CELSO DE ÁVILA RAMOS

SISTEMA NEURAL ANTIFURTO VEICULAR

LAVRAS – MG

2016

CELSO DE ÁVILA RAMOS

SISTEMA NEURAL ANTIFURTO VEICULAR

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes e Sistemas Embarcados, para a obtenção do título de Mestre.

Orientador

Dr. Wilian Soares Lacerda

LAVRAS – MG

2016

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Ramos, Celso de Ávila.
Sistema Neural Antifurto Veicular / Celso de Ávila Ramos. – Lavras : UFLA, 2016.
81 p. : il.

Dissertação (mestrado acadêmico)–Universidade Federal de Lavras, 2016.
Orientador(a): Wilian Soares Lacerda.
Bibliografia.

1. Redes Neurais Artificiais. 2. Antifurto Veicular. 3. Classificação de Condutores. 4. Segurança Veicular. I. Universidade Federal de Lavras. II. Título.

CELSO DE ÁVILA RAMOS

SISTEMA NEURAL ANTIFURTO VEICULAR

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes e Sistemas Embarcados, para a obtenção do título de Mestre.

APROVADA em 20 de abril de 2016.

Dr. Cristiano Leite de Castro	UFMG
Dr. Danton Ferreira	UFLA

Dr. Wilian Soares Lacerda

Orientador

LAVRAS – MG

2016

À minha esposa Liliana, como demonstração de minha entrega não apenas aos meus objetivos pessoais, mas também ao nosso futuro como família.

DEDICO.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus pela luz que me guiou por todo o trajeto, à Nossa Senhora, pela interseção que nunca me permitiu sequer pensar em sucumbir.

Agradeço a meus pais Luzia e Domingos que, à sua maneira, sempre me prestaram apoio incondicional nesta trajetória.

Agradeço à minha esposa Liliana pelo incentivo, pela compreensão, pelo amor e pelo privilégio de poder ser seu marido nas horas alegres ou difíceis a que esta trajetória me expôs.

Agradeço ao meu orientador, Dr. Wilian Soares Lacerda, que sempre esteve à disposição para me auxiliar e me nortear no caminho correto em direção aos objetivos deste trabalho.

Agradeço ao professor Luiz Henrique Andrade Correia, pelos ensinamentos e disponibilidade.

Agradeço ao amigo Wesley Natanael Gallo pela colaboração, pelo incentivo nos estudos, pelas palavras de amizade e apoio em todos os momentos e, claro, pelos cafés.

Agradeço aos amigos Heitor Scalco Neto e Ariel Marques que me ensinaram o valor de uma amizade colaborativa e sincera.

À Universidade Federal de Lavras e ao Programa de Pós-Graduação em Ciência da Computação. A todos, serei eternamente grato.

RESUMO

Atualmente, a preocupação com a segurança de bens tem sido uma constante na população, principalmente em países onde os índices de furtos são elevados. Diante de um cenário preocupante, questões sobre como desenvolver tecnologias e soluções que consigam reduzir os índices de furtos devem ser abordadas, buscando aprimorar técnicas existentes e/ou elaborar novas. Este trabalho tem por objetivo, verificar a viabilidade do uso de redes neurais artificiais para a detecção de condução desautorizada de veículos e implementar um sistema automático em tempo real, baseado em uma rede neural artificial treinada para a classificação do condutor, mediante sua forma de conduzir o veículo, a partir de dados obtidos do próprio automóvel. Para tanto, foi utilizado o dispositivo OBD-II, comumente empregado para obter dados dos sensores do veículo. As variáveis de posição do acelerador, aceleração em x, aceleração em y e aceleração em z foram utilizadas como entradas de uma rede neural para a classificação do condutor como sendo autorizado ou não autorizado a conduzir o veículo. Foi desenvolvido um aplicativo Android que envia os dados obtidos do OBD-II para um Web Service Python. Este Web Service possui uma função de verificação que utiliza uma rede neural treinada para classificar o condutor e retornar uma resposta para o usuário. O treinamento utilizou algoritmo *backpropagation*, obtendo resultados satisfatórios durante os testes, com 88% de acertos da rede neural treinada. O índice de eficiência dos testes foi medido por meio do coeficiente Kappa, apresentando um resultado considerado excelente para este índice. O Sistema Neural Antifurto Veicular é uma ferramenta que pode auxiliar proprietários a monitorar a condução de seu automóvel. Espera-se, também, que o sistema possa auxiliar outras áreas de interesse como autoridades e empresas de seguro. O uso de Redes Neurais Artificiais para a classificação do condutor mostrou-se viável e eficaz para este fim. Também é importante ressaltar que o dispositivo OBD-II pode ser empregado para outras finalidades que vão além do diagnóstico dos componentes do veículo para sua correta manutenção. O sistema desenvolvido comprovou que é possível avaliar o comportamento do motorista por meio de dados fornecidos pelo próprio veículo que este conduz.

Palavras-chave: Redes Neurais Artificiais. Antifurto Veicular. Classificação de Condutores. Segurança Veicular.

ABSTRACT

Currently, the concern for the safety of properties has constantly been among the population, especially in countries where the theft rates are high. Faced with a worrying scenario, issues about developing technologies and solutions that are able to reduce theft rates must be addressed, seeking to improve existing techniques and / or develop new ones. This study aims to verify the viability of using artificial neural networks for the detection of unauthorized driving of vehicles and implement an automated real-time system, based on an artificial neural network trained to classify the driver as to how they drive the vehicle, based on data obtained from the automobile itself. Therefore, we used the OBD-II device commonly used to obtain data from vehicle sensors. Variables like throttle position, acceleration in x, acceleration in y and acceleration in z were used as inputs to a neural network to classify the driver either as authorized or not authorized to drive the vehicle. An Android app that sends data from the OBD-II to a Web Service Python was developed. This Web Service has a scan function that uses a neural network trained to classify the driver and return an answer to the user. The training algorithm used was backpropagation, obtaining satisfactory results during the tests, with 88% of the trained neural network hits. The test of the efficiency ratio was measured by the Kappa coefficient, with a result as excellent for this index. The Neural Vehicle Anti-Theft System is a tool that can help owners monitor the driving of their car. It is hoped, too, that the system can help other areas of interest, as authorities and insurance companies. The use of Artificial Neural Networks to classify the driver was proved to be feasible and effective for this purpose. It is also important to note that the OBD-II device can be used for other purposes that go beyond the diagnosis of vehicle components for its proper maintenance. The developed system proved that it is possible to assess the behavior of the driver by means of data supplied by the vehicle they conduct.

Keywords: Artificial Neural Networks. Vehicular Anti-theft. Conductors Classification. Vehicular Safety.

LISTA DE FIGURAS

LISTA DE QUADROS E TABELAS

LISTA DE ABREVIATURAS

ABS – *Anti-braking System*

API – *Application Programming Interface*

DENATRAN – Departamento Nacional de Trânsito

EQM – Erro médio quadrático

GPS – *Global Position System*

HTTP – *Hiperttext Transfer Protocol*

ISO – *International Organization for Standardization*

OBD-II – *On Board Diagnostic* versão II

PMC – *Perceptron* multicamadas

RNA – *Rede Neural Artificial*

ROC – *Receiver Operator Characteristic*

ROM – *Ready Only Memory*

SAE – *Society of Automotive Engeneers*

SDK – *Software Development Kit*

SMS – *Short Message Service*

SOAP – *Simple Object Access Protocol*

UDP – *User Datagram Protocol*

TA – *Termo de aprendizagem*

TM – *Termo momentum*

XML – *Extensible Markup Language*

W3C – *World Wide Web Consortium*

WSDL – *Web Service Definition Language*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, a preocupação com a segurança de bens tem sido uma constante na população, principalmente em países onde os índices de furtos, em diversas dimensões, são elevados. No ano de 2013, somente na cidade de São Paulo, onde a frota de veículos é uma das maiores do país, 99.206 veículos foram roubados ou furtados (D'AGOSTINO; REIS; MACEDO, 2014). Diante de um cenário preocupante, questões sobre como desenvolver tecnologias e soluções que consigam reduzir os índices de furtos devem ser abordadas, buscando aprimorar técnicas existentes e/ou elaborar novas.

A maioria dos sistemas antifurto dos veículos atuais, como travas e alarmes restringem-se a dispositivos físicos instalados no automóvel sem qualquer conectividade com o proprietário. Estas soluções não alertam os proprietários em tempo real sobre a ocorrência de um furto ou não provêm mecanismos para que uma solução imediata seja aplicada. Tais problemas podem ser passiveis de solução, aplicando-se inteligência computacional aos dispositivos cujos fins são promover a segurança e assegurar a legitimidade da pessoa que conduz o veículo. Algumas aplicações vêm sendo desenvolvidas com o intuito de melhorar estas condições e propiciar maior segurança, em termos de perdas patrimoniais, para os proprietários de veículos automotivos, como é o caso do sistema OnStar da Chevrolet[1].

Também percebe-se, atualmente, que analisar o comportamento do motorista ao dirigir é essencial na prevenção de acidentes que causem danos físicos e/ou materiais a pessoas e instituições. O comportamento ao dirigir pode fornecer informações relevantes sobre a autenticidade de um condutor, ou seja,

[1] http://www.chevrolet.com.br/onstar.html

observar se um motorista conduz o veículo de forma peculiar ao seu estilo de condução. Observações fora de um determinado padrão, também podem caracterizar um possível furto. O fator comportamento do motorista é um dos objetos deste trabalho.

1.1 Justificativa

Existem no mercado, diversos produtos de monitoração e segurança veicular com as mais diversas funcionalidades que vão desde o fornecimento da localização do veículo, identificação de legalidade quanto a ser ou não um carro roubado por meio de consulta à base nacional do DENATRAN (INFOSEG, 2014), até a interrupção de ignição através de acesso remoto por meio de mensagens SMS (*Short Message Service*). Dada a importância do assunto e busca por soluções que intensifiquem a proteção dos proprietários de veículos, pesquisas que envolvem este tema estão se tornando cada vez mais tônicas.

Desta forma, propor soluções eficientes que auxiliem tanto a população quanto as autoridades no combate ao furto de veículos é uma necessidade vigente, pois ao mesmo tempo em que inibe o criminoso, também tranquiliza o proprietário quanto à segurança de seu patrimônio.

1.2 Objetivos

Este trabalho tem por objetivo geral verificar a viabilidade do uso de redes neurais artificiais para a detecção de condução desautorizada de veículos. Uma vez constatada esta viabilidade, pretende-se, como objetivo específico, implementar um sistema automático em tempo real baseado em uma rede neural artificial treinada para a classificação do condutor, mediante sua forma de conduzir o veículo, a partir de dados obtidos do próprio automóvel.

Espera-se que, com os resultados desta pesquisa, seja possível estabelecer um padrão de dirigibilidade que identifique de forma inteligente, a legitimidade do condutor veicular.

1.3 Organização do texto

O texto deste trabalho está organizado da seguinte forma:

No capítulo 1 é apresentada uma introdução, abordando uma visão geral sobre o tema, os problemas inerentes a ele, bem como a justificativa para a sua realização e seus objetivos.

O capítulo 2 apresenta uma revisão bibliográfica, dissertando sobre as teorias acerca dos temas relevantes para o desenvolvimento deste trabalho, bem como os sistemas e tecnologias empregados atualmente na segurança veicular.

O capítulo 3 apresenta uma descrição das técnicas e metodologias empregadas para a pesquisa e desenvolvimento do protocolo.

O capítulo 4 apresenta os resultados obtidos, discutindo-os em detalhes.

O capítulo 5 apresenta uma conclusão sobre o estudo e o sistema desenvolvido, apontando suas vantagens e limitações, bem como propondo alternativas para estudos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo será apresentada uma revisão de literatura sobre os principais tópicos relacionados ao desenvolvimento deste trabalho, abordando definições, especificações das tecnologias empregadas e a relevância de cada um no contexto atual do tema.

2.1 O comportamento do motorista

Obter e analisar dados relacionados ao comportamento do motorista ao dirigir tem sido o foco de vários estudos, seja para realizar simulações, prevenir acidentes ou compreender, sob o aspecto psicológico, as ações tomadas por condutores de veículos. Identificar comportamentos que possam oferecer risco ao condutor, pedestres e fatores de trânsito em geral é o passo inicial para apresentação de soluções para diversos problemas relacionados à condução de automotivos.

Newman et al (2011) afirmam que a direção ocupacional, executada por trabalhadores de transporte de cargas, polícia e serviços de emergência, é responsável por grande parte dos acidentes no trabalho. Os autores propuseram um estudo com o objetivo de apontar os tipos de comportamentos inseguros ao dirigir no contexto ocupacional, baseando-se em estatísticas internacionais que afirmam que a maioria dos acidentes com mortes no trabalho são causados pelo comportamento inseguro do motorista.

