	Análise de resultados	48
9.6.	Considerações Finais	49
10.	Conclusões	50
	Bibliografia ou Referências Bibliográficas	52
	Anexos	53
	Glossário	54

Lista de Figuras

Figura 1 - Software Utilizado para Capturar os Dados do UAH-driveset. FONTE: Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks (2017)	24
Figura 2 - Gráfico Ilustrativo do Resampling dos Dados. FONTE: Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks (2017)	25
Figura 3 - Resultados da Matriz de Confusão da Implementação Proposta em Comparação com a Classificação de Comportamento de Condução Descrita pelo uah-driveset. FONTE: Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks (2017)	25
Figura 4.1 - Texto ilustrativo da figura 1.	Error! Bookmark not defined.
Figura 4.2 - Texto ilustrativo da figura 2.	Error! Bookmark not defined.
Figura 6 - Diagrama Ilustrativo de uma RNN	6
Figura 7 - Diagrama Ilustrativo de uma LSTM.....	9
Figura 8 - Diagrama Ilustrativo da Forget Gate	10
Figura 9 - Diagrama Ilustrativo da Input Gate	11
Figura 10 - Diagrama Ilustrativo da Output Gate.....	12
Figura 11 - Diagrama Representativo de uma LSTM Bidirecional. FONTE: A Deep Learning Approach for Human Activities Recognition From Multimodal Sensing Devices (2020)	15
Figura 12 - Estágios e Camadas de uma Rede Neural Convolucional. FONTE: Adaptado de Katagaru (2020).....	16
Figura 13 - Diagrama Ilustrativo de uma Implementação CNN 1D usando Max Pooling. FONTE: Predicting the Travel Distance of Patients to Access Healthcare using Deep Neural Networks (2021)	17
Figura 14 - Diagrama Ilustrativo do Tratamento dos Dados	38
Figura 15 - Diagrama Ilustrativo Da Classificação dos Dados.....	40
Figura 16 - Diagrama Ilustrativo Da Função Max Of Vectors Aplicada ao Acelerometro	41
Figura 17 - Diagrama Ilustrativo Da Função Max Of Vectors Aplicada ao Giroscópio	42
Figura 18 - Diagrama Ilustrativo da Normalização dos Dados	42
Figura 19 - Diagrama Ilustrativo da Separação dos Dados	44
Figura 20 - Diagrama Ilustrativo Da Estrutura do Modelo	44
Figura 21 - Gráfico Ilustrativo do Desempenho do Modelo ao Longo de Trinta Épocas.....	Error! Bookmark not defined.

Lista de Tabelas

Tabela 1 - Tabela representativa das reuniões e dos tópicos abordados	3
Tabela 3 - Tabela Comparativa entre LSTM e RNN	13
Tabela 4 - Glossário de Cores Utilizadas nos Diagramas Ilustrativos	35

Lista de siglas e acrónimos

Elemento a figurar, **quando aplicável**.

CNN	Convolutional Neural Networks
ESTG	Escola Superior de Tecnologia e Gestão
IA	Inteligência Artificial
LSTM	Long Short-Term Memory
LLM	Large Language Model
JSON	JavaScript Object Notation
RNN	Recurrent Neural Network

Cuidados na elaboração da lista de siglas e acrónimos:

- Ordenação alfabética;
- Apenas as que sejam relevantes para a leitura do texto.

Adicionar mais entradas à tabela, caso seja necessário (a tabela não tem contornos, mas está no texto).

1. Introdução

O presente projeto, intitulado “AI Driving Classification”, propõe-se a explorar, analisar e classificar a condução de condutores através de uma rede neuronal. Este projeto surge da necessidade de compreender os padrões de comportamento ao volante e as implicações que têm na segurança rodoviária.

1.1. Enquadramento do tema

O objetivo central deste projeto é a classificação da condução com base em dados adquiridos através de uma aplicação. Por meio da análise destes dados, procura-se identificar padrões de comportamento do condutor, tais como tipo de aceleração, tipo de travagem, entre outros indicadores relevantes para a classificação.

1.2. Justificação e Pertinência do Tema

A análise da condução é de extrema importância em diversos contextos, desde a segurança rodoviária ou até ao desenvolvimento de sistemas de assistência á condução. Compreender os padrões de condução pode contribuir significativamente para a prevenção de acidentes rodoviários.

1.3. Objetivos

Os objetivos gerais deste projeto consistem em desenvolver um modelo de classificação de condução o mais preciso e confiável possível, capaz de identificar diferentes estilos e padrões direção.

1.4. Métodos e técnicas utilizados

1.4.1. Aquisição de Dados

Utilizou-se um conjunto de dados de comportamentos de condução obtido a partir da Aplicação Movel desenvolvida por

1.4.2. Pré-processamento de Dados

##TODO## Os dados foram pré-processados para remover ruídos e normalizar as variáveis. Técnicas como suavização de sinais e segmentação de janelas temporais foram aplicadas.

1.4.3. Desenvolvimento do Modelo

Um modelo LSTM foi implementado utilizando a biblioteca TensorFlow. A arquitetura do modelo foi otimizada através de ajustes de hiperparâmetros.

1.4.4. Treino e Validação

O modelo foi treinado com uma divisão de dados em treino, validação e teste. O treino foi realizado utilizando o algoritmo de otimização Adam e a função de perda binary cross-entropy.

1.4.5. Avaliação do Desempenho

O desempenho do modelo foi avaliado utilizando métricas como precisão, recall, F1-score e **AUC-ROC**. Testes adicionais foram realizados para avaliar a robustez do modelo em diferentes cenários de condução.

1.5. Estrutura do trabalho

O relatório está organizado da seguinte forma:

- Capítulo 1 – Introdução: Apresenta o objeto, justificativa, objetivos, métodos e a estrutura do trabalho.
- Capítulo 2 - Fundamentação Teórica: Aqui fazemos a revisão da literatura existente sobre o nosso tema.
- Capítulo 3 - Estado da Arte: Tecnologias Atuais e limitações

2. Metodologia

2.1. Reuniões

Um dos pilares fundamentais do planeamento do projeto foram as reuniões que foram realizadas ao longo de todo o processo. Nestas reuniões estiveram presentes os orientadores do projeto e os alunos que desenvolveram o mesmo.

Estas reuniões foram realizadas de semana a semana com o intuito de monitorar o processo de desenvolvimento do projeto visando analisar todo o trabalho feito na semana antes da reunião e projetar novos objetivos para a semana seguinte.

Com estas reuniões conseguimos identificar possíveis erros no desenvolvimento do projeto e otimizações que puderam ser feitas e melhorando assim a qualidade do produto entregue.

Foram realizadas __ reuniões no total, tendo como tópicos abordados os seguintes:

Tabela 1 - Tabela representativa das reuniões e dos tópicos abordados

3. Fundamentação científica

3.1. Inteligência Artificial

A inteligência artificial (IA) emergiu como um campo de estudo e aplicação que está a revolucionar diversos setores e indústrias, trazendo soluções inovadoras e melhorias significativas em processos e produtos.

A origem da inteligência artificial tem mais de 50 anos, com base nos estudos iniciais de cientistas como Alan Turing, John McCarthy e Marvin Minsky. Entretanto, somente nos últimos anos é que os progressos tecnológicos, principalmente na área do processamento de dados e algoritmos de aprendizagem, colocaram a inteligência artificial em destaque.

Atualmente, a inteligência artificial é utilizada em diversas aplicações, como assistentes virtuais em dispositivos móveis, sistemas de diagnóstico médico e veículos autónomos. Empresas de todos os tamanhos e áreas estão constantemente à procura de formas de utilizar a inteligência artificial para melhorar a eficiência operacional, personalizar a experiência do cliente e impulsionar a inovação.

Além disso, a inteligência artificial está a mudar o modo como os profissionais realizam as suas tarefas, trazendo novas competências e exigindo um maior conhecimento dos dados e algoritmos. Com o avanço da IA, questões éticas, de transparência, privacidade e segurança dos dados se tornam cada vez mais relevantes, sendo preciso desenvolver uma abordagem cautelosa e responsável na criação e aplicação de sistemas de IA.

No campo da condução, a inteligência artificial tem sido fundamental para o desenvolvimento dos veículos autónomos. A IA é essencial para que este tipo de carros consigam entender o ambiente que os rodeiam, tomar decisões rapidamente e navegar com segurança nas estradas. Tecnologias avançadas de inteligência artificial, como redes neurais profundas e algoritmos de visão computacional, possibilitam que os veículos autónomos identifiquem sinais de trânsito, pedestres, outros veículos e obstáculos, agindo de forma similar a um ser humano a dirigir. Além disso, a IA também está a ser utilizada para otimizar rotas, prever condições de tráfego e melhorar a eficiência energética dos veículos, contribuindo para uma condução mais segura, eficiente e autónoma.

3.2. Redes Neurais (RN)

As redes neurais representam uma abordagem poderosa e versátil no campo da inteligência artificial, inspirada no funcionamento do cérebro humano.

Redes neurais, ou sistemas neurais artificiais, são estruturas computacionais compostas por unidades interconectadas, denominadas neurónios artificiais. Estes neurónios, semelhantes aos neurónios biológicos, recebem inputs, processam-nos através de operações matemáticas e produzem saídas. A interconexão de múltiplos neurónios em camadas forma a arquitetura de uma rede neural.

A arquitetura de uma rede neural geralmente inclui uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada camada contém um conjunto de neurónios interligados, e as conexões entre os neurónios têm pesos que são ajustados durante o treino da rede. Este treino visa minimizar uma função de perda, que quantifica a discrepância entre as previsões da rede e os valores reais, através de algoritmos de otimização.

Existem diversos tipos de redes neurais, cada uma com arquiteturas e aplicações específicas.

As redes neurais têm uma vasta gama de aplicações em diversos domínios, incluindo reconhecimento de padrões, processamento de linguagem natural, visão computacional, previsão de séries temporais, entre outros. São utilizadas com sucesso em tarefas como reconhecimento de voz, classificação de imagens, tradução automática e diagnóstico médico.

Apesar dos avanços significativos, as redes neurais ainda enfrentam desafios como o sobreajuste, a interpretabilidade e a eficiência computacional. No entanto, com o desenvolvimento contínuo de novas arquiteturas e técnicas de treino, as redes neurais estão cada vez mais a tornar-se uma ferramenta poderosa para resolver problemas complexos em diversas áreas.

3.3. Redes Neurais Recorrentes (RNN)

Uma rede neural recorrente (RNN) é um modelo de “Deep Learning” treinado para processar e converter uma entrada de dados sequenciais numa saída de dados sequenciais específica.

Um RNN é um tipo de sistema que consiste em muitos componentes interconectados que tentam imitar a maneira como os humanos realizam conversões sequenciais de dados, como traduzir textos de um idioma para outro. Cada vez mais as RNNs estão a ser substituídas por IA e LLM, que são muito mais eficientes no processamento sequencial de dados.