Meseguer et al (2013) desenvolveram uma pesquisa para avaliar o comportamento do motorista em sistemas de *eco-driving*, com ênfase na

identificação dos principais fatores que afetam o consumo de energia, tais como: economia de combustível, mudanças bruscas na aceleração e alta velocidade. Os autores aplicaram técnicas de mineração de dados e redes neurais para gerar uma classificação dos estilos de direção de usuários baseada em seus traços de mobilidade. Para recolher os dados do veículo, foi empregada uma interface OBD-II *bluetooth*. Os autores concluíram que a classificação do estilo de dirigir do motorista por uma rede neural apresenta alta correlação com os comportamentos observados em ambiente real.

Chong et al (2013) propuseram um modelo em rede neural baseado em regras para simular o comportamento do condutor em termos de ações longitudinais e laterais associadas a condições de tráfego que levassem a situações críticas de segurança e, então ser possível fazer comparações com dados naturalistas.

Considerou-se como relevante para esta dissertação, o trabalho desenvolvido por Shi-Huang Chen, Jeng-Shyang Pan e Kaixuan Lu (2015), que propuseram um modelo de análise de comportamento de condutores baseado em informações provenientes de um dispositivo denominado OBD-II (*On Board Diagnostic*) e algoritmos Adaboost que coletam informações do veículo como velocidade, rotação do motor, posição do acelerador. Este método faz uso de algoritmos Adaboost que criam um modelo de classificação do comportamento do condutor veicular, podendo determinar se o comportamento do motorista atual pertence ou não a uma categoria segura. Os autores consideram que o método empregado demonstra com exatidão o comportamento do motorista e alcançou uma taxa de precisão de 99,8% em várias simulações de condução. Consideram, também que o método proposto tem potencial de aplicação no mundo real em sistemas de assistência ao condutor.

As pesquisas citadas neste tópico refletem a preocupação em obter e analisar o comportamento do motorista por meio de diversas técnicas, para atingir diferentes objetivos. Percebe-se que modelos que empregam redes neurais artificiais propiciam uma compreensão satisfatória deste comportamento e oferecem informações significativas para a tomada de decisões de diversos aspectos, principalmente no que diz respeito à segurança do condutor, pois são capazes de analisar e processar uma quantidade considerável de dados em um tempo adequado de resposta.

2.2 Sistemas embarcados

Kamal (2008) define um sistema embarcado como sendo um sistema que tem um software embutido e um hardware que o torna dedicado a uma aplicação, ou parte específica de uma aplicação ou produto, ou parte de um sistema maior.

Um sistema embarcado deve receber entradas por meio de sensores, converter os dados analógicos em digitais, processá-los e fornecer uma saída. Nos dias atuais, os sistemas embarcados estão por toda parte, nos lares, escritórios, hospitais, veículos etc.

Um sistema embarcado inclui três componentes principais:

1. Um hardware, similar a um computador. Compreende um software inserido em memória ROM (*Ready Only Memory*) ou flash. Não necessita de um disco secundário ou CD como um computador.
2. Um software de aplicação principal que executa uma tarefa específica ou até mesmo vários processos ou threads.

3. Um sistema operacional de tempo real que supervisiona a execução do software de aplicação e organiza os acessos aos recursos de acordo com as prioridades das tarefas no sistema (KAMAL, 2008).

A Figura 1 mostra os componentes de hardware de um sistema embarcado.

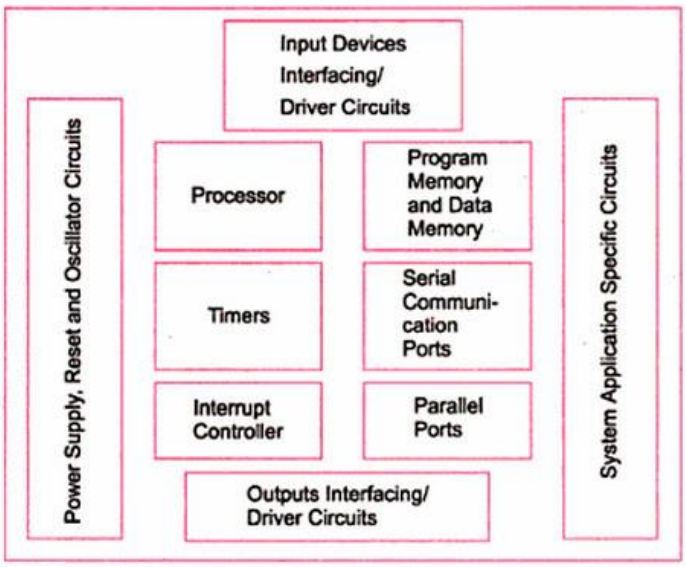

Figura 1 Componentes de hardware de um sistema embarcado
Fonte (KAMAL, 2008)

De acordo com Marwedel (2011), algumas áreas de aplicação de sistemas embarcados são:

- Eletrônica automotiva: sistemas de controle de *air bag*; sistemas de controle de motor; sistemas de frenagem ABS (*anti-braking system*); programas de estabilidade eletrônica e outras características de segurança; ar condicionado; sistemas de GPS (*Global Position System*); proteção antifurto e muito mais.

- Aviação: sistemas de controle de voo; sistemas anticolisão; sistemas de informação ao piloto e outros.
- Ferrovias: para ferrovias, os sistemas são semelhantes ao apresentados para carros e aviões, com destaque para as características de segurança.
- Telecomunicações: o uso de telefones celulares tem crescido significativamente nos últimos anos. Para telefonia móvel, os sistemas embarcados estão presentes na utilização de rádio frequência e processamento de sinal digital.
- Área de saúde: a importância de produtos relacionados ao cuidado com a saúde tem aumentado. Há um grande potencial para a melhoria de serviços médicos. Há diversas técnicas que podem ser aplicadas nesta área. Vieira (2013), por exemplo, desenvolveu um sistema embarcado de dispensação de medicamentos a pacientes idosos hipertensos, na cidade de Ribeirão Preto, São Paulo, com grande aceitação pelos pacientes, devido à facilidade de interação e baixo custo do produto.
- Segurança: sistemas embarcados podem se empregados para promover segurança em diversos aspectos, incluindo autenticação e identificação de pessoas, através de leitura de impressão digital ou reconhecimento de face.
- Eletrônicos de consumo: controles de TV de alta definição; telefones multimídia; console de jogos, entre outros.
- Construções inteligentes: o processamento de informações pode ser usado para aumentar o conforto nos edifícios, diminuindo o consumo de energia, aumentando a segurança. Existe a possibilidade de integração entre sistemas de ar condicionado, iluminação e distribuição de informação em um único sistema.
- Logística: sistemas embarcados podem ser empregados em sistemas que fazem uso de rádio frequência para identificação de objetos em uma área ou redução do consumo de energia.

- Robótica: a maioria das características descritas anteriormente, também se aplica à robótica. Atualmente, alguns novos tipos de robôs vêm sendo modelados com funcionalidades particulares controladas por sistemas embarcados.
- Aplicações militares: processamento de informações, controles de mísseis etc.

O processamento de dados em um sistema embarcado é realizado por um microcontrolador que tem funcionalidades semelhantes a um microprocessador, porém com características mais simples, como por exemplo o número de instruções. Existem muitos tipos de microcontroladores no mercado que podem ser aplicados em vários tipos de sistema, de acordo com a necessidade, assim como plataformas para a sua utilização.

Um projeto de sistema embarcado possui restrições com relação ao desempenho, energia, tamanho do projeto e custos de fabricação (KAMAL, 2008). Por isso, o projetista deve levar em consideração as características do hardware sobre o qual está desenvolvendo um projeto, buscando minimizar o código do software, bem como delimitar com exatidão a funcionalidade do sistema.

2.3 O sistema OBD-II

A obtenção de informações para diagnóstico de um veículo pode ser conseguida por meio de um sistema denominado *On Board Diagnostic*, ou simplesmente OBD-II. Este dispositivo é incorporado ao computador de bordo de veículos mais modernos com o objetivo de monitorar diversos componentes e sistemas.

Atualmente, emprega-se a segunda geração de OBDs, o OBD-II, utilizado desde 1996, que é uma evolução do padrão OBD I. Este sistema é capaz de monitorar os componentes que afetam o desempenho de um veículo, emitindo avisos ao condutor que o auxiliam a tomar providências cabíveis para cada situação monitorada, como por exemplo, o momento em que se faz necessária uma revisão de rotina no automóvel (GODAVARTY; BROYLES; PARTEN, 2000).

A padronização de conectores de OBD II compreende uma interface de 16 pinos (2x8), de acordo com a norma SAE (Society of Automotive Engeneers) J1962 ou ISO 15031-3. O conector OBD II deve estar localizado até 0,61m do volante, ao alcance do motorista (BASTOS, 2012). A Figura 2 mostra um exemplo de conector OBD II e o Quadro 1 mostra a identificação de pinos do conector OBD II.

Figura 2 Conector macho OBD-II
Fonte (CARPLUGS, 2014)

Quadro 1 Identificação de pinos do conector OBD-II

Contato	**Alocação**
1	Reservado ao fabricante
2	Sinal positivo do SAE J1850 PWM e VPW
3	Reservado ao fabricante
4	Taxa de carroceria
5	Terra do sinal
6	Rede CAN-High da ISO 15765-4 e SAE J2234
7	Rede K Line da ISO 9141-2 e ISSO 14230-4
8	Reservado ao fabricante
9	Reservado ao fabricante
10	Sinal negativo do SAE J1850 PWM
11	Reservado ao fabricante
12	Reservado ao fabricante
13	Reservado ao fabricante
14	Rede CAN-Low da ISO 15765-4 e SAE J2284
15	Rede L Line da ISO 9141-2 e ISSO 14230-4
16	Voltagem de bateria

Fonte: SAE J1962 (2002)

Todos os automóveis com OBD-II adotam um código de diagnóstico de problemas padronizado e a interface de conexão ISO J1962 (SAE, 2002). Vários sistemas de controle internos do carro recebem a entrada de informações sobre o estado do veículo a partir de sensores instalados em várias partes do mesmo. Tais informações podem ser obtidas, externamente, via conector OBD-II (BAEK et al, 2011).

O OBD-II é um sistema fundamental para a obtenção de informações do comportamento de um veículo, pois fornece dados preciosos para as mais diversas necessidades que vão desde a simples verificação da necessidade de revisão até uma análise detalhada para a descoberta e solução de defeitos que possam ocorrer. Novas possibilidades de utilização destas informações devem ser consideradas e estudadas para tirar um proveito ainda maior de tudo o que OBD-II oferece.

Conforme afirmam Godavarty, Broyles e Parten (2000), as informações obtidas do computador de bordo dos veículos modernos, por meio do OBD-II, ajudam a melhorar a compreensão da relação entre os hábitos de direção e o desempenho do veículo. As informações captadas podem ser baixadas e transmitidas por um serviço de telefonia celular para ajudar em reparos remotos, por exemplo.

2.4 Redes neurais artificiais

"As Redes Neurais Artificiais (RNAs) são modelos matemáticos que se assemelham às estruturas neurais biológicas e que têm capacidade computacional adquirida por meio de aprendizado e generalização" (BRAGA; CARVALHO; LUDEMIR, 2014). O aprendizado das redes neurais está

relacionado com a sua capacidade de adaptarem seus parâmetros de acordo com a interação com o meio externo.

As respostas fornecidas por uma rede neural são provenientes de treinamentos que representam uma análise sucessiva de parâmetros a fim de se encontrar um padrão que seja condizente com o comportamento do objeto do meio externo analisado. A generalização de uma RNA se dá quando esta é capaz de fornecer respostas coerentes até mesmo para dados não fornecidos a ela durante o processo de treinamento (REZENDE, 2005).

Conforme salienta Rezende (2005), o processamento de informação de RNAs é feito por meio de estruturas neurais artificiais, sendo o armazenamento e processamento das informações feito de forma paralela e distribuída por mecanismos processadores relativamente simples, denominados neurônios artificiais.

A utilização de redes neurais artificiais se deu a partir do ano de 1943. Neste ano, os pesquisadores McCulloch e Pitts delinearam o primeiro modelo formal de um simples neurônio computacional. A partir de então, muitas implementações de “computadores neurais” foram testadas, no entanto os mecanismos para a aprendizagem dos sistemas nesta época eram muito fracos para suportar problemas computacionais complexos. Somente em meados da década de 1980 é que as redes neurais artificiais começaram a dar resultados significativos (HUANG; ZHANG, 2007).

A Figura 3 mostra uma visão esquemática do neurônio artificial de McCulloch e Pitts. Nela, as entradas do neurônio correspondem a um vetor de entrada x de dimensão n. Para cada uma das entradas do vetor x há um peso correspondente w_i que são valores utilizados para ponderar cada uma das variáveis de entrada na rede, permitindo-se quantificar as suas relevâncias em

relação à funcionalidade do respectivo neurônio. O combinador linear ∑ tem por função agregar todos os sinais de entrada que foram ponderados pelos respectivos pesos sinápticos a fim de produzir um valor de potencial de ativação. O limiar de ativação Θ é uma variável que especifica qual será o patamar apropriado para que o resultado produzido pelo combinador linear possa gerar um valor de disparo em direção à saída do neurônio. O potencial de ativação (u) é o resultado produzido pela diferença do valor produzido entre o combinador linear e o limiar de ativação. Se o valor é positivo, o neurônio produz um potencial excitatório, caso contrário, o potencial será inibitório. A função de ativação g tem como objetivo limitar a saída do neurônio dentro de um intervalo de valores razoáveis. O sinal de saída y é o valor final produzido pelo neurônio em relação ao um conjunto de sinais de entrada (SILVA, SPATTI, FLAUZINO, 2010).

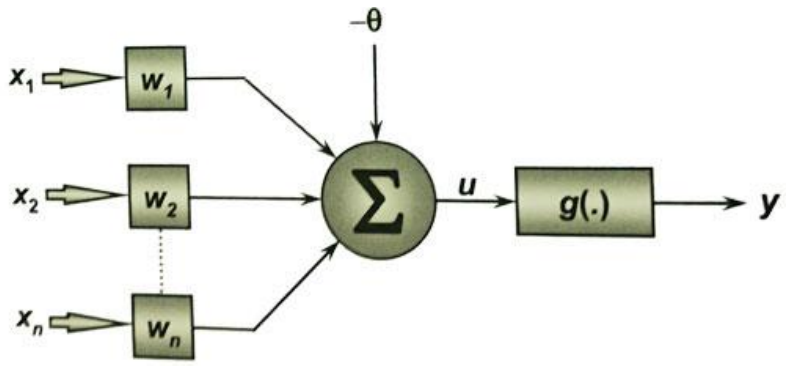

Figura 3 Visão esquemática do neurônio artificial de McCulloch e Pitts
Fonte (SILVA, SPATTI, FLAUZINO, 2010).

De a acordo com Silva, Spatti e Flauzino (2010), a estrutura de redes neurais artificiais é inspirada nos sistemas nervosos biológicos do próprio cérebro humano. Os elementos computacionais denominados neurônios artificiais são modelos bem simplificados dos neurônios biológicos e inspirados

a partir da análise da geração e propagação de impulsos elétricos pela membrana celular dos neurônios.

Os neurônios artificias são não lineares, com saídas tipicamente contínuas e realizam funções como coletar os sinais existentes em suas entradas, agrega-los de acordo com uma função operacional e gerar uma resposta, considerando-se sua função de ativação.

Segundo Braga, Carvalho e Ludermir (2014), o principal atrativo das redes neurais artificiais é a sua capacidade de generalização, ou seja, a capacidade de produzir saídas adequadas para entradas que não estavam presentes entre os dados de treinamento na etapa de aprendizagem. As redes neurais artificiais também são capazes de realizar a extração de informações que não foram apresentadas, de forma explícita, através de exemplos.

Um conjunto de neurônios artificiais conectados na forma de uma rede (neural) é capaz de resolver problemas de complexidade elevada. A Figura 4 mostra um exemplo de configuração possível de neurônios artificiais interconectados na forma de redes neurais artificiais.

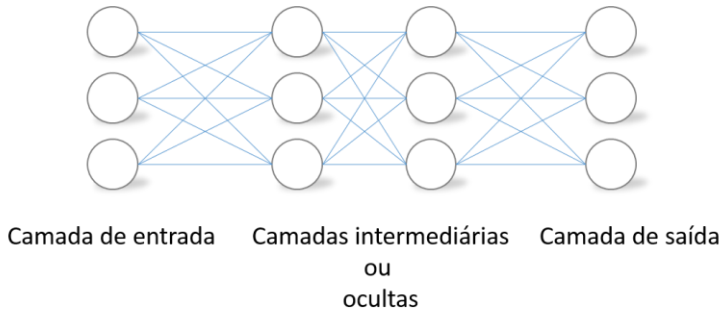

Figura 4 Arquitetura Rede Neural PMC (*Perceptron* Multicamadas)

2.4.1 Treinamento de RNA

Uma das características mais importantes das redes neurais artificiais está em sua capacidade de aprender a partir da apresentação de amostras (padrões) que exprimem o comportamento de um sistema. Após o aprendizado a partir do relacionamento entre as entradas e saída da rede, esta é capaz de apresentar soluções generalizadas que se aproximam daquelas esperadas a partir de quaisquer sinais inseridos em suas entradas (SILVA; SPATTI; FLAUZINO, 2010).

O processo de treinamento de uma rede neural exige a aplicação de passos ordenados necessários para o ajuste dos pesos sinápticos e limiares de seus neurônios para que seja possível a generalização de soluções. O conjunto destes passos, de acordo com Silva, Spatti e Flauzino (2010) é denominado de algoritmo de aprendizagem.

Normalmente, o conjunto total de amostras disponíveis sobre o comportamento do sistema é dividido em dois subconjuntos, denominados de dados de treinamento e dados de teste. O subconjunto de treinamento pode ser composto por cerca de 60% a 90% da totalidade das amostras. Já o conjunto de teste é formado por 10% a 40% da totalidade das amostras do sistema e é empregado para verificar se os aspectos relacionados à generalização de soluções já se encontram em patamares aceitáveis. Durante o treinamento da rede neural, cada apresentação completa das amostras, pertencentes ao subconjunto de treinamento, cujo objetivo é ajustar os pesos sinápticos é denominada de época de treinamento (SILVA; SPATTI; FLAUZINO, 2010).

Segundo Braga, Carvalho e Ludermir (2014), o treinamento de redes neurais pode ser:

- Supervisionado: implica a existência de um supervisor, responsável por estimular as entradas da rede por meio de padrões de entrada e observar a saída calculada, comparando-as com a saída desejada. A resposta da rede é função dos valores atuais do seu conjunto de pesos, que são ajustados de forma a aproximar da saída desejada. Este tipo de treinamento é recomendado para problemas em que se deseja obter um mapeamento entre padrões de entrada e saída. São exemplos de treinamento supervisionado os algoritmos de regra delta e sua generalização para redes de múltiplas camadas, o algoritmo *backpropagation*.
- Não-supervisionado: não há um supervisor externo para acompanhar o processo de aprendizado. Durante o processo de aprendizado, os padrões de entrada são apresentados continuamente à rede. A existência de regularidades torna possível o treinamento. O treinamento não-supervisionado é adequado a problemas que visam a descoberta de características estatisticamente relevantes nos dados de entrada, como a descoberta de agrupamentos ou classes.
- Por reforço: considerado na literatura como um caso particular de treinamento supervisionado. Avalia constantemente a defasagem de valor entre a resposta produzida pela rede em relação à respectiva saída desejada. Os algoritmos ajustam os parâmetros internos dos neurônios por meio de quaisquer informações quantitativas ou qualitativas provenientes da interação com o sistema ou ambiente que está sendo mapeado, para então utilizá-las para medir o desempenho do aprendizado. O processo de treinamento da rede é realizado tipicamente por tentativa e erro. Quando a resposta é considerada satisfatória, são efetuados aumentos graduais nos pesos sinápticos e limiares, visando reforçar a condição comportamental envolvida com o sistema.

Ainda, de acordo com Silva, Spatti e Flauzino (2010),

> Na aprendizagem usando lotes de padrões (*off-line* ou *batch*), os ajustes efetuados nos vetores de pesos são só realizados após a apresentação de todo o conjunto de treinamento, pois cada passo de ajuste leva em consideração o total de desvios observados nas amostras de treinamento frente aos respectivos valores desejados para as saídas.

Na aprendizagem em lotes, as redes necessitam de pelo menos uma época de treinamento para ajustar seus pesos.

Na aprendizagem usando padrão-por-padrão (*on-line*), os ajustes nos pesos são efetuados após a apresentação de cada amostra de treinamento. Assim, a pós a execução do passo de ajuste, a respectiva amostra pode ser descartada. Este tipo de aprendizagem é empregado quando o comportamento do sistema sofre variações de forma bastante rápida, sendo quase impraticável a adoção do aprendizado *off-line*, devido ao fato de que amostras utilizadas em um determinado instante podem não mais ser úteis em instantes posteriores. Desta forma, a rede só passará a apresentar respostas mais precisas, decorrido um número significativo de amostras.

2.4.2 O algoritmo *backpropagation*

O algoritmo *bacpropagation*, também conhecido como algoritmo de retropropagação de erro, é aplicado durante o processo de treinamento em redes perceptron multicamadas (PMC). O processo de treinamento em redes PMC é comumente realizado em duas fases. A primeira, denominada *forward*, os sinais de uma amostra do conjunto de treinamento são inseridos na entrada da rede e propagados camada a camada até a produção das respectivas saídas. As respostas produzidas pelas saídas da rede são comparadas com as respostas desejadas. Os erros entre as respostas desejadas e aquelas produzidas pelos

neurônios de saída são, então, calculados e serão usados em seguida para ajustar pesos e limiares de todos os neurônios.

Em função dos valores de erros, aplica-se em seguida, a segunda fase, do algoritmo *backpropagation*, denominada *backward*, onde os ajustes dos pesos sinápticos e limiares de todos os neurônios são executados. As aplicações sucessivas de *forward* e *backward* fazem com que os pesos sinápticos e os limiares dos neurônios sejam ajustados automaticamente a cada iteração, provocando uma gradativa diminuição da soma dos erros produzidos pelas respostas em comparação às saídas desejadas (SILVA; SPATTI; FLAUZINO, 2010).

Segundo Haykin (2001), o desempenho de aprendizagem da rede é medido pelo erro médio quadrático (EQM) para um dado conjunto de treinamento, que é obtido através do erro. A saída desejada $d_j(n)$ é fornecida para a rede. O sinal de saída para o neurônio j, representado por $y_j(n)$, é comparado com a saída desejada. Assim, o erro na saída, para o exemplo n, é definido por:

$$e_j(n) = d_j(n) - y_j(n) \quad (1)$$

O valor instantâneo da energia do erro é definido por:

$$\frac{1}{2}(e_j(n))^2 \quad (2)$$

O valor instantâneo da energia total do erro é dado pela soma das energias dos neurônios:

$$\varepsilon(n) = \frac{1}{2}\sum_{j=1}^{c}(e_j(n))^2, \quad (3)$$

em que c é o número total de neurônios na camada de saída da rede.

O erro médio quadrático é calculado a partir da média das energias instantâneas totais para todos os n padrões de entrada da rede, dada por:

$$EQM = \frac{1}{N}\sum_{n=1}^{N}\varepsilon(n). \quad (4)$$

O objetivo do processo de aprendizagem da rede é ajustar os pesos de forma a minimizar o EQM. Assim, o algoritmo de retropropagação aplica uma correção $\Delta w_{ji}(n)$ ao peso sináptico $w_{ji}(n)$, que é proporcional à derivada parcial $\partial\varepsilon(n)/\partial w_{ij}(n)$. Como o vetor gradiente possui a mesma direção da maior variação do erro, o ajuste dos pesos ocorre na direção contrária a este, $\Delta w_{ji}(n) \propto -\nabla\varepsilon$, justificando o sinal negativo da correção.

A correção do erro para cada camada é dada por:

$$\Delta w_{ji}(n) = -\eta\frac{\partial\varepsilon(n)}{\partial w_{ji}(n)}, \quad (5)$$

em que η é a taxa de aprendizado que define a velocidade com que os pesos são modificados. Ou, ainda:

$$\Delta w_{ji}(n) = \eta\delta_j(n)y_i(n)$$

(6)

O gradiente local $\delta_j(n)$ varia de acordo com a fase do processamento. Na fase de *forward*, o sinal de entrada $x_j(n)$ é propagado camada a camada e a saída $y_j(n)$ é comparada à saída desejada $d_j(n)$ na camada de saída.

O gradiente local da saída $\delta_j(n)$ é dado por:

$$\delta_j(n) = e_j(n)f'((u_j(n)) \quad (7)$$

em que $f'(.)$ é a derivada da função de ativação e $u_j(n)$ é o resultado da junção somadora dos sinais de entrada, ponderados pelo vetor de pesos $w_{ji}(n)$.

Não existe uma resposta esperada para o neurônio na camada oculta. Assim, o sinal de erro é determinado em função dos neurônios da camada posterior à qual está conectado. O gradiente local para neurônios da camada oculta l é dado por:

$$\delta_j(n) = f'(u_j^{(l)}(n)) \sum_k \delta_k^{(l+1)}(n) w_{kj}^{(l+1)}(n), \qquad (8)$$

em que k se refere a um neurônio que está em uma camada à direita do neurônio j, quando este pertence a uma camada oculta.