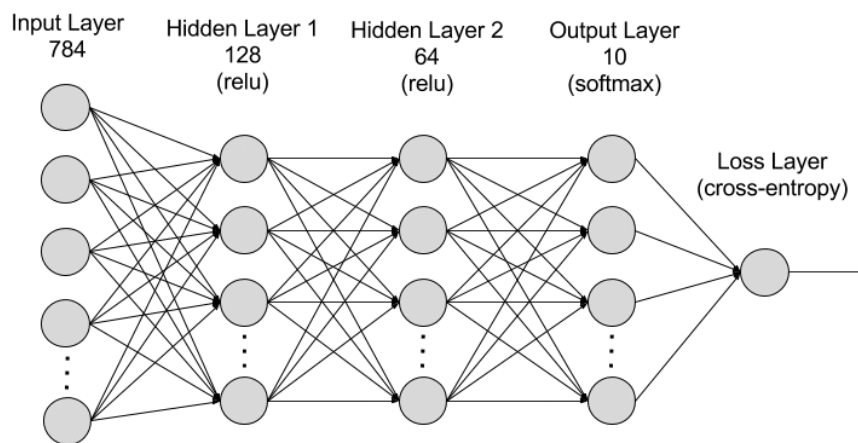


Figura 1 - Diagrama Ilustrativo de uma RNN

3.3.1. Tipos de RNN

- **One-to-Many**

- Este tipo RNN canaliza uma entrada para várias saídas. Ele permite aplicações linguísticas como legendagem de imagens, gerando uma frase a partir de uma única palavra-chave.

- **Many-to-Many**

- O modelo usa múltiplas entradas para prever múltiplas saídas. Por exemplo, podemos criar um tradutor com uma RNN, que analisa uma frase e estrutura corretamente as palavras num idioma diferente.

▪ Many-to-One

- Várias entradas são mapeadas para uma saída. Isto é útil em aplicações como análise de sentimentos, onde o modelo prevê sentimentos dos utilizadores como positivos, negativos e ou neutros a partir de depoimentos de entrada.

3.3.2. Estrutura e Funcionamento das RNNs

Uma RNN é composta por unidades recorrentes que iteram através de uma sequência de dados. Em cada ponto de tempo, a RNN usa uma entrada atual e o estado anterior para produzir uma saída e um novo estado. Este processo é descrito pelas seguintes equações:

$$h_t = \sigma(W_t \cdot x_t + U_h \cdot h_{t-1} + b_h)$$

$$y_t = \phi(W_y \cdot h_t + b_y)$$

- h_t é o estado oculto no tempo t.
- x_t é a entrada no tempo t.
- W_t e U_h são pesos de entrada e recorrentes, respetivamente.
- b_h é o bias do estado oculto.
- y_t é a saída no tempo t.
- W_y é o peso de saída.
- b_y é o bias da saída.
- σ e ϕ são funções de ativação, geralmente não lineares, como tanh ou ReLU.

As RNNs diferenciam-se das redes feedforward pela capacidade de utilizarem estados anteriores para influenciar as saídas atuais e isso é o que permite a captura de dependências temporais nos dados.

3.3.3. Vantagens e Limitações das RNNs

3.3.3.1. Vantagens

Capacidade Sequencial: As RNNs são projetadas para capturar dependências temporais, tornando-as ideais para tarefas sequenciais.

Flexibilidade: Podem ser aplicadas a uma ampla gama de problemas, desde processamento de linguagem natural até previsão de séries temporais.

3.3.3.2. Limitações

Problema do Gradiente Desaparece/Explode

O problema do gradiente desaparece/explode é uma das principais dificuldades encontradas no treino de Redes Neurais Recorrentes (RNNs). Este problema impacta negativamente a capacidade de a rede aprender dependências de longo prazo nos dados sequenciais, o que é crucial para muitas aplicações práticas.

Durante o treino da rede, o algoritmo de retropropagação (backpropagation) é usado para ajustar os pesos da rede. Este processo envolve o cálculo do gradiente do erro em relação a cada peso, que é utilizado para atualizar os pesos de maneira a minimizar o erro. No entanto, em redes profundas e RNNs, os gradientes podem sofrer um de dois tipos de problemas:

- **Vanishing Gradient Problem (Gradiente Desaparece):**
 - Os gradientes dos estados anteriores tendem a diminuir exponencialmente à medida que a informação é propagada de volta através da rede. Isto resulta em gradientes muito pequenos, o que significa que os pesos destas camadas são atualizados muito lentamente. Como resultado, a rede tem dificuldade em aprender e ajustar estes pesos corretamente, especialmente para sequências longas.
- **Exploding Gradient Problem (Gradiente Explode):**
 - Trata-se do oposto do problema do gradiente a desaparecer, sendo que neste problema os gradientes podem aumentar exponencialmente, resultando em valores extremamente grandes. Isto leva a atualizações de pesos muito grandes, que podem causar divergência no processo de treino, tornando o modelo instável e incapaz de aprender.

Dificuldade em Capturar Dependências de Longo Prazo: As RNNs tradicionais podem ter dificuldade em lembrar informações de estados distantes no tempo.

3.4. Long Short-Term Memory (LSTM)

Long Short-Term Memory é uma versão melhorada da recurrent neural network (RNN) projetada por Hochreiter & Schmidhuber. O LSTM é adequado para tarefas de previsão de sequências e destaca-se na captura de dependências de longo prazo.

Uma RNN tradicional possui um único estado oculto que é transmitido ao longo do tempo, o que pode dificultar para a rede aprender dependências de longo prazo. As LSTMs resolvem esse problema introduzindo uma célula de memória, que é um contêiner que pode armazenar informações por um período prolongado. As redes LSTM são capazes de aprender dependências de longo prazo em dados sequenciais, o que as torna adequadas para tarefas como tradução de linguagem, reconhecimento de fala e previsão de séries temporais. As LSTMs também podem ser usadas em combinação com outras arquiteturas de rede neural, como Redes Neurais Convolucionais (CNNs) para análise de imagens e vídeos.

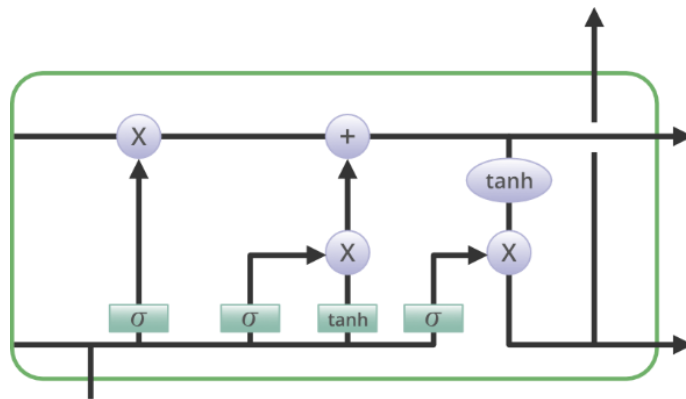


Figura 2 - Diagrama Ilustrativo de uma LSTM

3.4.1. Arquitetura de uma LSTM: Estrutura e Funcionamento

Uma LSTM é composta por uma série de células de memória, que são blocos responsáveis por armazenar e processar informações ao longo do tempo. Cada célula LSTM contém três Gates: Input Gate, Forget Gate, Output Gate.

3.4.2. Forget Gate

A informação que já não é útil no estado da célula é removida com a porta de esquecimento. Dois inputs, x_t (entrada no tempo específico) e h_{t-1} (saída da célula anterior), são alimentados na porta e multiplicados por matrizes de pesos, seguido pela adição de um bias. O resultado é passado por uma função de ativação que fornece uma saída binária.

Se, para um determinado estado da célula, a saída for 0, a informação é esquecida e, para saída 1, a informação é retida para uso futuro. A equação para a Forget Gate é a seguinte:

$$f_t = (W_f [h_{t-1}, x_t] + b_f)$$

onde:

- W_f representa a matriz de pesos associada à Forget Gate.
- $[h_{t-1}, x_t]$ denota a concatenação da entrada atual e do estado oculto anterior.
- b_f é o bias com a Forget Gate.
- σ é a função de ativação sigmoide

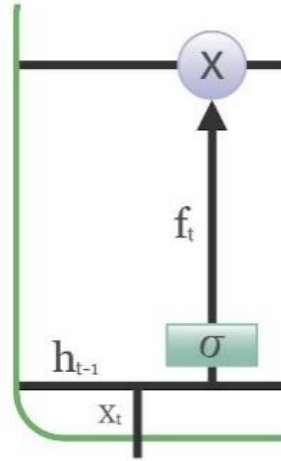


Figura 3 - Diagrama Ilustrativo da Forget Gate

3.4.3. Input Gate

A adição de informações úteis ao estado da célula é feita pela Input Gate. Primeiro, a informação é regulada usando a função sigmoide que filtra os valores a serem lembrados usando as entradas h_{t-1} e x_t . Depois, um vetor é criado usando a função \tanh que fornece uma saída de -1 a +1, que contém todos os valores possíveis de h_{t-1} e x_t . Por fim, os valores do vetor e os valores regulados são multiplicados para obter as informações úteis. A equação para a Input Gate é:

$$i_t = (W_i [h_{t-1}, x_t] + b_i)$$

$$\hat{c}_t = \tanh (W_c [h_{t-1}, x_t] + b_c)$$

Multiplicamos o estado anterior por f_t não utilizando a informação que optamos anteriormente por ignorar. A seguir, incluímos $i_t * C_t$. Isto representa os valores candidatos atualizados, ajustados pelo valor que escolhemos para atualizar cada “state value”.

$$C_t = f_t C_{t-1} + i_t \hat{c}_t$$

onde:

- \odot denota multiplicação elemento a elemento
- \tanh é a função de ativação \tanh

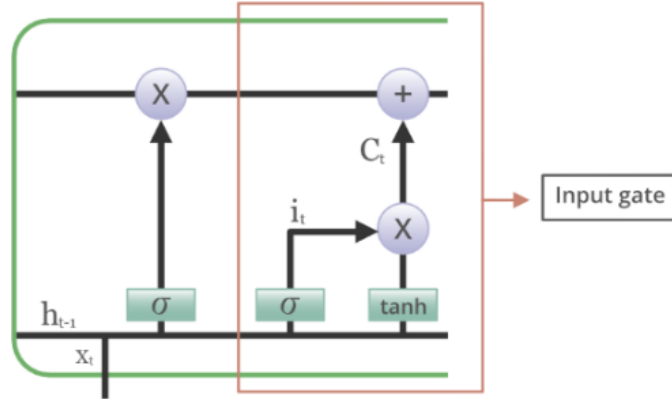


Figura 4 - Diagrama Ilustrativo da Input Gate

3.4.4. Output Gate

Quem extrai as informações úteis do estado atual da célula para serem apresentadas como “output” é a Output Gate. Primeiro, um vetor é gerado aplicando a função \tanh na célula. Depois, a informação é regulada através da função sigmoide e filtrada pelos valores a serem lembrados através das entradas h_{t-1} e x_t . Por fim, os valores do vetor e os valores regulados são multiplicados para serem enviados como “output” e “input” para a próxima célula. A equação para a Output Gate é:

$$o_t = (W_o [h_{t-1}, x_t] + b_o)$$

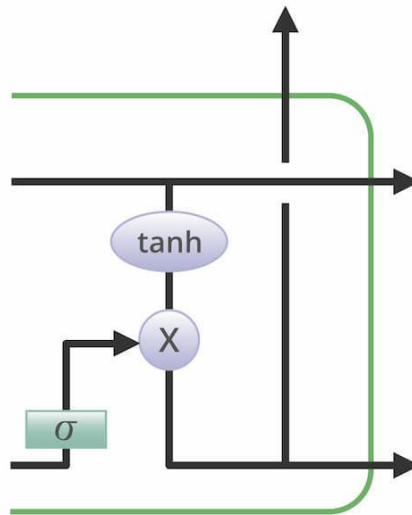


Figura 5 - Diagrama Ilustrativo da Output Gate

3.5. LSTM VS RNN

As RNNs são redes neurais com conexões cíclicas, permitindo que informações anteriores influenciem as previsões atuais. No entanto, as RNNs padrão sofrem de um problema conhecido como "vanishing gradient", onde as informações relevantes podem ser perdidas ao longo do tempo devido à propagação do gradiente. Isso limita a sua capacidade de reter informações importantes em sequências longas.