No decorrer do treinamento, pode ser empregado o termo *momentum*, que se configura como uma das variações mais simples de ser efetuada no algoritmo *backpropagation*. Trata-se de um parâmetro que visa ponderar o quanto as matrizes sinápticas foram alteradas entre as diversas iterações da rede. Seu valor está compreendido entre 0 e 1. Assim, quando o valor do *momentum* for igual a zero, tem-se o *backpropagation* convencional. Para valores diferentes de zero, a taxa de *momentum* passa a ser relevante, pois quando a solução atual estiver longe da solução final (mínimo da função erro), a variação na direção oposta ao gradiente da função erro quadrático entre duas iterações sucessivas será também grande. Isto significa que há uma diferença considerável entre as matrizes de pesos entre as iterações, cabendo imprimir um incremento maior aos pesos em direção ao mínimo da função erro. O termo *momentum* é o responsável pela medição desta variação (SILVA; SPATTI; FLAUZINO, 2010).

Segundo Silva, Spatti e Flauzino (2010), formalmente, considerando-se os neurônios pertencentes à l-ésima camada, tem-se:

$$w_{ji}^{(l)}(t+1) = w_{ji}^{(l)}(t) + \alpha.(w_{ji}^{l}(t) - w_{ji}^{l}(t-1) + \eta.\delta_{j}^{(l)}.y_{i}^{(l-1)} \quad (9)$$

onde α é definida como taxa de *momentum* e seu valor está compreendido entre 0 e 1.

A Figura 5 ilustra a contribuição do termo *momentum* (TM) e do termo de aprendizagem (TA) visando a convergência em direção ao erro mínimo W^{OT} da função erro quadrático.

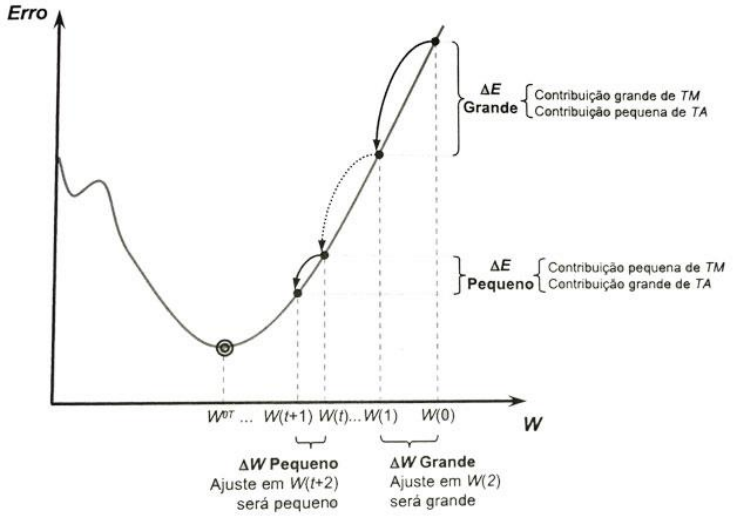

Figura 5 Processo de treinamento usando o método de inserção do termo de *momentum*
Fonte (SILVA; SPATTI; FLAUZINO, 2010)

Quando a solução atual estiver longe da solução final, a variação na direção oposta ao gradiente da função erro quadrático entre duas iterações sucessivas será também grande. Isto implica que a diferença entre as matrizes de pesos dessas duas iterações será bem considerável, podendo imprimir-se, assim um passo maior de incremento para $w^{(L)}$ em direção ao mínimo da função erro.

A execução desta tarefa fica a cargo do termo *momentum*, uma vez que o mesmo é responsável pela medição desta variação.

Por outro lado, quando a solução atual estiver bem próxima da solução final, as variações nas matrizes de pesos serão mínimas, pois a variação do erro quadrático entre as duas iterações sucessivas será baixa. Assim, a contribuição do termo *momentum* no processo de convergência é bem pequena. A partir deste instante, os ajustes nas matrizes de peso são conduzidos quase que em sua totalidade apenas pelo termo de aprendizagem, conforme ocorre no *backpropagation* convencional.

2.4.3 Aplicações de RNA

Segundo Rezende (2005), qualquer problema de aproximações de funções contínuas pode ser resolvido por RNAs, independentemente do número de variáveis. As RNAs são basicamente aplicadas a problemas de predição, classificação, aproximação, categorização e otimização. Como exemplos, pode-se citar reconhecimento de caracteres, reconhecimento de voz, predição de séries temporais, modelagem de processos, controle, entre outros.

A classificação é um dos problemas mais frequentemente tratados em RNAs. Zhang (2000) aponta que a classificação ocorre quando um objeto precisa ser associado dentro de um grupo ou classe predefinido, baseado em um número de atributos observados de um objeto. Muitos problemas na medicina, indústria e ciência podem ser tratados como problemas de classificação, por exemplo, predição de falência bancária, diagnósticos médicos, controle de qualidade, reconhecimento de caracteres escritos à mão, reconhecimento de voz etc.

Para problemas de controle, as RNAs têm sido empregadas, dentre outros campos, no controle de semáforos inteligentes, visando a segurança no trânsito e prevenção de acidentes em cruzamentos. Li (2012) aponta, em seu estudo, parâmetros que requerem análises para buscar soluções inteligentes para semáforos urbanos, tais como: o grau de correlação entre interseções de vias, mudanças de fluxo de veículos, ocupação das vias, velocidade dos veículos.

Outros estudos, empregando RNAs vêm sendo realizados com foco no tráfego de veículos e trânsito em geral, principalmente visando a segurança das vias e a prevenção de acidentes, o que demonstra a viabilidade e importância da utilização de inteligência computacional em novas soluções para o cotidiano da população.

2.5 Avaliação da precisão de uma classificação

O coeficiente Kappa é uma métrica de concordância aplicada para medir o nível de concordância ou discordância de classificadores observando um mesmo fenômeno (COHEN et al, 1960); (ARAÚJO; SHINODA; OLIVEIRA, 2013).

Para avaliar o quão precisa é uma classificação, Congalton (1991) afirma que o uso de coeficiente Kappa (K) é satisfatório na avaliação de precisão de uma classificação temática, pelo fato de levar em consideração toda a matriz de confusão no seu cálculo, inclusive os elementos de fora da diagonal principal, os quais representam as discordâncias na classificação, diferentemente da exatidão global, por exemplo, que utiliza somente os elementos diagonais (concordância real).

Para o cálculo do coeficiente Kappa, é necessário construir um mapa de verdade de campo, para que seja possível fazer uma tabulação cruzada, indicando a proporção de casos presentes e/ou ausentes nos mapas: Mapa Classificado e o Mapa Verdade. Considerando uma situação com duas classes, o resultado é representado em uma tabela onde a célula *a* indica a proporção em que o real (1) e o classificado (2) são corretos; a célula *b*, quando 1 for correto e 2 for errado; *c* quando a observação 1 for correta e a 2 for errada; e *d* quando ambas forem erradas. A Tabela 1 mostra esta abordagem (LOBÃO et al, 2005).

Tabela 1 Tabulação cruzada para cálculo do coeficiente Kappa

Mapa verdade	**Mapa classificado**		
		SIM	**NÃO**
	SIM	a	b
	NÃO	c	d

As equações 10, 11 e 12 mostram como calcular o coeficiente Kappa, de acordo com Lobão et al (2005).

$$Po = a + d \tag{10}$$

$$Pe = (a + b) * (a + c) + (b + d) * (c + d) \tag{11}$$

$$K = (Po - Pe)/(1 - Pe) \tag{12}$$

O valor de K representa a consistência dos resultados obtidos. O coeficiente K neste trabalho está qualificado de acordo com a Tabela 2, adaptada de Galparsoro e Fernández (2001).

Tabela 2 Valor para avaliar o grau de concordância a partir do índice Kappa

Valor do Kappa	**Concordância**
< 0.20	Pobre
0.21 – 0.40	Fraca
0.41 – 0.60	Moderada
0.61 – 0.80	Boa
0.81 – 1.0	Muito boa

2.6 Linguagem Python

Python[2] é uma linguagem de alto nível, orientada a objeto, de tipagem dinâmica, interpretada e interativa. Possui uma vasta coleção de módulos prontos para uso, além de *frameworks* de terceiros que podem ser adicionados (BORGES, 2010).

Quando se diz que Python é uma linguagem interpretada, isto significa que seus comandos podem ser compreendidos tanto pelo sistema operacional quanto pelo hardware. Seu interpretador pode ser intermediado por uma interface de desenvolvimento ou por digitação de comandos.

O código-fonte de um programa Python é identificado pela extensão .py e pode ser executado por meio do comando python seguido do nome do arquivo que contém o código-fonte. Sua sintaxe é bem simples e menos rigorosa quando comparada a de outras linguagens como C, por exemplo.

[2] http://www.python.org

Para o uso de redes neurais artificiais, Python suporta a biblioteca PyBrain (Python-Based Reinforcement Learning, Artificial Intelligence and Neural Network). Trata-se de uma biblioteca de máquina de aprendizado modular para Python. Seu objetivo é fornecer flexibilidade, facilidade de uso, bem como poderosos algoritmos para tarefas de aprendizado. Esta biblioteca contém algoritmos para redes neurais, por reforço de aprendizagem, por aprendizagem supervisionada e evolução (PYBRAIN, 2016).

2.7 Web Services

Um Web Service, segundo a W3C - World Wide Web Consortium (2007) é um sistema de software desenvolvido para suportar interoperabilidade na interação máquina a máquina sobre uma rede. Um Web Service permite o acesso a funções remotas independente da linguagem utilizada. Isto é possível através do protocolo SOAP (Simple Object Access Protocol), baseado em XML (Extensible Markup Language).

O SOAP é um protocolo destinado à troca de informações estruturadas em um ambiente distribuído. Usa o padrão XML para definir uma estrutura de mensagens que podem ser trocadas entre diversos protocolos, sendo o HTTP, o mais comum. O SOAP foi projetado para ser independente de qualquer plataforma de programação particular. Isto define sua característica de interoperabilidade. Consiste de três partes:

- Um envelope que define um quadro para descrever o que está em uma mensagem e como processá-lo;
- Um conjunto de regras de codificação para expressar instâncias de tipos de dados definidos pelo aplicativo;

- Uma convenção para representar chamadas de procedimentos remotos e respostas.

A Figura 6 mostra, de forma geral, como é o fluxo de dados em uma aplicação que utiliza web services.

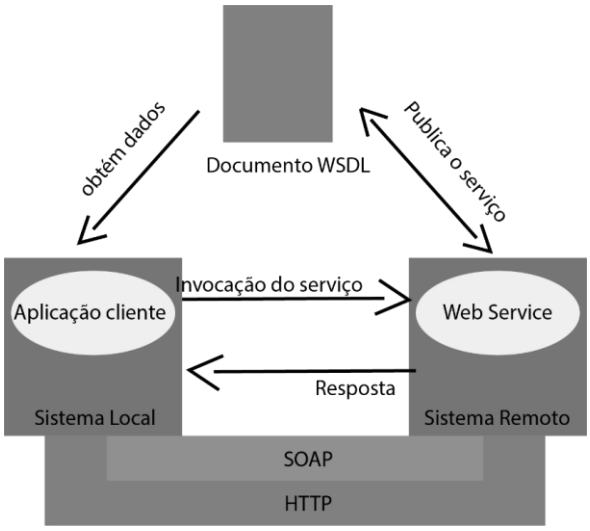

Figura 6 Fluxo de ações de um Web Service

O documento WSDL (Web Service Definiton Language) é um arquivo no formato XML que descreve os serviços em uma rede. As operações do Web Service são descritas de forma abstrata e depois ligadas a um formato de protocolo de rede e uma mensagem concreta W3C – Word Wide Web Consortium (2007). É útil tanto para o sistema local quanto para o sistema remoto a fim de que ambos reconheçam os serviços disponíveis para acesso.

No sistema local, a aplicação cliente faz uma requisição, invocando um método remoto registrado no Web Service. O método encontra-se registrado no documento WSDL, o que permite que o cliente saiba quais métodos poderá invocar no sistema remoto. No sistema remoto, a requisição é processada no

Web Service e é devolvida uma resposta ao cliente, por meio do protocolo SOAP. As requisições são feitas sobre o protocolo HTTP (Hipertext Transfer Protocol).

2.8 Plataforma Android

Android[3] é uma pilha de software para dispositivos móveis que inclui um sistema operacional, um middleware e aplicações-chave. O Android SDK (*Software Development Kit*) provê as ferramentas e APIs (*Application Programming Interfaces*) necessárias para o desenvolvimento de aplicações para esta plataforma, usando a linguagem de programação Java (GLAUBER, 2015).

Conforme afirma Glauber (2015), o sistema operacional Android tem como base o kernel Linux, responsável pelo gerenciamento de processos, memória, drivers e energia. O middleware é responsável por controlar a interação entre aplicativos instalados em um aparelho. As aplicações-chave são programas comuns como navegadores, discadores, gerenciadores de mensagens etc (GLAUBER, 2015).

De acordo com o site oficial para desenvolvedores Android, Developers Android (2016), o Android fornece uma estrutura de aplicativo adaptativa que permite fornecer recursos exclusivos para diferentes configurações de dispositivos. Por exemplo, é possível criar diversos documentos de configuração XML para diferentes tamanhos de tela e o sistema escolhe qual o melhor layout para aplicar com base no tamanho da tela do dispositivo atual. A Figura 7 apresenta telas de uma mesma aplicação, configuradas com arquivos de layout XML distintos.

[3] https://www.android.com/intl/pt-BR_br/

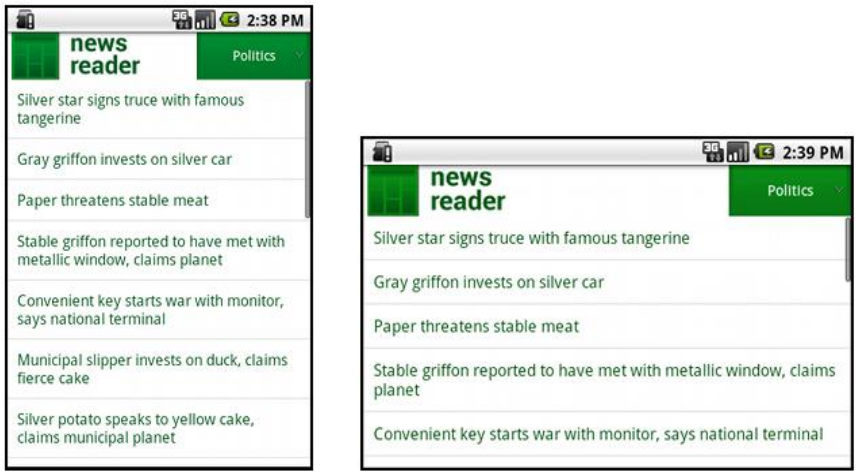

Figura 7 Telas com layouts XML distintos
Fonte (DEVELOPERS ANDROID, 2016)

A primeira versão do Android foi lançada em 2008. Suas versões posteriores seguiram uma nomenclatura onde atribui-se um nome de um doce e um número sequencial denominado API *Level*. Esta informação é relevante para se saber quais recursos, classes e bibliotecas estão disponíveis em cada versão.

A Tabela 3 mostra as versões do Android lançadas até a conclusão desta dissertação.

Tabela 3 Versões do Android (GLAUBER, 2015)

Nome da versão	Versão	API Level
CupCake	1.5	3
Donut	1.6	4
Eclair	2.0 2.0.1 2.1	5 6 7
Froyo	2.2	8
Gingerbread	2.3 2.3.3	9 10
Honeycomb	3.0 3.1 3.2	11 12 13
Ice Cream Sadwich	4.0 4.0.3	14 15
Jellybean	4.1 4.2 4.3	16 17 18
KitKat	4.4 4.4W (wear)	19 20
Lollipop	5.0	21

Segundo Glauber (2015), conhecer as versões é importante para saber quais APIs estão disponíveis para as aplicações, precavendo-se de que sejam acessadas classes que só estejam disponíveis em uma determinada versão, evitando, assim, possíveis erros.

A cada nova versão, o Google[4] disponibiliza a chamada versão pura da plataforma, ou seja, uma versão sem modificações feitas pelo fabricante do aparelho. Esses aparelhos compõem a linha Nexus.

No próximo capítulo, passarão a ser discutidas as técnicas e a metodologia empregadas para o desenvolvimento de um sistema inteligente de

[4] http://www.google.com

classificação de condutores, com a finalidade de alertar proprietários de veículos quanto à condução desautorizada de seus automóveis.

3 MATERIAIS E MÉTODOS

Este capítulo descreve a metodologia e as técnicas empregadas para o desenvolvimento do sistema proposto, destacando as tecnologias e dispositivos necessários.

3.1 Visão geral do sistema proposto

O Sistema Neural Antifurto Veicular tem como função verificar a autenticidade de um condutor, mediante sua classificação através de uma Rede Neural Artificial. Para isso, o sistema conta com a implantação de um dispositivo OBD-II, com conexão *bluetooth* ao veículo. Tal dispositivo fornece dados provenientes dos sensores do veículo para o aplicativo Torque. Este aplicativo grava dados pré-selecionados em um arquivo .csv, tais como a posição do acelerador e as acelerações em x (aceleração na reta), aceleração em y (aceleração na subida) e aceleração em z (aceleração na curva) do veículo. Estes dados, então, constituem o vetor de entrada de 4 posições (posição do acelerador, aceleração em x, aceleração em y e aceleração em z) para uma rede neural que classifica o condutor como autorizado ou não autorizado a conduzir o veículo.

Um aplicativo Android recolhe os dados gravados no arquivo .csv do Torque e os envia, através de um socket para um servidor remoto. Neste servidor, há um Web Service Python que faz uma invocação ao método que utiliza a rede neural treinada para classificar o condutor. As saídas esperadas são: 0 para o condutor desautorizado e 1 para o condutor autorizado. Uma vez realizada a classificação, a resposta é enviada a um servidor Web, via protocolo

HTTP para que o cliente (proprietário) do veículo possa ser informado sobre a condução autorizada ou não do veículo. A Figura 8 ilustra o funcionamento do sistema, de forma geral.

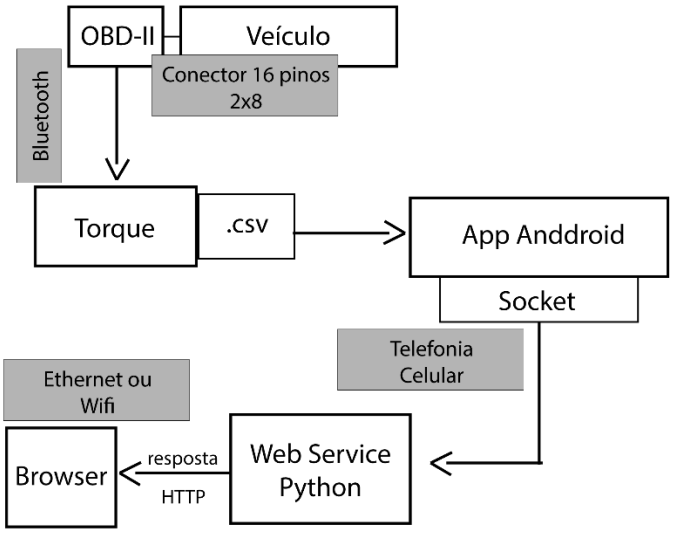

Figura 8 Visão geral do sistema

3.2 Coleta e seleção dos dados

Para a realização deste trabalho, inicialmente, foi conduzida uma pesquisa bibliográfica, visando verificar a viabilidade da aplicação de redes neurais artificiais para classificar condutores de veículos mediante seu comportamento na direção. Kumar e Prasad (2015) apontam dois tipos de modelos para análise de comportamento de motoristas: análise do comportamento do motorista e modelos de predição do comportamento do motorista. O primeiro tipo, adotado neste trabalho, se baseia em um conjunto de

ações que o motorista deve realizar para garantir a segurança das pessoas, bem como o cumprimento das normas de condução.

Uma vez que o objetivo principal deste trabalho é desenvolver um sistema inteligente de classificação de condutores de veículos, optou-se por utilizar o dispositivo OBD-II como meio de obtenção de dados para alimentar as entradas de uma rede neural artificial. Este dispositivo é conectado ao computador de bordo do veículo e transmite os dados passíveis de obtenção por meio de uma conexão *bluetooth*, específica do modelo utilizado neste trabalho.

Conforme afirmam Godavarty, Broyles e Parten (2000), as informações obtidas do computador de bordo dos veículos modernos, por meio do OBD II, ajudam a melhorar a compreensão da relação entre os hábitos de direção e o desempenho do veículo.

Para receber os dados provenientes do OBD-II, foi utilizado o aplicativo Torque[5]. O aplicativo é oferecido nas versões gratuita e paga para a plataforma Android. Permite selecionar os dados a serem obtidos, gerando um arquivo no formato .csv que contém as informações monitoradas. Este arquivo foi utilizado na filtragem dos dados. A Figura 9 mostra um exemplo de parte de um arquivo gerado pelo Torque, contendo dados recebidos pelo smartphone Android via Bluetooth. Nela, pode-se observar a obtenção da posição do pedal do acelerador, a aceleração em x (reta) e a aceleração em y (subida). Nesta figura, não aparece a captura da aceleração em z, devido à dimensão da tela de captura do dispositivo, porém esta variável também é capturada. Muitos outros valores retornados pelos sensores do veículo podem ser obtidos. Um maior detalhamento sobre eles está na documentação do aplicativo Torque, disponível em http://torque-bhp.com/wiki/Dials.

[5] http://torque-bhp.com

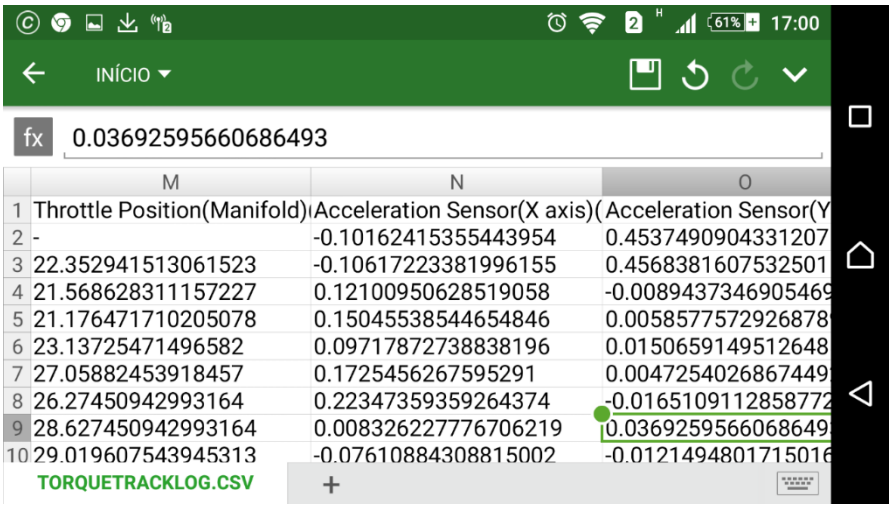

Figura 9 Exemplos de dados preliminares obtidos pelo aplicativo Torque durante teste com condutor

3.3 Pré-processamento dos dados

Para coletar os dados do veículo, por meio do dispositivo OBD-II, três diferentes condutores foram utilizados, sendo dois do sexo masculino, um com idade de 42 anos e outro com 55 anos e um do sexo feminino com idade de 33 anos. O veículo utilizado por todos os condutores foi um Chevrolet Ágile 1.4. O percurso utilizado para a coleta dos dados foi o mesmo para todos os condutores, perfazendo aproximadamente 10 Km, incluindo retas, aclives, declives e curvas, a fim de que fossem capturadas todas as variáveis úteis ao estudo.

Os dados referentes à posição do acelerador, aceleração em x, aceleração em y e aceleração em z são obtidos do OBD-II em um intervalo de 2 segundos, a fim de capturar informações com maior detalhamento. Isto gera um número considerável de amostras, sendo importante filtrá-las. O processo de filtragem ajuda a eliminar ruídos que podem gerar informações incorretas ao sistema e

também excluir valores discrepantes que podem fornecer resultados incoerentes. Assim, considera-se o método das médias móveis importante neste sentido, pois é apropriado quando se tem uma série temporal cuja componente sazonal varia com o tempo, ou seja, para séries cuja sazonalidade é estocástica (VILLAMAGNA, 2013).

A média móvel mostra o valor médio dos dados em determinado período, conforme a seguinte fórmula:

$$M_t = \frac{(R_t + R_{t-1} + R_{t-2} + \cdots + R_{t-n+1})}{n} \quad (13)$$

Onde:

M_t: Média móvel do período t

R_t: Valor real observado no período t

n: Número de períodos considerados na média móvel

As médias móveis são utilizadas para suavizar ruídos, de maneira que seja fácil identificar e definir tendências. O resultado da média móvel depende dos valores cuja média está sendo calculada e a amplitude do intervalo temporal desejado.

Após o pré-processamento, onde foi utilizado o cálculo de médias móveis em um intervalo de 2 minutos entre as medições para filtrar os dados, foram recolhidas 4298 amostras das variáveis selecionadas entre 3 condutores, sendo um considerado como proprietário autorizado a conduzir o veículo e outros dois como desautorizados. Do total de amostras, 80% foi utilizado para treinamento e 20% para testes.