Por outro lado, as LSTMs foram projetadas para resolver este problema introduzindo unidades de memória especiais chamadas "memory cells". Estas células possuem uma estrutura mais complexa de várias gates, que regulam o fluxo de informações na rede. Isso permite que as LSTMs aprendam quais informações devem ser lembradas ou esquecidas ao longo do tempo, facilitando a captura de dependências de longo prazo.

Em termos de desempenho, as LSTMs tendem a superar as RNNs convencionais em tarefas que exigem modelagem de sequências mais longas, devido à sua capacidade de reter informações por períodos prolongados. No entanto, as LSTMs são mais complexas computacionalmente e podem ser mais difíceis de treinar em conjuntos de dados menores devido à quantidade maior de parâmetros.

Tabela 2 - Tabela Comparativa entre LSTM e RNN

Funcionalidade	LSTM	RNN
Memoria	Tem uma unidade de memória específica que possibilita a aprendizagem de sequência de dados	Não tem uma unidade de memória
Direccionalidade	Pode ser treinada para processar data em várias direções	Só pode ser treinada para processar data numa direção
Treino	Mais difícil devido a complexidade das gates e da unidade de memoria	Mais fácil de treinar
Long-term dependency learning	Sim	Limitado
Capacidade de aprender dados sequenciais	Sim	Sim

3.6.Parâmetros comuns de LSTM

A seguir, estão listados os parâmetros comuns que podem ser configurados ao utilizar camadas LSTM em modelos de redes neurais:

Units

- Refere-se ao número de unidades LSTM na camada.
- Por exemplo, LSTM(50) define uma camada LSTM com 50 unidades.

Input Shape

- Corresponde a um formato tridimensional: (batch_size, timesteps, features)
 - **batch_size**: Indica o número de amostras em cada lote de dados.

- *timesteps*: Representa o número de passos de tempo em cada sequência de entrada.
- *features*: Indica o número de características em cada passo de tempo.

Activation

- Refere-se à função de ativação aplicada às saídas da camada.

Recurrent Activation

- Indica a função de ativação usada para o portão de ativação recorrente.

Utilização de Bias

- Indica se a camada utiliza um bias nas suas operações.

Kernel Initializer

- Refere-se ao esquema de inicialização para os pesos da camada.

Recurrent Initializer

- Indica o esquema de inicialização para os pesos da conexão recorrente.

Bias Initializer

- Indica o esquema de inicialização para os bias.

```
• lstm_layer = LSTM(units=64,activation='tanh',  
    recurrent_activation='sigmoid', use_bias=True, dropout=0.2,  
    recurrent_dropout=0.2, return_sequences=True, return_state=False,  
    kernel_regularizer=None, recurrent_regularizer=None,  
    bias_regularizer=None)
```

3.7. Arquitetura Bidirecional

A arquitetura bidirecional tem sido amplamente escolhidas para desempenhar tarefas que requerem uma compreensão profundo das direções de uma sequência. Este tipo de arquitetura trata-se de uma extensão das LSTM's tradicionais que processam a informação sequencialmente em ambas as direções fazendo com que seja possível captar padrões contextuais mais eficazmente.

3.7.1. Estrutura e Funcionamento

As LSTM's Bidirecionais, ou BiLSTM, consistem em duas redes LSTM's combinadas. Uma processa a sequência que entra, na direção forward, ou seja, da esquerda para a direita, e outra na direção backward, da direita para a esquerda. Este tipo de estrutura permite que a rede capte dependências temporais passadas e futuras melhorando assim a qualidade dos dados que posteriormente irá ser treinado.

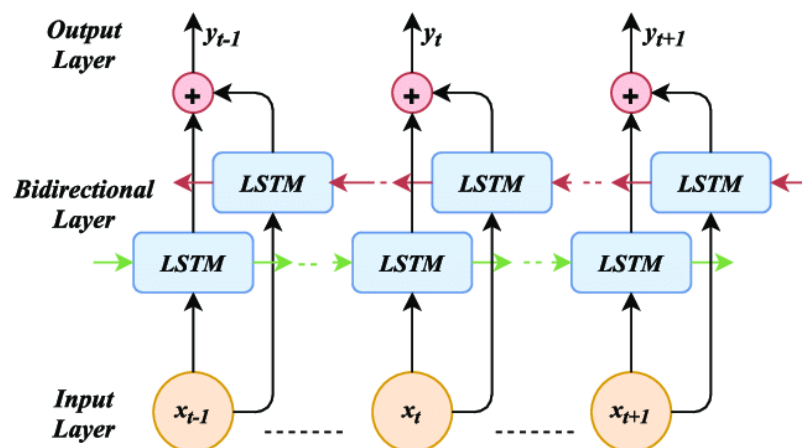


Figura 6 - Diagrama Representativo de uma LSTM Bidirecional. FONTE: A Deep Learning Approach for Human Activities Recognition From Multimodal Sensing Devices (2020)

3.7.2. Vantagens e Desvantagens

Teoricamente, o uso desta arquitetura influencia, de forma positiva, significativamente na precisão do modelo pela sua capacidade de considerar não só o contexto passado, mas também o contexto futuro de cada ponto na sequência.

Embora as BiLSTMS's possam ter um impacto significativo no desempenho do modelo, também apresentam algumas desvantagens que devemos considerar. O uso deste tipo de arquitetura tem um impacto considerável na complexidade computacional por serem muito mais complexas que as LSTM's tradicionais. Também tem uma maior quantidade de parâmetros o que poderá levar ao aumento do consumo de memória. Poderá existir um risco

de Overfitting, especialmente em datasets pequenos, devido ao aumento de parâmetros que deve ser controlado com os respectivos métodos de contenção.

3.8. Redes Neurais Convolucionais

As convolution neural network (CNN) são um tipo de redes neurais que foram propostas pelo pesquisador francês Yann Lecun que se destacam pela eficácia obtida na análise de dados que estejam no formato *grade*. Tal como o nome indica, convolution, significa uma função matemática que deriva da integração de outras duas funções completamente distintas. A arquitetura das CNN deriva das *Artificial Neural Networks* (ANNs). Estes tipos de redes neurais costumam trabalhar sobre imagens e sinais de áudio, mas, no entanto, também existem outras vertentes que aplicam este conceito a dados de texto, séries temporais de dados e sequências de dados.

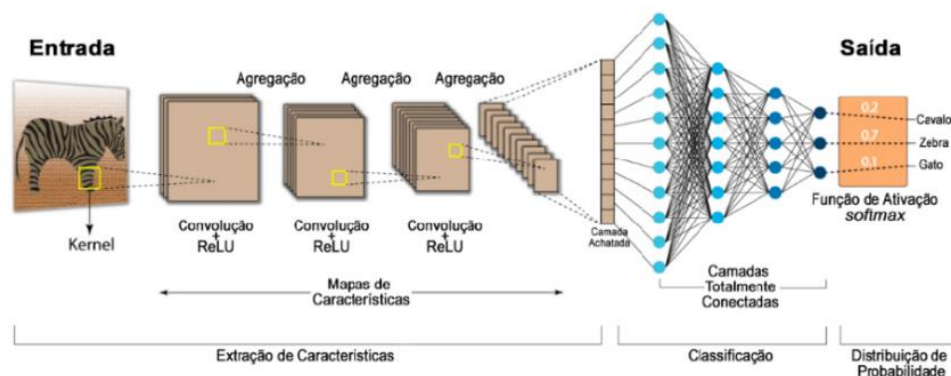


Figura 7 - Estágios e Camadas de uma Rede Neural Convolucional. FONTE: Adaptado de Kategaru (2020)

3.9. Redes Convolucionais Unidimensionais

Redes convulsionais unidimensionais são uma variação das tradicionais que trabalham em duas dimensões. Neste formato, existe uma adaptação para este tipo de rede lidar com dados sequenciais em vez de operar com dados bidimensionais fazendo com que possa processar sequencias unidimensionais tais como texto e sinais de áudio.

3.9.1. Estrutura e Funcionamento

Uma CNN pode ser dividida em três etapas principais: Convolution com função de ativação, Pooling e Fully-connected.

Na primeira camada, Convolution, é processada a operação que realiza a convulsão dos dados de entrada. Irá ser aplicado um *kernel*, que se trata de um filtro, sobre o array de dados

de entrada, que irá calcular o valor da convolução para depois utilizá-lo como o valor de uma célula do array de saída. Durante este processo, o *kernel* é deslocado por uma janela deslizante que foi predefinida pelo array que deu entrada. Os parâmetros utilizados pelos *kernels* são definidos durante o processo de treino exceto o parâmetro que controla a dimensão do *kernel* por se tratar de um hiper parâmetro definido previamente ao processo de treino.

A segunda camada, Pooling, é bastante semelhante à camada convulsional tendo como principal objetivo diminuir o tamanho do espaço que é ocupado pelas variáveis convulsionais. Uma técnica bastante utilizada nesta etapa é o *max-pooling* que consiste em reduzir subpartes dos dados originais pelo maior valor encontrado nessas sub-regiões. Como estamos a trabalhar com sequencias unidimensionais e o pooling precisa duas dimensões, podemos fazer um *reshape* do *input shape* para solucionar este problema.

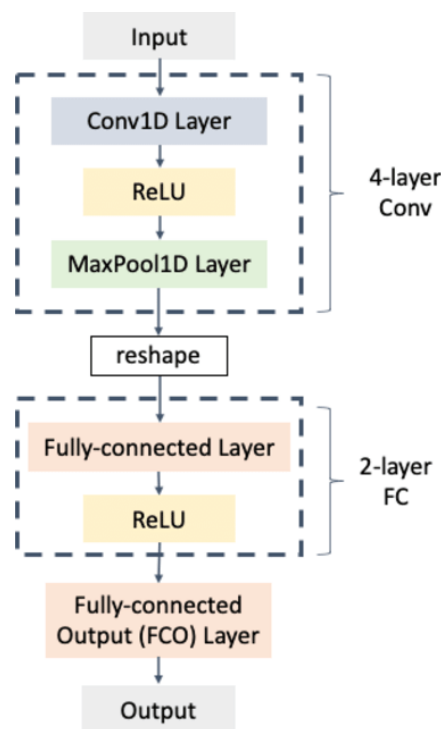


Figura 8 - Diagrama Ilustrativo de uma Implementação CNN 1D usando Max Pooling. FONTE: Predicting the Travel Distance of Patients to Access Healthcare using Deep Neural Networks (2021)

A última etapa da rede é a fully-connected que irá fornecer a probabilidade, no nosso problema, de a condução ser classificada como agressiva ou não agressiva.

3.9.2. Vantagens e Desvantagens

As CNN's, mais concretamente as Conv 1D, são menos onerosas computacionalmente que as LSTM's tradicionais sendo mais rápidas de treinar e executar. Este tipo de arquitetura é extremamente eficaz na extração de padrões importantes em subsequências o que terá influência no resultado obtido. Como as CNN's permitem-nos aplicar camadas de *pooling*, a dimensão dos dados é reduzida o que faz com que a complexidade do modelo também acabe por reduzir evitando assim um potencial *overfitting* da rede. Mas a principal vantagem que se destaca entre as outras é a capacidade desta arquitetura se combinar com outras, como LSTM, para que aprimorar os resultados fazendo com que a escalabilidade seja possível.

Contudo, as Conv 1D também possuem desvantagens que devemos ter em conta na tomada de decisão da escolha do tipo de arquitetura a utilizar. Entre estas desvantagens está a limitação em capturar dependências de longo prazo, pois são projetadas para capturar padrões locais. A tarefa da escolha dos melhores Hiperparâmetros também pode ser um processo complexo e requer tomar uma abordagem tentativa e erro. A eficácia das Conv 1D depende principalmente da estrutura de dados de entrada já que em casos onde os dados não possuam um padrão temporal ou sequências que seja claro, as Conv 1D não irão produzir resultados precisos.

4. Estado de arte

4.1.1. Tecnologias e Ferramentas Atuais

A área de classificação de condução e análise comportamental ao volante tem evoluído significativamente nos últimos anos, impulsionada pelo avanço em tecnologias de inteligência artificial (IA) e aprendizado de máquina (ML). As principais tecnologias e ferramentas atualmente utilizadas neste campo incluem:

- Redes Neurais
- Sensores de de medição inercial
- GPS
- Técnicas de Pré-processamento de dados
- Ambientes de desenvolvimento e bibliotecas de IA
- Big Data
- Computação em Nuvem

4.2. Tecnologias e ferramentas utilizadas

4.2.1. Linguagens

4.2.1.1. Python

Python é uma linguagem de programação de alto nível, reconhecida por sua legibilidade e simplicidade. Tornou-se amplamente utilizada em diversos campos, incluindo inteligência artificial.

4.2.2. Bibliotecas

4.2.2.1. os

‘os’ é uma biblioteca padrão do Python que fornece funcionalidades para interagir com o sistema operativo, como manipulação de arquivos e diretorias. É essencial para operações que envolvem o sistema de arquivos e outros aspetos dependentes sistemas.

4.2.2.2. math

‘math’ é uma biblioteca padrão que oferece funções matemáticas básicas, incluindo operações trigonométricas, logarítmicas e aritméticas. É amplamente utilizada para cálculos matemáticos em scripts Python.

4.2.2.3. NumPy

O NumPy é uma biblioteca fundamental para computação científica em Python. Proporciona suporte para arrays multidimensionais e funções matemáticas de alto nível para operarem nestes arrays, sendo essencial para operações numéricas eficientes.

4.2.2.4. pandas

O Pandas é uma biblioteca essencial para analisar dados, permitindo manipulação e análise de estruturas de dados como DataFrames. Oferece as ferramentas precisas para a leitura, transformação e agregação de dados, facilitando o trabalho com grandes conjuntos de dados.

4.2.2.5. Matplotlib

O Matplotlib é uma biblioteca de visualização de dados que permite a criação de gráficos estáticos, animados e interativos em Python. É amplamente utilizada para gerar visualizações detalhadas e personalizadas de dados.

4.2.2.6. Seaborn

O Seaborn é uma biblioteca baseada no Matplotlib que fornece uma interface de alto nível para a criação de gráficos estatísticos atraentes e informativos. Facilita a visualização de dados complexos de maneira perceptível e estética.

4.2.2.7. TensorFlow

O TensorFlow é uma biblioteca de software de código aberto desenvolvida pelo Google, utilizada para construir e treinar modelos de Machine Learning e Deep Learning. É reconhecida pela flexibilidade e escalabilidade, permitindo a criação de uma variedade de modelos complexos em uma ampla gama de plataformas, desde dispositivos móveis até grandes clusters de servidores.

4.2.2.8. Keras

O Keras é uma biblioteca de código aberto em Python, especialmente dedicada ao Deep Learning. Destaca-se pela sua interface simplificada, que facilita a criação e treino de redes neurais artificiais. É compatível com o TensorFlow e oferece flexibilidade para desenvolver uma variedade de arquiteturas de redes neuronais.

- **keras.models.Sequential, load_model:** Classes e funções para criar e carregar modelos sequenciais.
- **keras.layers:** Inclui várias camadas usadas em redes neurais, como LSTM, Dense, Dropout, Conv1D, Flatten, Bidirectional, BatchNormalization.
- **keras.callbacks:** Contém ferramentas como EarlyStopping, ReduceLROnPlateau e ModelCheckpoint para otimizar e guardar modelos durante o treino.

4.2.2.9. scikit-learn

O Scikit-learn é uma biblioteca para Machine Learning em Python que oferece ferramentas eficientes para a construção e avaliação de dados.

- **metrics:** Inclui várias funções para avaliar a performance de modelos, como `precision_score`, `f1_score`, `recall_score`, `accuracy_score`, `hamming_loss`, `jaccard_score`, `coverage_error`, `label_ranking_loss`.
- **preprocessing:** Fornece métodos para pré-processamento de dados, como `LabelEncoder`.
- **model_selection:** Inclui funções para dividir os dados em conjuntos de treino e teste, como `train_test_split`.

4.2.2.10. Folium

O Folium é uma biblioteca para criar mapas interativos utilizando Leaflet.js, permitindo adicionar marcadores, camadas e plugins de maneira simples.

- **folium.plugins.MarkerCluster, FastMarkerCluster:** Plugins para agrupar marcadores em clusters, melhorando a visualização de mapas com muitos pontos de dados.
- **folium.plugins:** Outros plugins úteis para funcionalidades adicionais em mapas interativos.

4.2.3. Plataformas

4.2.3.1. GitHub

O GitHub é uma plataforma online muito utilizada para controlo de versões de projetos. Foi nesta plataforma que guardamos as várias versões do projeto num repositória que ambos os desenvolvedores tinham acesso. Desta forma conseguíamos ter sempre o código atualizado e sincronizado facilitando assim o processo de desenvolvimento.

5. Trabalhos Relacionados

Análise de projetos e estudos que têm objetivos ou metodologias semelhantes.

5.1. Driving Behavior Classification Based on Sensor Data Fusion Using LSTM Recurrent Neural Networks

Este projeto visa a classificação de comportamentos de condução utilizando uma metodologia baseada na fusão de dados e sensores, mais concretamente utilizando RNN's do tipo LSTM. Esta abordagem tem como finalidade identificar três classes distintas de condução: **normal**, **agressiva** e **sonolenta**.

5.1.1. Metodologia Proposta

O processo de classificação da condução foi tratado como um problema de classificação de series temporais. Para cada instante T de uma viagem, uma janela sequencia S de dados de sensores foi classificada como sendo uma das três categorias de condução.

5.1.2. Arquitetura do Modelo

O modelo que foi proposto denomina-se por Stacked-LSTM e consiste em duas camadas de células de memória, cada uma com 100 neurónios ocultos. Os dados de entradas incluem nove vetores com características vindas dos sensores internos do smartphone usado para os recolher. Os sensores utilizados e as respetivas características captadas foram os seguintes:

- Sensores de Inercia:
 - Aceleração ao longo dos eixos X, Y e Z.
 - Ângulos de rotação e inclinação.
- Sensor GPS:
 - Velocidade do veículo.
- Sensor da Câmera do Smartphone:
 - Distância para o veículo da frente.
 - Número de veículos detetados.

5.1.3. Processo de Treino do Modelo

O processo de treino do modelo foi tratado como um problema de minimização da função de perda “Softmax”. O algoritmo de otimização utilizado foi o “Adam” com uma

taxa de aprendizagem correspondente a 0,0025 juntamente com uma regularização L2 para prevenir overfitting.

5.1.4. Dataset Utilizado

O conjunto de dados utilizado para o treino do modelo foi o “UAH-DriveSet”. Este dataset contém dados de várias viagens que foram capturadas através de um smartphone, que incluem medições dos sensores: acelerómetro, giroscópio, GPS e frames da câmara do smartphone.



Figura 9 - Software Utilizado para Capturar os Dados do UAH-driveset. FONTE: Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks (2017)

5.1.5. Preparação e Pré-processamento de Dados

Os dados retirados do dataset passaram por várias etapas de preparação e pré-processamento antes de seguirem para o modelo. Nesta etapa foi incluída a sincronização dos diferentes tipos de dados. Também foi feita a normalização das medições feitas pelos sensores e a respetiva segmentação das sequências adequadas para o modelo LSTM.

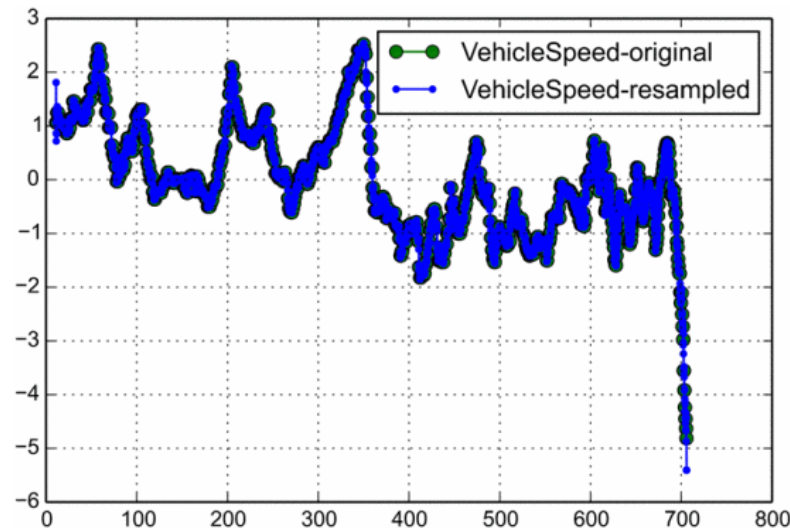


Figura 10 - Gráfico Ilustrativo do Resampling dos Dados. FONTE: Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks (2017)

5.1.6. Resultados Obtidos

Esta implementação obteve resultados bastante concisos visto ter conseguido alcançar com elevada precisão a classificação dos comportamentos de condução. A avaliação foi feita utilizando um conjunto de teste separado, e os resultados foram comparados com abordagens também já existentes, demonstrando assim a eficácia do modelo.