3.4 Implementação da Rede Neural Artificial

Para treinar uma RNA com a tarefa de classificar um condutor veicular como sendo autorizado ou não a dirigir um automóvel, é preciso estabelecer alguns parâmetros e critérios, de acordo com a metodologia de Redes Neurais Artificiais (RNA), a saber:

a) Separar dados para treinamento e teste;

b) Normalizar os dados;

c) Definir a arquitetura da rede (número de neurônios na camada de entrada, camada escondida e camada de saída; número de camadas escondidas; escolha da função de ativação);

d) Seleção do algoritmo de treinamento;

e) Definição da taxa de aprendizado e termo momento;

f) Definir critérios de parada de treinamento.

O uso correto desta metodologia implica a separação dos dados a serem analisados em duas partes: dados de treinamento para calibrar a rede e dados de teste para validá-la. Neste trabalho, separou-se 80% dos dados obtidos para treinamento e 20% para teste.

Para uma melhor classificação, os dados devem estar definidos sob determinados limites, a fim de evitar que valores de uma determinada magnitude possam inviabilizar alguns tipos de modelos. Dependendo da aplicação e/ou tecnologias empregadas, é comum que os dados sejam normalizados, dividindo-se os valores de entrada pelo valor máximo do conjunto de dados, obtendo-se, assim, valores entre 0 e 1. Para este trabalho, esta normalização não foi necessária, uma vez que as bibliotecas Python empregadas fazem o pré-

processamento dos dados, entregando-os já normalizados, dividindo os valores de entrada pelo valor máximo do conjunto de dados, obtendo valores entre 0 e 1.

A função de ativação utilizada foi a sigmoide, pois fornece a saída no intervalo de 0 a 1. Assim, optou-se por classificar um condutor como sendo 1 para um condutor autorizado e 0 para um condutor desautorizado.

Foram utilizadas redes de múltiplas camadas, compreendendo uma camada de entrada, 2 camadas escondidas e uma camada de saída. Foram utilizados 10 neurônios na camada de entrada e 10 neurônios nas camadas escondidas. Estes valores apresentaram melhores resultados durante os treinamentos realizados. O algoritmo de aprendizagem empregado foi o *backpropagation*. Segundo Villamagna (2013), este algoritmo modifica os pesos da rede de forma orientada, para encontrar na superfície de erro, valores para os pesos sinápticos que minimizam os erros da rede. Vários treinamentos foram realizados até se chegar àquele em que se obteve a melhor rede neural treinada a ser implementada. A Figura 10 ilustra a rede neural utilizada.

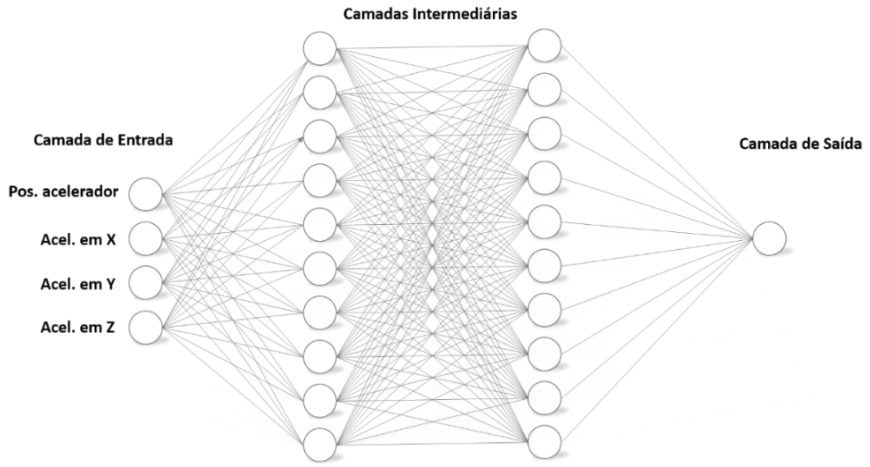

Figura 10 Rede neural utilizada

As figuras 11, 12 e 13 apresentam gráficos de treinamentos realizados até que se chegasse àquele com os melhores resultados.

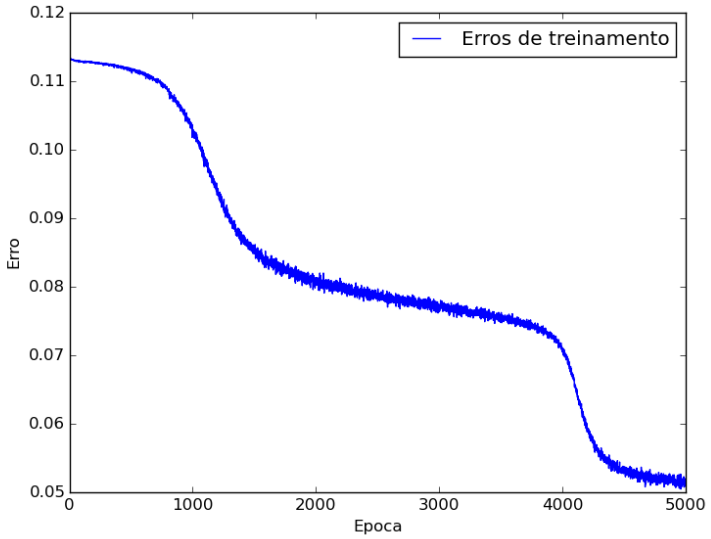

Figura 11 Treinamento com taxa de aprendizagem 0,001 e 5000 épocas

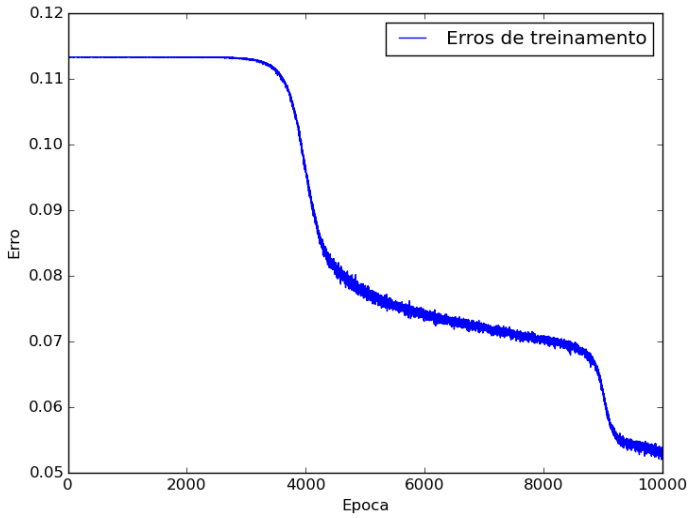

Figura 12 Treinamento com taxa de aprendizagem 0,001 e 10000 épocas

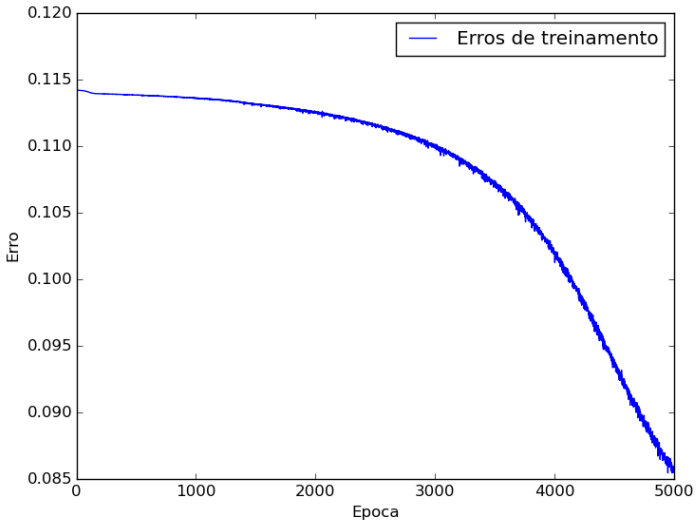

Figura 13 Treinamento com taxa de aprendizagem 0,0025 e 5000 épocas

Os testes foram realizados considerando uma taxa de aprendizado de 0,001. Tal valor foi escolhido por apresentar uma variação de erro menor durante as iterações do treinamento. Outros valores foram testados, apresentando consideráveis oscilações. De acordo com Haykin (2001), uma taxa de aprendizado muito pequena torna o treinamento lento, enquanto que valores muito grandes podem provocar divergência do processo de treinamento.

A constante momento pode ou não ser usada no algoritmo de treinamento. Ela é útil para evitar que o processo de aprendizagem termine em um mínimo local na superfície do erro. Seu valor pode variar entre 0 e 1. O valor escolhido foi 0,025. O critério de parada escolhido foi um número de 8000 épocas. Os valores selecionados para o momento e o número de épocas foram os que produziram melhores resultados durante o treinamento.

Assim, após os testes realizados, verificou-se que os parâmetros que proporcionaram melhores resultados foram: taxa de aprendizado: 0,001; *momentum*: 0,025 e número de épocas: 8000, sendo considerados adequados às expectativas de respostas.

3.5 Desenvolvimento de Web Service Python

Neste trabalho, para criar um Web Service Python que disponibiliza a função de classificação do condutor, empregando metodologia RNA, foi utilizada a biblioteca SOAPpy[6]. Esta biblioteca permite desenvolver Web Services Python sobre vários protocolos, incluindo o SOAP. Para a execução do

[6] https://pypi.python.org/pypi/SOAPpy

Web Service Python, um servidor Web Apache, versão 2, foi configurado em uma máquina Linux sob o endereço IP 187.108.64.177.

Uma vez disponibilizado o Web Service, foi desenvolvido um aplicativo Android para enviar os dados recebidos do OBD-II para a classificação na rede neural. A rede neural treinada encontra-se no servidor remoto. O aplicativo recebe os dados do veículo que são armazenados em um arquivo .csv no dispositivo por meio do aplicativo Torque, conectado ao OBD-II e os envia à função do Web Service. A Figura 14 mostra um exemplo do momento em que o servidor está recebendo dados do aplicativo Android.

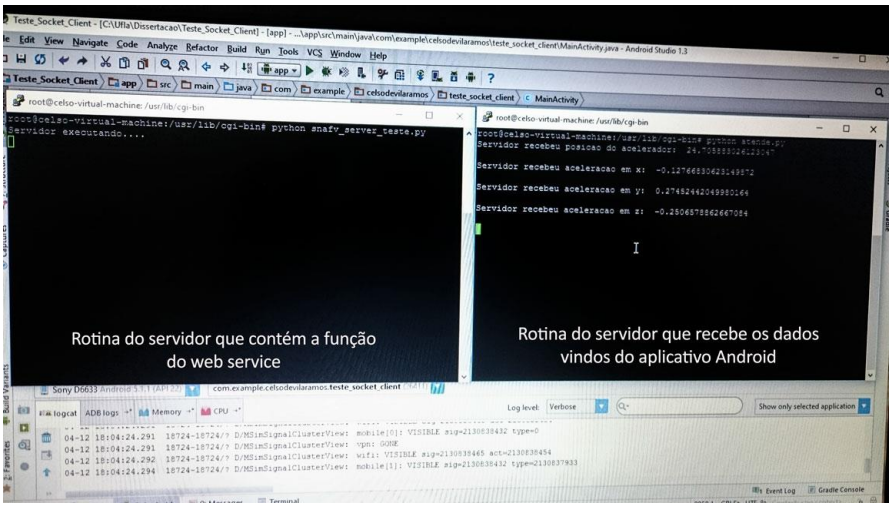

Figura 14 Servidor recebendo dados vindos do aplicativo Android

A função remota que está no Web Service recebe os dados de entrada, classifica o condutor baseada na rede neural treinada, armazenada no servidor e retorna a classificação do condutor como sendo um condutor autorizado do veículo ou um condutor desautorizado, conforme ilustra a figura 8 no início deste capítulo. O envio dos dados armazenados no arquivo .csv do dispositivo para o Web Service remoto é realizado através de pacotes UDP (User Datagram

Protocol), por meio de um socket de comunicação. No servidor, após o recebimento, os dados são gravados em um arquivo texto. Este arquivo é usado pela função de verificação para recuperar os últimos dados recebidos e enviá-los ao Web Service para a classificação pela rede neural. A figura 15 ilustra este processo.

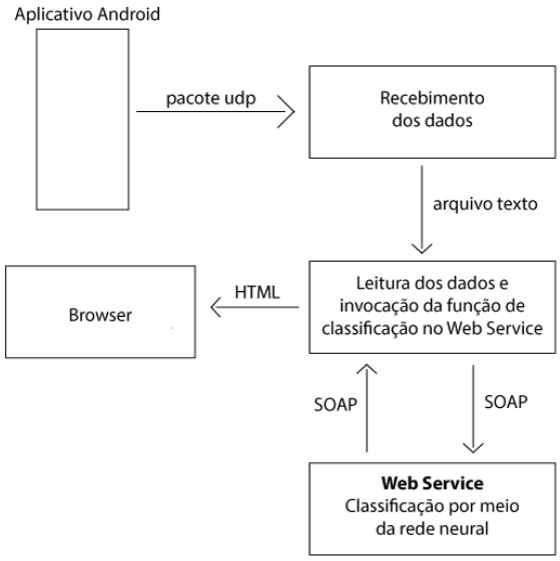

Figura 15 Fluxo dos dados desde o aplicativo Android até a classificação e exibição no browser

3.6 Preparação dos testes

Para a realização dos testes, foi necessário recolher 20% das amostras compreendendo dados de um condutor autorizado e de condutores não autorizados. Estes valores foram, então, submetidos a uma rotina de testes escrita em Python, a fim de identificar o desempenho do sistema em termos de

acertos e erros, o que possibilitou analisar com precisão as respostas fornecidas pela rede neural e o quão poderia ser eficiente sua atuação em um universo real.