		DriveSafe [6]			proposed Stacked-LSTM		
Road Type	True State	Normal	Aggressive	Drowsy	Normal	Aggressive	Drowsy
Motorway	Normal	7.3	1.5	1.2	8.4	0.9	0.7
	Aggressive	3.5	5.2	1.3	1.1	8.0	0.9
	Drowsy	3.0	1.6	5.4	0.6	0.2	9.2
Secondary	Normal	7.7	1.1	1.2	9.5	0.1	0.4
	Aggressive	0.6	7.3	2.1	0.4	9.6	0.0
	Drowsy	1.9	1.9	6.2	0.7	0.0	9.3

Figura 11 - Resultados da Matriz de Confusão da Implementação Proposta em Comparação com a Classificação de Comportamento de Condução Descrita pelo uah-driveset. FONTE: Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks (2017)

5.1.7. Autores

Khaled Saleh

- Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Australia

Mohammed Hossny

- Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Australia

Saeid Nahavandi

- Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Australia

<https://ieeexplore.ieee.org/document/8317835>

5.2. Driving Behavior Classification Based on Oversampled Signals of Smartphone Embedded Sensors Using an Optimized Stacked-LSTM Neural Networks

5.2.1. Metodologia Proposta

A metodologia proposta neste artigo é baseada na construção de um modelo LSTM otimizado para análise do tipo de condução, selecionando a melhor configuração e parâmetros em cada fase de desenvolvimento. Começa com a formulação do problema de classificação do comportamento do condutor e segue com o pré-processamento dos dados pertencentes ao dataset. Depois, uma arquitetura LSTM é desenvolvida para resolver o classificar a condução em duas tarefas de classificação: três classes (normal, sonolento e agressivo) e binária (agressivo e não agressivo).

5.2.2. Arquitetura do Modelo

O modelo que foi proposto também foi uma Stacked-LSTM e consiste em duas camadas de células de memória, cada uma com 120 neurônios ocultos. As camadas utilizam a função de ativação ReLU e aplicam regularização L2 para evitar overfitting. A última camada do modelo é uma camada Softmax que recebe os vetores de características da segunda camada LSTM e produz uma pontuação de classificação para as três classes de comportamento de direção: normal, sonolento ou agressivo. Ao contrário de trabalhos anteriores que usaram uma janela de 64 vetores com nove características, este modelo usa uma janela de 16 vetores com 13 características.

5.2.3. Processo de Treino do Modelo

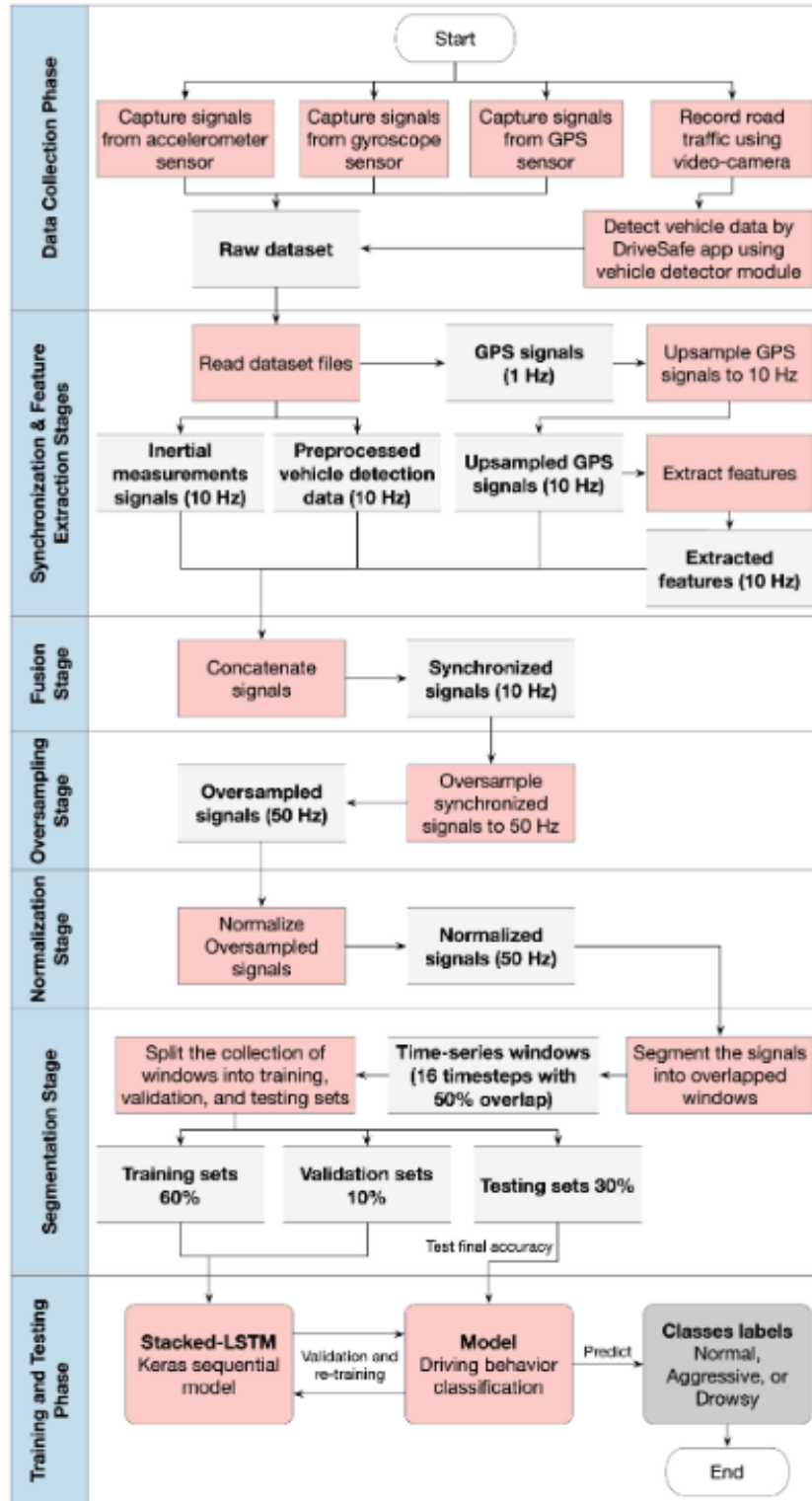
No treino dos modelos, os dados foram divididos em conjuntos de treino e teste. O modelo é treinado usando a função de loss e o otimizador Adam, e os melhores parâmetros são selecionados com base no desempenho do modelos sobre os dados de validação. Este processo também envolveu o uso de técnicas de oversampling para balancear o conjunto de dados e desta forma generalizar o modelo.

5.2.4. Dataset Utilizado

O conjunto de dados utilizado para o treino do modelo também foi o “UAH-DriveSet”.

5.2.5. Preparação e Pré-processamento de Dados

O pre-processamento dos dados envolveu varias etapas para preparar os dados que foram lidos dos sensores antes de estes alimentarem o modelo. Foi feita uma normalização, segmentação e aplicação de técnicas de oversampling para balancear das classes presentes no conjunto de dados. Estas etapas foram essenciais para melhorar a accuracy e a robustez do modelo.



5.2.6. Resultados Obtidos

Os resultados obtidos nesta implementação mostram que o modelo LSTM alcançou uma alta precisão na classificação dos comportamentos de condução. As métricas de desempenho, como precisão, recall e F1-score, indicam que o modelo é extremamente eficaz em distinguir e classificar comportamentos de condução, tanto na versão de classificação de três classes como na classificação binária.

Classification Type	Model	Accuracy	Precision	Recall	F1-score
Three-class classification	3-CCM based on FCPT-3C	99.47%	99.50%	99.48%	99.49%
	SL-LSTM based on FCPT-3C	97.47%	97.59%	97.44%	97.51%
Binary classification	2-CCM based on FCPT-2C	99.30%	99.26%	99.33%	99.34%
	SL-LSTM based on FCPT-2C	98.02%	98.02%	97.75%	97.88%

5.2.7. Autores

Moayed A. Khodairy

- Institute of Computing and Information Technology, King Abdulaziz University, Saudi Arabia
- Specialist in Analysis and Software Development at Saudi Electricity Company, Saudi Arabia

Gibrael Abosamra

- Institute of Computing and Information Technology, King Abdulaziz University, Saudi Arabia

<https://ieeexplore.ieee.org/abstract/document/9312195>

6. Análise

6.1. Análise dos sensores de dispositivos móveis

Os dispositivos moveis modernos, como smarthphone e tablets, estão equipados com uma variedade de sensores que permitem a captura de dados detalhados sobre o ambiente e os movimentos do dispositivo. Esses sensores são usados numa ampla gama de aplicações, desde jogos a navegação até classificação de condução. No contexto do nosso projeto sobre classificação de condução, utilizaremos o acelerómetro, giroscópio e GPS para recolher dados que ajudam a classificar padrões de condução.

6.1.1. Definição e Função

Os sensores de dispositivos móveis são componentes eletrônicos que detetam e respondem a estímulos físicos do ambiente. Eles convertem esses estímulos em sinais elétricos que podem ser processados pelo dispositivo para diversas finalidades. No caso de smartphones, esses sensores são integrados em um único chip, o que permite que sejam compactos e eficientes em termos de energia.

6.1.2. Tipos de Sensores Utilizados

Neste projeto, utilizamos três tipos principais de sensores:

Acelerómetro:

Função: Deteta mudanças na velocidade do dispositivo ao longo dos três eixos (x, y, z).

Aplicações: Útil para detetar movimentos lineares, como acelerações e travagens bruscas.

Giroscópio:

Função: Mede a taxa de rotação do dispositivo em torno dos três eixos (x, y, z).

Aplicações: Detecção de mudanças de direção e rotações, como curvas durante a condução.

GPS (Sistema de Posicionamento Global):

Função: Fornece a localização geográfica precisa do dispositivo, além de informações sobre velocidade e direção.

Aplicações: Monitorar a posição do veículo em tempo real, calcular a velocidade média e traçar rotas.

6.2. Análise dos datasets

Os datasets fornecidos contêm valores de sensores durante uma viagem de carro, do ponto A ao ponto B. Estes valores são utilizados para identificar diferentes tipos de manobras e comportamentos durante a condução.