No próximo capítulo, serão apresentados e discutidos os resultados dos testes realizados.

4 RESULTADOS E DISCUSSÃO

Neste capítulo, serão apresentados os resultados obtidos com o Sistema Neural Antifurto Veicular, abordando a utilização de Redes Neurais Artificias, a aplicação Android desenvolvida e a disponibilização do Web Service Python no servidor Apache.

4.1 Aplicação de Redes Neurais Artificiais

A utilização de Redes Neurais Artificiais para a classificação de condutores de um veículo segundo informações obtidas do próprio automóvel se mostrou adequada para o auxílio a proprietários de veículos automotivos. Percebeu-se que a seleção correta de variáveis de entrada para o treinamento da rede neural é crucial para a obtenção de respostas mais precisas, com menor ocorrência de erros. Variáveis como velocidade do veículo, rotação do motor e consumo de combustível apresentaram variações muito amenas quando comparadas entre diferentes condutores. Por este motivo, foram descartadas neste estudo, sendo mantidas apenas aquelas que apresentaram maiores variações, a fim de se obter uma classificação mais confiável.

O trabalho desenvolvido por Shi-Huang Chen, Jeng-Shyang Pan e Kaixuan Lu (2015) que também utilizou o OBD-II para obtenção de dados do veículo para classificar o estilo de condução do motorista foi extremamente relevante para a seleção das variáveis de entrada neste trabalho, a saber: posição do acelerador, aceleração em x, aceleração em y e aceleração em z.

O cálculo das médias móveis possibilitou suavizar os ruídos gerados na captura dos dados brutos, uma vez que este são obtidos em um intervalo de 2 segundo pelo OBD-II. Assim, optou-se por fazer uma filtragem, por meio do cáclulo de médias móveis em um intervalo de 2 minutos, de forma a caracterizar, com maior precisão, o estilo de cada condutor. A Figura 16 apresenta os dados sem filtragem para a variável posição do acelerador, em um intervalo de 250 segundos. A Figura 17 apresenta um gráfico cujos dados obtidos do dispositivo OBD-II pelo aplicativo Torque foram filtrados por média móvel simples, em janelas de 2 minutos, considerando a variável posição do acelerador.

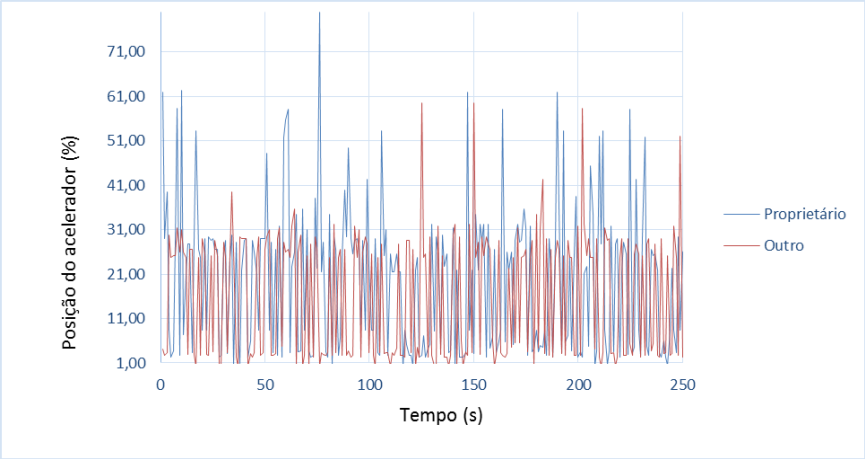

Figura 16 Dados sem filtragem por média móvel para a variável posição do acelerador

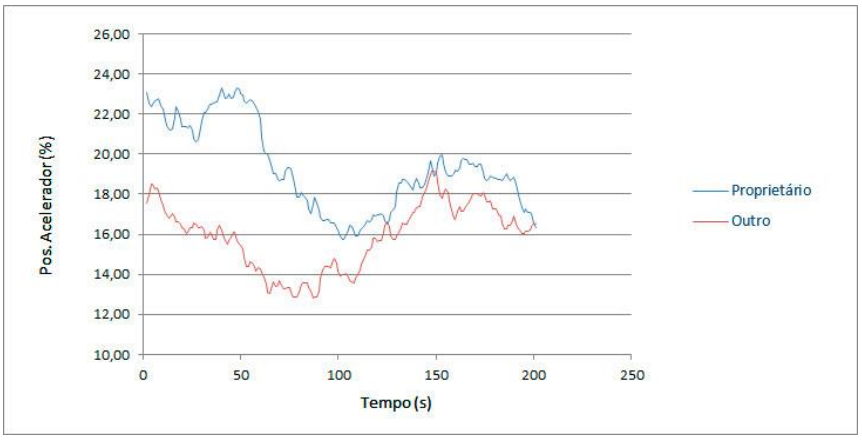

Figura 17 Suavização de ruído por média móvel para a variável posição do acelerador

Observa-se que, com os dados brutos, em intervalos de 2 segundos, a complexidade em visualizar e separar as características de dois condutores tomando-se como referência a posição do acelerador, neste caso, é considerável. Por outro lado, ao se aplicar a filtragem por médias móveis, é possível observar melhor os dados e distinções entre dois condutores, pela suavização das curvas para a mesma variável, posição do acelerador.

Foram recolhidas 4298 amostras das variáveis selecionadas entre 3 condutores, sendo um considerado como proprietário autorizado a conduzir o veículo e outros dois como desautorizados. Do total de amostras, 80% foi utilizado para treinamento e 20% para testes. A Figura 18 mostra o gráfico de erros ao final do treinamento, cujos parâmetros utilizados geraram os melhores resultados.

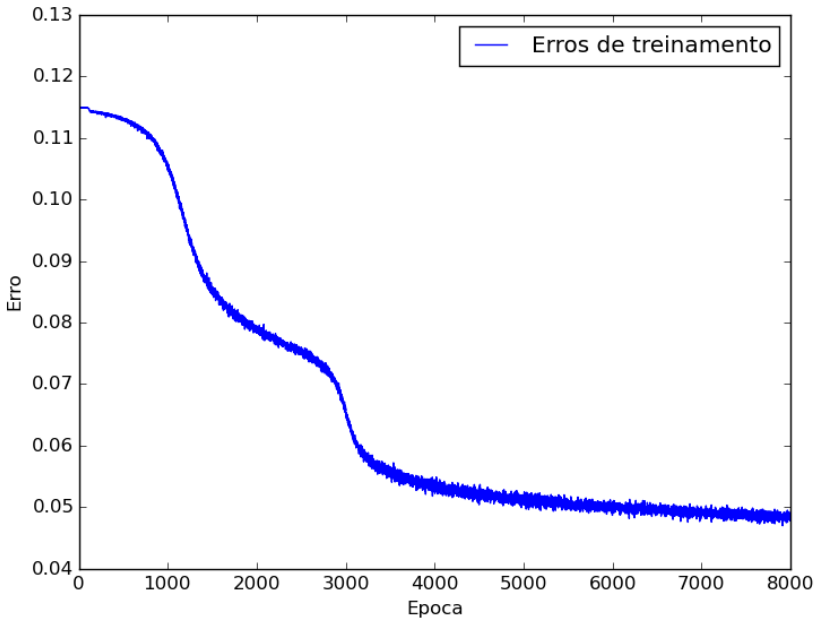

Figura 18 Treinamento com taxa de aprendizagem 0,001 e 8000 épocas

Os 20% das amostras de dados destinados aos testes apresentaram um erro aproximado de 0,045, resultando em 88% de acertos. Outros testes foram realizados, a fim de se encontrar o menor erro e uma porcentagem ótima de acertos. Dentre os testes realizados, a média de acertos foi de 86,2%, com desvio de 1,6% para mais ou para menos (86,2%±1,6%), conforme mostra a tabela 4. Esta tabela também apresenta as respectivas taxas de aprendizagem, número de épocas e valor do termo *momentum* de cada treinamento.

Tabela 4 Média e desvio padrão dos treinamentos realizados

Testes	Taxa de aprendizagem	Épocas	Momentum	Acertos (%)
1	0,0025	5000	0,5	84
2	0,001	10000	0,5	85
3	0,0025	6000	0,5	87
4	0,001	5000	0,5	87
5	0,001	8000	0,025	88
Média				**86,2**
Desvio padrão				**1,6**

A avaliação da eficiência da classificação da rede neural foi medida por meio do coeficiente Kappa e seu resultado é demonstrado na matriz de confusão da Tabela 5. A tabela mostra os valores obtidos no treinamento que apresentou o melhor resultado, com taxa de aprendizagem 0,001; 8000 épocas e *momentum* 0,025. Observa-se que o valor de K resultou em um índice de concordância considerada muito boa, de acordo com os valores apresentados na Tabela 2 deste trabalho.

Tabela 5 Matriz de confusão e resultados de coeficiente Kappa

		Classificado		
		Autorizado	**Desautorizado**	**Total**
Obtido do OBD	**Autorizado**	562 (65,35%)	19 (2,21%)	581 (67,56%)
	Desautorizado	78 (9%)	201 (23,37%)	279 (32,37%)
	Total	640 (74,35%)	220 (25,58%)	
K	**99,8%**			

4.2 Comunicação com aplicativo Android

Foi desenvolvido um aplicativo para a plataforma Android, cuja funcionalidade principal é ler os dados do veículo que foram obtidos do dispositivo OBD-II e armazenados em um arquivo .csv no dispositivo pelo aplicativo Torque. Assim que os dados são lidos, são imediatamente enviados em pacotes UDP, por meio de sockets para o servidor, onde o sistema de classificação de condutores está sendo executado. A classificação é realizada e a resposta é enviada para uma aplicação Web, acessível em qualquer programa navegador. Isto possibilita acessar as respostas fornecidas pelo sistema tanto em um *smartphone* quanto em qualquer outro dispositivo que suporte um navegador web, por meio do endereço 187.108.64.177/cgi-bin/snafv_verifica.py. A Figura 19 mostra um acesso ao sistema via navegador, onde os dados de um condutor foram enviados pelo aplicativo Android ao Web Service. Neste exemplo, a rede neural retornou uma saída igual a 0,032..., o que caracteriza um resultado menor que 0,5, considerado como condutor desautorizado pelo sistema.

Figura 19 Resposta do Web Service enviada ao navegador do usuário, caracterizando um condutor desautorizado

A figura 20 mostra um acesso ao sistema via navegador em que a rede neural retornou uma saída igual a 0,97, caracterizando um condutor autorizado.

Figura 20 Resposta do Web Service enviada ao navegador do usuário, caracterizando um condutor autorizado

5 CONCLUSÃO

O Sistema Neural Antifurto Veicular é uma ferramenta que pode auxiliar proprietários a monitorar a condução de seu automóvel. Espera-se, também, que o sistema possa auxiliar outras áreas de interesse como autoridades e empresas de seguro, por exemplo.

O uso de Redes Neurais Artificiais para a classificação do condutor mostrou-se viável e eficaz para este fim. Também é importante ressaltar que o dispositivo OBD-II pode ser empregado para outras finalidades que vão além do diagnóstico dos componentes do veículo para sua correta manutenção. O sistema desenvolvido comprovou que é possível avaliar o comportamento do motorista por meio de dados fornecidos pelo próprio veículo que este conduz.

Quanto à comunicação entre o aplicativo Android e o sistema inteligente, a transmissão de pacotes UDP pode gerar possíveis perdas de dados, devido às características deste tipo de pacote, em que não se estabelece conexão entre as partes, não sendo possível garantir a entrega dos pacotes. No entanto, este fator não se mostrou preocupante durante os testes, pois os dados são sempre enviados e atualizados em intervalos de 1 minuto. Assim, a possível perda de um pacote em um instante pode ser compensada no instante seguinte, sem maiores danos para o desempenho do sistema.

Estudos futuros poderão adicionar outras funcionalidades ao sistema, tais como envio de SMS ao proprietário do veículo, contendo informações sobre a condução desautorizada; envio da localização geográfica do veículo, em caso de condução desautorizada; envio de sinais ao veículo para a interrupção de um ou mais sistemas e utilização do sistema com um microcontrolador para obter os

dados do veículo e enviar ao servidor para a classificação. Pode-se ainda, testar a rede neural com um número maior de amostras. Para tanto, sugere-se a disponibilização de um banco de dados colaborativo para a coleta de novos dados e, consequentemente, a realização de mais testes. Os dados coletados neste trabalho são, em sua maioria, relacionados ao proprietário do veículo, devido à maior disponibilidade para o recolhimento das amostras. Sugere-se uma maior proporcionalidade em termos de condução autorizada e desautorizada entre as amostras de cada condutor.