6.2.1. Estrutura dos Dados

- **Aceleração** (m/s²): Eixos X, Y, Z
- **Giroscópio** (°/s): Eixos X, Y, Z
- **GPS**: Latitude e Longitude

```
accelerometerXAxis: Float - Measures acceleration along the X-axis.  
accelerometerYAxis: Float - Measures acceleration along the Y-axis.  
accelerometerZAxis: Float - Measures acceleration along the Z-axis.  
  
gyroscopeXAxis: Float - Measures angular velocity along the X-axis.  
gyroscopeYAxis: Float - Measures angular velocity along the Y-axis.  
gyroscopeZAxis: Float - Measures angular velocity along the Z-axis.  
  
latitude: Float - Represents geographical latitude coordinates.  
longitude: Float - Represents geographical longitude coordinates.
```

7. Arquiteturas propostas

Para abordar a tarefa de classificação de condução com base nos dados dos sensores de smartphones, propomos o uso de várias arquiteturas de redes neurais LSTM, cada uma com suas vantagens específicas para lidar com dados temporais. As arquiteturas propostas são: Stacked LSTM, Bidirectional LSTM e Convolutional LSTM. A seguir, detalharemos cada uma dessas arquiteturas.

7.1. Stacked LSTM

7.1.1. Introdução ao Stacked LSTM

Uma Stacked LSTM consiste em várias camadas de LSTMs sobrepostas uma sobre a outra. Essa estrutura em camadas permite que o modelo aprenda representações hierárquicas dos dados sequenciais, onde cada camada pode capturar diferentes níveis de abstração temporal.

7.1.2. Estrutura da Stacked LSTM

- 1) Entrada: Dados preprocessados
- 2) Camadas LSTM
 - a. Primeira Camada: Recebe a sequência de entrada e aprende as representações iniciais dos dados temporais.
 - b. Camadas intermédias: Cada camada subsequente recebe a saída da camada anterior, permitindo modelar padrões mais complexos e abstrações de longo prazo.
 - c. Última Camada LSTM: Produz a representação final da sequência temporal, integrando informações de todas as camadas anteriores.
- 3) Camadas Densa (Fully connected Layer): Ligar a saída da última camada LSTM a uma ou mais camadas densas.
- 4) Camada de saída: Camada densa cujo o numero de neurónios corresponde ao numero de classes que o modelo precisa de prever.

7.2. Bidirectional LSTM

7.2.1. Introdução ao Bidirectional LSTM

A Bidirectional LSTM é uma variação das redes LSTM que processa a sequência de dados em duas direções: do passado para o futuro e do futuro para o passado. Essa abordagem permite que o modelo capture melhor as dependências temporais nos dados, utilizando informações contextuais de ambas as direções.

7.2.2. Estrutura da Bidirectional LSTM

- 1) Entrada: Dados preprocessados.
- 2) Camadas Bidirectional LSTM:
 - a. Camada Direcional Frontal: Processa a sequência de entrada na ordem original, aprendendo representações temporais avançadas.
 - b. Camada Direcional Reversa: Processa a sequência na ordem inversa, permitindo que o modelo capture padrões que podem depender de informações futuras.
 - c. Concatenação de Saídas: As saídas das duas direções são concatenadas para formar uma representação mais rica e contextualizada da sequência.
- 3) Camadas Densa (Fully connected Layer): Conecta a saída concatenada a uma ou mais camadas densas.
- 4) Camada de saída: Camada densa cujo número de neurônios corresponde ao número de classes que o modelo precisa prever.

7.3. Convolutional LSTM

7.3.1. Introdução ao Convolutional LSTM

A Convolutional LSTM combina as capacidades das camadas convolucionais e LSTM para lidar com dados sequenciais com características espaciais. Essa arquitetura é especialmente útil quando os dados temporais têm uma estrutura espacial, como séries temporais multivariadas que podem ser vistas como imagens ao longo do tempo.

7.3.2. Estrutura da Convolutional LSTM

- 1) Entrada: Dados preprocessados com estrutura espacial-temporal.
- 2) Camadas Convolucionais (Conv1D):
 - a. Primeira Camada Conv1D: Aplica filtros convolucionais para extrair características locais dos dados temporais.
 - b. Camadas Convolucionais Adicionais: Podem ser adicionadas para capturar características mais complexas e abstrações espaciais.
 - c. Camadas LSTM: Após a extração das características espaciais, as camadas LSTM processam as representações temporais.
 - d. Primeira Camada LSTM: Modela dependências temporais a curto prazo.
 - e. Camadas LSTM Intermédias: Cada camada subsequente captura padrões temporais mais complexos.
 - f. Última Camada LSTM: Produz a representação final das dependências temporais.
- 3) Camada de Saída: Camada densa cujo número de neurônios corresponde ao número de classes que o modelo precisa prever, com ativação sigmoid.

8. Desenvolvimento

Este capítulo aborda o desenvolvimento teórico e prático do projeto, assim como as principais decisões tomadas em função do mesmo.




Como o referido anteriormente, este projeto assenta em duas grandes componentes, a componente de Pesquisa e a componente de Implementação.





8.1. Implementação

A implementação deste projeto passa por várias etapas cruciais que vão desde as descrições até a compilação e treino do modelo. Esta secção irá detalhar cada um dos processos de implementação da solução, fornecendo uma visão abrangente sobre o processamento e a análise dos dados obtidos, além das técnicas de classificação e normalização utilizadas.

Para facilitar a compreensão de cada um dos processos, utilizamos diagramas para descrever a implementação de cada uma das fases. A tabela que se segue tem como propósito ajudar a interpretar os diagramas, proporcionando uma visão organizada dos esquemas de cores utilizados nos mesmos:

Tabela 3 - Glossário de Cores Utilizadas nos Diagramas Ilustrativos

Cor	Significado
Amarelo 	Utilizado para identificar funções importantes.
Azul 	Utilizado para indicar conjuntos de dados de entrada ou saída.
Verde 	Indica estados positivos ou comportamentos classificados como agressivos.
Vermelho 	Indica estados negativos ou comportamentos classificados como não agressivos.

Roxo  Rosa  Azul-Turquesa 	Cores utilizadas para representar diferentes tipos de sensores utilizados na captação dos dados.
Castanho 	Utilizado para identificar a classificação dos dados.
Laranja 	Cor utilizada para identificar threshold.

8.2. Descrição e Caracterização dos Dados

Os dados fornecidos têm por base um dataset em formato CSV que contém várias características relacionadas com a condução. Estas características serão utilizadas para identificar diferentes tipos de manobras e comportamentos durante a condução.

8.2.1. Estrutura dos Dados

- **Tempo:** Timestamp da recolha dos dados
- **Velocidade:** Medida em km/h
- **Aceleração:** Medida em m/s^2
- **Latitude e Longitude:** Coordenadas GPS

8.2.2. Sensores Utilizados

- **Acelerómetros:** Eixos X, Y, Z
- **Giroscópio:** Eixos X, Y, Z

8.3. Tratamento dos Dados

Os dados foram processados para identificar e organizar diferentes manobras de condução como:

- Aceleração e desaceleração súbita
- Curvas e mudanças de faixa

- Paragens e arranques bruscos

8.3.1. Processo de Tratamento de Dados

O processo de Tratamento inicia-se com a captação de dados através de seis sensores, três giroscópios e três acelerómetros, um para cada eixo. Os acelerómetros medem a aceleração ao longo dos eixos X, Y e Z e os giroscópios medem a velocidade angular ao longo desses mesmos eixos.

Com a captação dos dados feita, aplicamos uma função que separa os valores positivos dos negativos para evitar que possíveis valores negativos possam influenciar no desempenho do modelo. Ficamos assim com duas colunas, uma coluna onde só são armazenados os valores positivos e outra em que só são armazenados os valores negativos recolhidos pelos respetivos sensores.

Por fim estas duas colunas são organizadas num array 2D que é composto por 12 elementos no total, 6 elementos positivos e os 6 elementos negativos correspondentes aos dados processados pelos sensores.

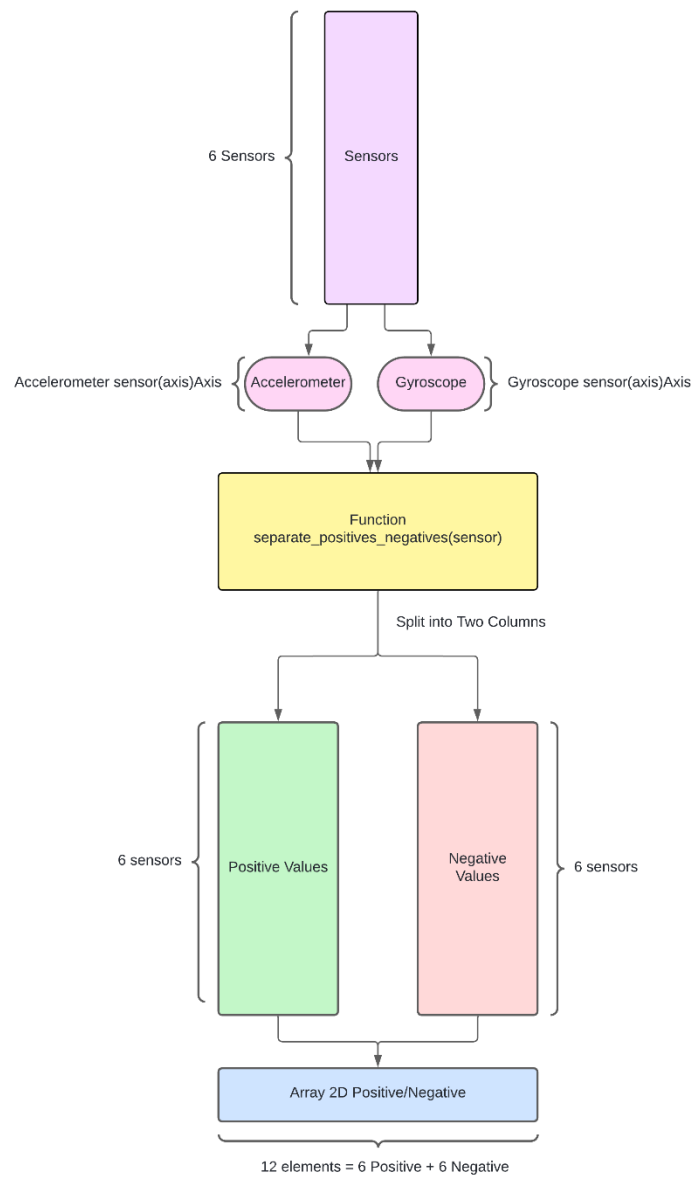


Figura 12 - Diagrama Ilustrativo do Tratamento dos Dados

8.4. Classificação dos Dados

Para efetuar a classificação dos dados foi feita uma função denominada por *y_classification*. Esta função recebe um conjunto de dados e um **threshold** e retorna uma matriz binária que indica se os valores em cada coluna excedem esses **threshold**.

8.4.1. Descrição dos Parâmetros de Entrada

- **Data:** Matriz de dados de entrada, onde cada coluna representa um sensor específico.
- **Threshold:** Valor percentual usado para determinar o limiar acima do qual os dados serão classificados como 1.

8.4.2. Processo de Classificação dos Dados

O processo de classificação dos dados inicia-se com a entrada de dois parâmetros, Data e o threshold, para a função '*y_classification(data, threshold)*'.

A função '*y_classification*' tem como objetivo classificar os dados de forma booleana, ou seja, 1 para agressivo e 0 para não agressivo. Para isso, a função começa por inicializar um vetor que irá servir de output no final do processo de classificação. Após isso irá ser feito um **loop** que percorre todas as **12 colunas** e que irá calcular o valor máximo de cada coluna dos dados já tratados no tópico anterior, utilizando uma função da biblioteca Numpy denominada por '*np.max(data[:, col])*'. Com o valor máximo de cada coluna calculado, efetuamos o cálculo do '**threshold_pos**', sendo este nada mais que o produto do valor máximo de cada coluna pelo valor do threshold fornecido como parâmetro de entrada, ou seja, '**max_value * threshold**'.

Depois de termos calculado o '**threshold_pos**', a classificação dos dados é feita como:

- Se o valor dos dados for maior ou igual ao '**threshold_pos**', o dado irá ser classificados como **1**, ou seja, **agressivo**.
- Se o valor dos dados for menor que '**threshold_pos**', o dado irá ser classificados como **0**, ou seja, **não agressivo**.

Por fim, com os dados já classificados, é retornado o vetor inicializado no início da função contendo esses mesmos dados.

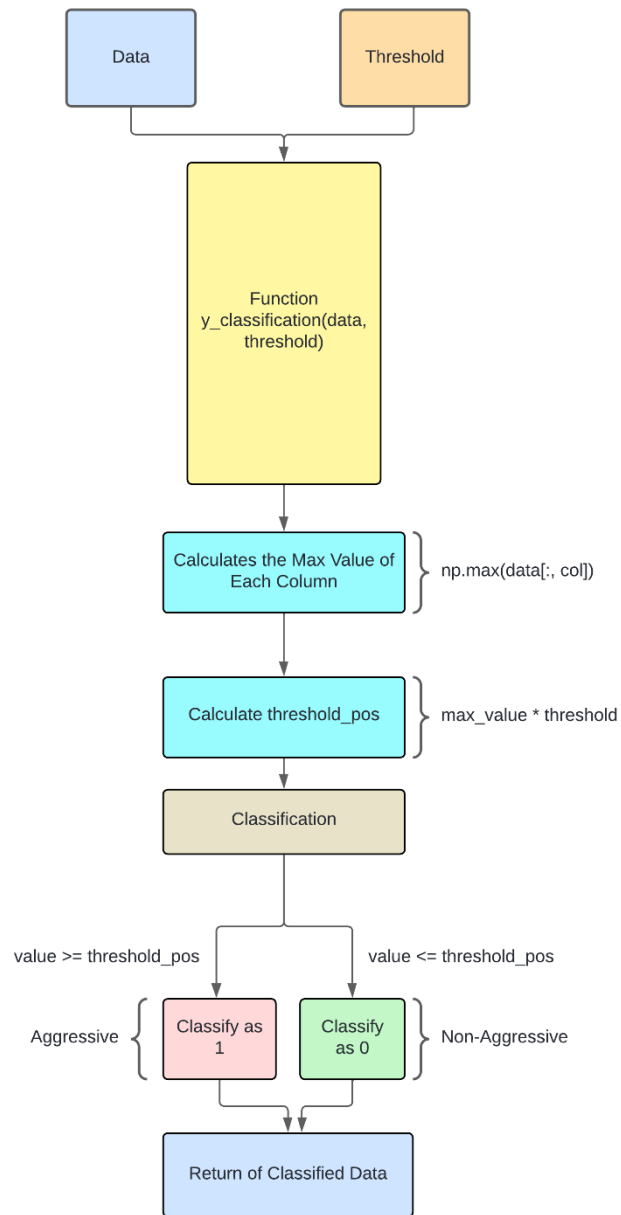


Figura 13 - Diagrama Ilustrativo Da Classificação dos Dados

8.5. Normalização dos Dados

Foi feita uma normalização dos dados que permitiu garantir que os diferentes tipos de dados fossem comparáveis e melhorar o desempenho do modelo. Esta normalização envolveu:

- **Escalonamento:** Ajustar os valores de cada variável para um intervalo comum compreendido entre [0, Max Value de cada Conjunto de Sensores]

8.5.1. Processo de Normalização dos Dados

O processo de normalização de dados começa com a aplicação da função `'max_of_vectors'` que pode ter duas variações:

- Para o acelerômetro, onde os parâmetros de entrada são os: `'turnRightX'`, `'turnLeftX'`, `'accelY'`, `'breakY'`, `'positiveZ'`, `'negativeZ'`;
- Para o giroscópio, em que os parâmetros de entrada são: `'gyrPositiveX'`, `'gyrNegativeX'`, `'gyrPositiveY'`, `'gyrNegativeY'`, `'gyrPositiveZ'` e `'gyrNegativeZ'`

O objetivo desta função é juntar todas as colunas de entrada e combina-las num vetor único calculando depois o valor máximo deste vetor. O resultado final é o retorno do valor máximo do acelerômetro ou do giroscópio que irá ter como papel o parâmetro de entrada da função `'normalize_between_0_and_max_v2(data, max_value)'`.

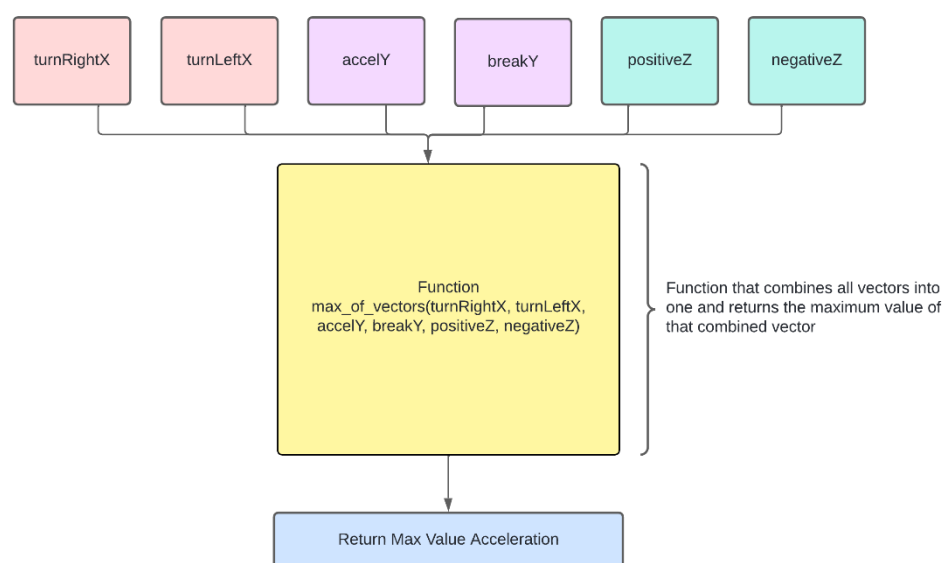


Figura 14 - Diagrama Ilustrativo Da Função Max Of Vectors Aplicada ao Acelerometro

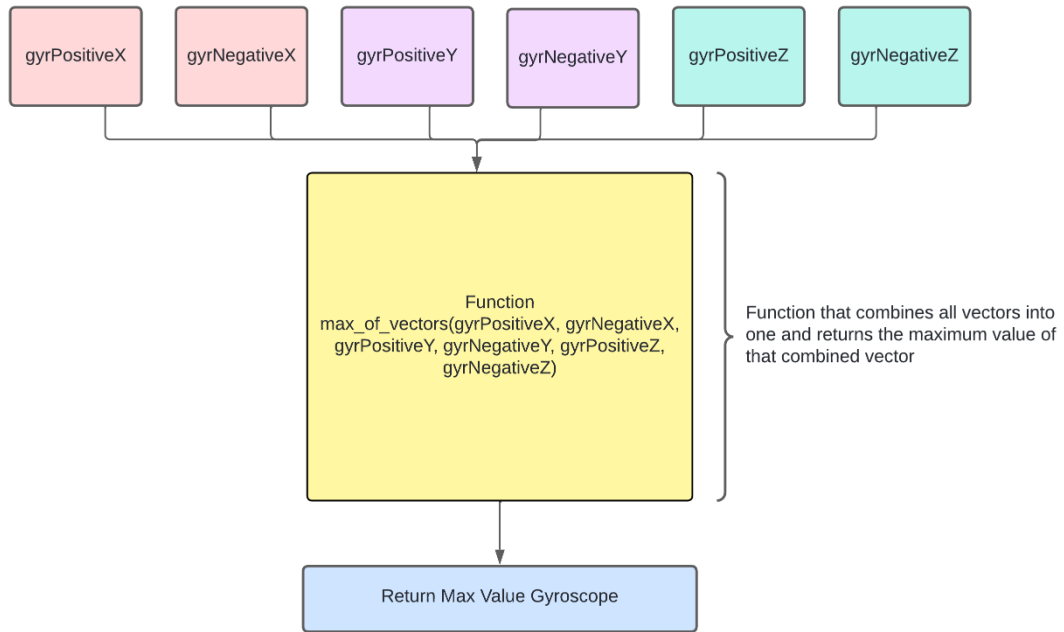


Figura 15 - Diagrama Ilustrativo Da Função Max Of Vectors Aplicada ao Giroscópio

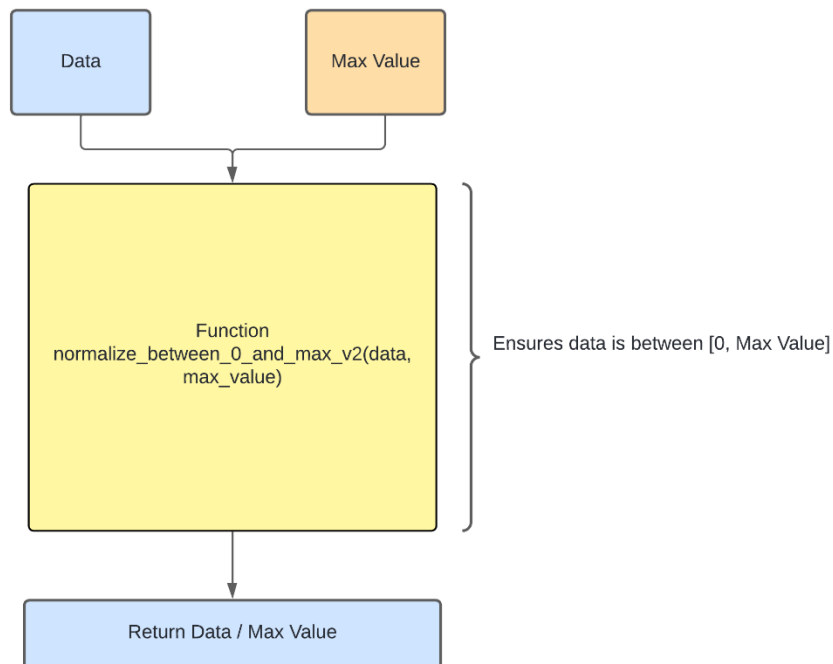


Figura 16 - Diagrama Ilustrativo da Normalização dos Dados

8.6. Representação Visual dos Dados

As posições de GPS das manobras identificadas são guardadas em ficheiros CSV para posterior visualização no Google Maps.

```
def save_manovers_positions_to_csv_file(gps_positions, manovers, filename):
    output = np.zeros_like(gps_positions)
    for i in range(len(manovers)):
        if manovers[i] == 1:
            output[i] = gps_positions[i]
    output = output[~np.all(output == 0, axis=1)]
    np.savetxt(filename, output, delimiter=',', fmt='%.9f')

positions = np.array(list(zip(latitude, longitude)))
save_manovers_positions_to_csv_file(positions, y[:, 2], "acclY.csv")
save_manovers_positions_to_csv_file(positions, y[:, 3], "breakY.csv")
save_manovers_positions_to_csv_file(positions, y[:, 0], "turnRightX.csv")
save_manovers_positions_to_csv_file(positions, y[:, 1], "turnLeftX.csv")
save_manovers_positions_to_csv_file(positions, y[:, 10], "gyrPositZ.csv")
save_manovers_positions_to_csv_file(positions, y[:, 11], "gyrNegZ.csv")
```

8.7. Separação dos Dados em Treino e Teste

Os dados foram divididos em conjuntos de treino e teste para avaliar a performance do modelo. Esta divisão foi feita na seguinte proporção:

- **80%:** Dados para Treino
- **20%:** Dados para Teste

8.7.1. Processo de Separação dos Dados

Para a divisão dos dados em conjuntos para treino e teste, foi utilizada a função ‘`split_train_test(data, test_size=0.2)`’, que tem como parâmetros de entrada a data e o tamanho que a sequencia de dados de teste irá ter. Este tamanho tem influência direta no tamanho da sequencia de treino, pois se o ‘`test_size=0.2`’ significa que **20%** dos dados serão utilizados para o modelo realizar os testes, o que faz com que a sequencia de treino tenha como tamanho **0.8**, ou seja **80%** dos dados.

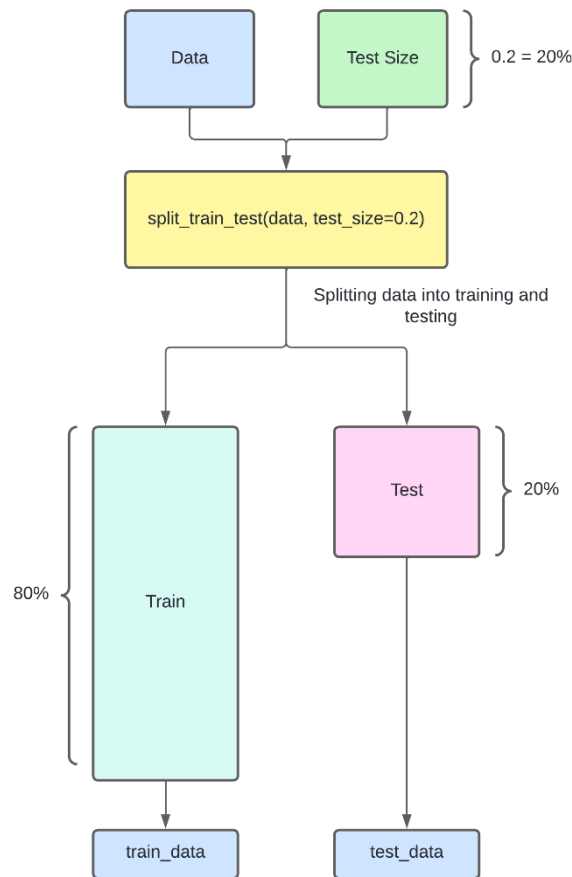


Figura 17 - Diagrama Ilustrativo da Separação dos Dados

8.8.Criação do Modelo

Para a criação do modelo, utilizamos uma abordagem baseada em RNN, mais concretamente a arquitetura LSTM.

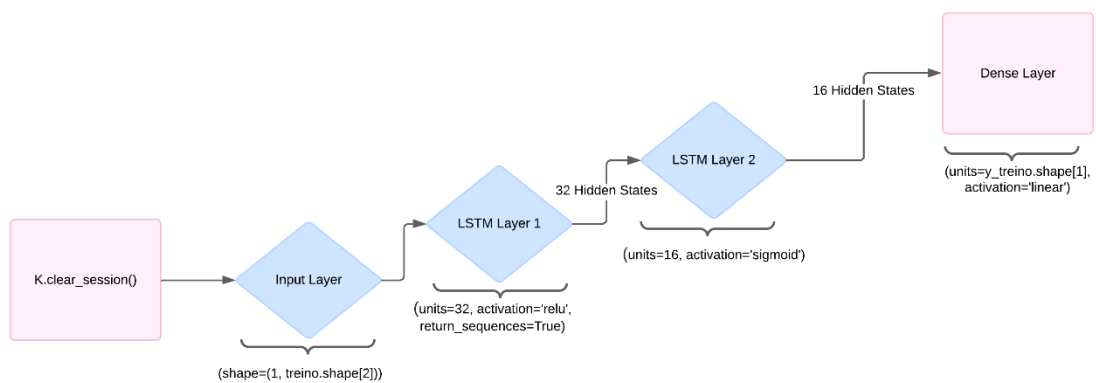


Figura 18 - Diagrama Ilustrativo Da Estrutura do Modelo

8.8.1. Estrutura do Modelo

- **Tipo de Modelo:** Sequencial
- **Primeira Camada:** 32 unidades com a *Relu* como função de ativação
- **Segunda Camada:** 16 unidades com a *Sigmoid* como função de ativação
- **Camada Densa:** Número de unidades igual ao número de classes do conjunto de treino utilizando uma função linear de ativação para prever valores contínuos.

8.9. Compilação e Treino

O modelo é compilado utilizando o otimizador *Adam* e a função de perda *Mean Squared Error*.

```
model_lstm.compile(loss='mean_squared_error', optimizer='adam')
early_stop = EarlyStopping(monitor='loss', patience=5, verbose=1)
```

9. Testes e Resultados

9.1. Objetivos dos Testes

O principal objetivo dos testes é avaliar a precisão e a robustez do modelo LSTM classificador de condução, garantindo que ele possa ser aplicado eficazmente em cenários reais.

9.2. Cenário de Teste

9.3. Testes de implementação

9.4. Testes funcionais

9.5. Testes de desempenho

9.6. Métricas de Avaliação

As métricas utilizadas para avaliar o desempenho dos modelos incluem:

9.6.1. Acurácia

A acurácia é a métrica mais intuitiva e simples para avaliação de modelos de classificação. É definida como a razão entre o número de previsões corretas e o número total de previsões. A fórmula é dada por:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

- TP (True Positives) são os verdadeiros positivos,
- TN (True Negatives) são os verdadeiros negativos,
- FP (False Positives) são os falsos positivos,
- FN (False Negatives) são os falsos negativos.

9.6.2. Precisão

A precisão mede a proporção de verdadeiros positivos entre todas as instâncias que foram classificadas como positivas. É uma métrica importante quando o custo de falsos positivos é alto. A fórmula é dada por:

$$\frac{TP}{TP + FP}$$

- TP (True Positives) são os verdadeiros positivos,
- FN (False Negatives) são os falsos positivos.

9.6.3. Recall (Sensibilidade)

O recall mede a proporção de verdadeiros positivos entre todas as instâncias que são realmente positivas. É uma métrica crucial quando o custo de falsos negativos é alto. A fórmula é dada por:

$$\frac{TP}{TP + FN}$$

- TP (True Positives) são os verdadeiros positivos,
- FN (False Negatives) são os falsos negativos.

9.6.4. F1-Score

O F1 Score é a média harmônica da precisão e do recall, proporcionando um único valor que considera ambas as métricas. É especialmente útil quando há um trade-off entre precisão e recall. A fórmula é dada por:

$$2 \times \frac{Precision \times Recall}{Precision + Recall}$$

9.6.5. Jaccard Score

O Jaccard Score mede a similaridade entre o conjunto de rótulos preditos e o conjunto de rótulos verdadeiros. Em problemas de classificação binária, a fórmula é dada por:

$$\frac{TP}{TP + FP + FN}$$

- TP (True Positives) são os verdadeiros positivos,
- FP (False Positives) são os falsos positivos,
- FN (False Negatives) são os falsos negativos.

9.6.6. Hamming Loss

A Hamming Loss mede a taxa de predições incorretas, onde cada predição incorreta conta igualmente. Para classificação binária, a fórmula é dada por:

$$\frac{1}{n} \sum_{i=1}^n 1(\hat{y}_i \neq y_i)$$

- N é o número total de amostras.
- \hat{y}_i é o rótulo previsto para a i -ésima amostra.
- y_i é o rótulo verdadeiro para a i -ésima amostra.
- $1(\hat{y}_i \neq y_i)$ é uma função indicadora que vale 1 se $\hat{y}_i \neq y_i$ e 0 caso contrário.

9.7. Resultados dos Testes

Metrics	<i>Proposed</i> ConvLSTM	BiLSTM	Stacked LSTM
Accuracy	97.71%	96.06%	96.38%
Precision	95.40%	84.66%	95.00%
F1 Score	93.02%	87.56%	85.65%
Recall	93.96%	81.66%	89.29%
Hamming Loss	0.20%	0.33%	0.32%
Jaccard Score	88.95%	72.69%	81.95%

9.8. Análise de resultados

Nesta seção, discutimos os resultados obtidos e comparamos o desempenho das diferentes arquiteturas de redes neurais. Observamos que:

9.9.Considerações Finais

10. Conclusões

Inserir aqui as conclusões ou conclusão. Trata-se de um elemento **obrigatório**.

A conclusão:

- Deve ser sucinta;
- Não deve conter informações ou ideias novas;
- Deve permitir concluir se se atingiram os objetivos enunciados na introdução.

Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão. Texto da conclusão.

11. Trabalho Futuro

Bibliografia ou Referências Bibliográficas

Inserir aqui a bibliografia ou referências bibliográficas. Trata-se de um elemento **obrigatório**.

Notas: o sistema a adotar para a apresentação das referências bibliográficas e as suas citações deve:

- Respeitar uma norma estabelecida;
- Seguir as práticas mais disseminadas na área em causa;
- Ser empregue de modo uniforme em todo o documento.

Bibliografia – quando se coloca toda a bibliografia consultada;

Referências bibliográficas – quando se faz referência apenas à bibliografia citada.

<https://repositorio.ufpb.br/jspui/bitstream/123456789/15606/1/DAR20052019.pdf>

Anexos

Elemento a figurar, **quando aplicável**.

Glossário

Elemento a figurar, **quando aplicável**.