Propõe-se, ainda, o emprego de curva ROC (*Receiver Operator Characteristic)* para auxiliar a descrever quantitativamente o desempenho dos testes. Esta ferramenta poderá fornecer informações importantes quanto à sensibilidade, ou seja, a probabilidade de o teste fornecer um resultado positivo (condutor autorizado) e a especificidade, ou seja, a probabilidade do teste fornecer um resultado negativo (condutor desautorizado) por meio de pontos de corte. Uma de suas vantagens é que a proporção das amostras coletadas não interfere em seus resultados

O Sistema Neural Antifurto Veicular, além de uma preciosa contribuição científica, também apresenta um grande potencial comercial, comprovado pelas necessidades atuais em segurança de bens patrimoniais e, certamente pode ser empregado para estes fins em um futuro bem próximo.

REFERÊNCIAS

ARAÚJO, N. V. de S.; SHINODA, A. A.; OLIVEIRA, R. de. **Kappa-artmap fuzzy**: uma metodologia para detecção de intrusos com seleção de atributos em redes de computadores. SBRC, Universidade de Brasilia, Brasilia - DF, Brasil, 2013.

BAEK, S.-H. **Implementation Vehicle Driving State System with**. 17th Asia-Pacific Conference on Communications (APCC). Sabah - Malasya: [s.n.]. 2011.

BASTOS, E. **Estudo das diferenças dos requerimentos das principais legislações de On Board Diagnostics para padronização de testes de desenvolvimento e validação de transmissão automática de automóveis**. Centro Universitário do Instituto Mauá de Tecnologia. São Caetano do Sul, p. 58p. 2012.

BORGES, L. E. **Python para desenvolvedores**. 2 ed. Rio de Janeiro: Edição do autor, 2010.

BRAGA, A.P; CARVALHO, A.P.L.F; LUDERMIR, T. B. **Redes neurais artificiais**: teoria e aplicação. 2 ed. São Paulo: LTC, 2014.

CARPLUGS. Premium Male Connector, Black Shell (full pins), 2014. Disponivel em: <http://www.carplugs.com/products.html>. Acesso em: 30 agosto 2014.

CHONG, L. et al. A rule-based neural network approach to model driver naturalistic behavior in traffic. **Transportation Research Part C: Emerging Technologies**, v. 32, p. 207-223.

COHEN, J. et al. A coeficient of agreement for nominal scales. **Educational and psychological measurement**, Durham, v. 20, n. 1, p. 37-46, 1960.

CONGALTON, R. G. 1991. A review of assessing the accuracy of classifications of remotely sensed data. **Remote Sensing of Environment**, volume 49, p. 1671-1678.

D'AGOSTINO, R.; REIS, T.; MACEDO, L. **G1**, 2014. Disponivel em: <http://g1.globo.com/sao-paulo/noticia/2014/02/em-sp-cada-5-veiculos-roubados-ou-furtados-2-sao-recuperados.html>. Acesso em: 20 agosto 2014.

DEVELPOERS ANDROID. **Getting started**. Disponível em: <http://developer.android.com/intl/pt-br/training>. Acesso em: 31 jan. 2016.

GLAUBER, N. **Dominando o Android**: do básico ao avançado. São Paulo: Novatec, 2015.

GODAVARTY, ; BROYLES, S.; PARTEN, M. Interfacing to the on-board diagnostic system. **Vehicular Technology Conference**, Boston, v. 4, n. 52, p. 2000-2004, september 2000.

HAYKIN, S. **Neural networks, a comprehensive foudation**. New Jersey: Englewood Cliffs. 2001. 842p.

HUANG, S. H.; ZHANG, H.-C. Applications of neural networks in manufacturing: a state-of-the-art survey. **International Journal of Production Research**, 33, n. 3, 2007. 705-728.

INFOSEG. **SENASP lança aplicativo para consulta de veículos roubados e furtados**, 2014. Disponivel em: <http://www.infoseg.gov.br/noticias/senasp-lanca-aplicativo-para-consulta-de-veiculos-roubados-e-furtados>. Acesso em: 30 agosto 2014.

KAMAL, R. **Embedded Systems - Architecture, Programming and Design**. 2. ed. New Delli: McGraw Hill, 2008.

LI,. **A survey of urban traffic coordination controls in intelligent transportation systems**. Service Operations and Logistics, and Informatics (SOLI), 2012 IEEE International Conference on. Suzhou: IEEE. 2012. p. 177-182.

LOBÃO, J. S. B.; ROCHA, W. J. S. F. & SILVA, A. B. 2005. Aplicação dos Índices KAPPA & PABAK na validação da classificação automática de imagem de satélite em Feira de Santana-BA. In Simpósio Brasileiro de Sensoriamento Remoto (SBSR), volume 12, Goiânia. São José dos Campos: INPE. p. 1207-1214.

MARWEDEL, P. **Embeddes systems Foundations of Cyber-Physical Systems**. 2. ed. Dortmund: Springer, , 2011.

MESEGUER, J. E. et al. DrivingStyles: A smartphone application to assess driver behavior. **IEEE Symposium on Computers and Communications (ISCC)**, Split, Croatia, p. 435-540, 2013.

NEWMAN, S. et al. Safety in occupational driving: development of a driver behaviour scale for the workplace context. **Applied Psichology**, v. 60, n. 4, p. 576-599, 2011.

PYBRAIN. **The Python learning machine library**. Disponível em: <http://pybrain.org>. Acesso em: 11 jan. 2016.

REZENDE, S. O. **Sistemas inteligentes**. Barueri: Manole, 2005.

SAE. **J1962 Diagnostic connector equivalent to ISO15031-3**. Warrandale, 2002

SHI-HUANG CHEN; JENG-SHYANG PAN; KAIXUAN LU. Driving Behavior Analysis Based on Vehicle OBD Information and AdaBoost Algorithms. **Proceedings of the International MultiConference of Engineers and Computer Scientists**, v. 1, 2015.

SILVA, I. N; SPATTI, D. H; FLAUZINO, R. A. **Redes neurais artificiais para engenharia e ciências aplicadas**. São Paulo: Artliber, 2010.

VIEIRA L. B. **Avaliação da adesão à terapêutica medicamentosa de pacientes idosos hipertensos antes e após o desenvolvimento e uso de um Sistema Eletrônico de Uso Personalizado e Controlado de Medicamentos (SUPERMED**) (tese de doutorado). Ribeirão Preto: Escola de Enfermagem de Ribeirão Preto, Universidade de São Paulo; 2013. Disponível em: <http://www.teses.usp.br/teses/disponiveis/22/22132/tde-17012014-110238/pt-br.php> Acesso em 25 mai. 2015.

VILLAMAGNA, M. R. **Seleção de modelos de séries temporais e redes neurais artificiais na previsão de consumo e demanda de energia elétrica**. Dissertação (mestrado). Lavras: UFLA, 2013. 113p.

W3C - WORLD WIDE WEB CONSORTIUM. **SOAP version 1.2**. Disponível em: <https://www.w3.org/TR/ws-arch/#whatis>. Acesso em: 13 fev. 2016

WEFKY, A. M. et al. Alternative Sensor System and MLP Neural Network for Vehicle Pedal Activity Estimation. **Sensors**, v. 10, n. 4, p. 3798, 2010.

ZHANG, P. Neural Networks for Classification: A Survey. **IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS -PART C: APPLICATIONS AND REVIEWS**, v. 30, n. 4, 2000.

APÊNDICES

APÊNDICE A – Código-fonte do método principal do aplicativo Android

```
public void onClick(View view) {

     switch (view.getId()) {
         case R.id.button:

             lerCSV(); //chamada do metodo que lê o
conteúdo do arquivo .csv

           //início da thread que envia os dados para o
servidor
             Thread t = new Thread() {

                 //@Override
                 public void run() {

                     try {
                         //configurações para comunicação
com o servidor
                         String ip_addr = "187.108.64.177";
                         InetAddress addr =
InetAddress.getByName(ip_addr);
                         int port = 5005;

                         //socket para enviar a posição do
acelerador
                         byte[] pa_socket =
dadosCondutor[0].getBytes();
                         DatagramPacket pkg = new
DatagramPacket(pa_socket, pa_socket.length, addr, port);
                         DatagramSocket ds = new
DatagramSocket();

                         ds.send(pkg);
                         ds.close();
                         Thread.sleep(2000);

                         //socket para enviar a acelaração
```

```
em x
                        byte[] ax_socket =
dadosCondutor[1].getBytes();
                        DatagramPacket pkg1 = new
DatagramPacket(ax_socket, ax_socket.length, addr, port);
                        DatagramSocket ds1 = new
DatagramSocket();

                        ds1.send(pkg1);
                        ds1.close();
                        Thread.sleep(2000);

                        //socket para enviar a acelaração
em y
                        byte[] ay_socket =
dadosCondutor[2].getBytes();
                        DatagramPacket pkg2 = new
DatagramPacket(ay_socket, ay_socket.length, addr, port);
                        DatagramSocket ds2 = new
DatagramSocket();

                        ds2.send(pkg2);
                        ds2.close();
                        Thread.sleep(2000);

                        //socket para enviar a acelaração
em z
                        byte[] az_socket =
dadosCondutor[3].getBytes();
                        DatagramPacket pkg3 = new
DatagramPacket(az_socket, az_socket.length, addr, port);
                        DatagramSocket ds3 = new
DatagramSocket();

                        ds3.send(pkg3);
                        ds3.close();
                        Thread.sleep(2000);

                    } catch (IOException ioe) {
                        ioe.getMessage();
                    }
                    catch(InterruptedException ie){
                        ie.getMessage();
                    }
                    StrictMode.setThreadPolicy(new
StrictMode.ThreadPolicy.Builder().detectDiskReads().detectD
iskWrites().detectNetwork().penaltyLog().build());
```

```
                }

            };
            t.start();

            Toast.makeText(this, "Dados enviados para o
servidor", Toast.LENGTH_SHORT).show();
            break;
    }

 }

//método que acessa o arquivo de logs gerado pelo Torque
 public void lerCSV(){
    try {

        File meuDir =
Environment.getExternalStorageDirectory();

        File arq = new File(meuDir,"torqueTrackLog.csv");

        FileInputStream fis = new FileInputStream(arq);

        // o método carregar obtém os dados do arquivo
.csv e atribui ao vetor de dados do condutor
        //que será enviado para o servidor

        dadosCondutor = Carregar(fis);

    }

    catch(FileNotFoundException ex){
        Log.e("Atenção on click!","Arquivo não
encontrado.");
    }
    catch (IOException e){
        Log.e("Atenção!", "Erro ao carregar o arquivo");
    }
 }

  /*Método para obter os dados do  arquivo .csv do
smatphone*/

 public String[] Carregar(FileInputStream fis) {
```

```
     String vResult[] = new String[16];
     String vAuxStr[] = new String[4];

     try{
         BufferedReader reader = new BufferedReader(new
InputStreamReader(fis));
         StringBuilder sb = new StringBuilder();
         String linha;

         int contador =0;

         String tabela[];

         Double vAux[] = new Double[4];

         String dx;
         double somaAZ = 0.0;
         double somaPosA = 0.0;
         double somaAX = 0.0;
         double somaAY = 0.0;
         int passos = 0; //variável para contar um
intervalo de 60 linhas para calcular a média

         double mediaAZ = 0.0;
         double mediaPosA = 0.0;
         double mediaAX = 0.0;
         double mediaAY = 0.0;


         while((linha= reader.readLine()) != null){
             tabela = linha.split(",");

             for(String cell : tabela){

                 if (!isNumber(cell)){
                     cell = "0.0";
                 }

                 passos++;

                 vResult[contador] = cell;

                 /*Calcula a soma de cada variável para
calcular a média depois*/
                 if (contador == 12){ //coluna da posição
do acelerador
```

```
                    somaPosA +=
Double.parseDouble(vResult[contador]);
                    vAux[0] =
Double.parseDouble(vResult[contador]);
                }
                else if(contador == 13){ //coluna da
aceleração em X
                    somaAX +=
Double.parseDouble(vResult[contador]);
                    vAux[1] =
Double.parseDouble(vResult[contador]);
                }
                else if(contador == 14){
                    somaAY +=
Double.parseDouble(vResult[contador]); //coluna da
aceleração em Y
                    vAux[2] =
Double.parseDouble(vResult[contador]);
                }
                else if(contador == 15){
                    somaAZ +=
Double.parseDouble(vResult[contador]);//coluna da
aceleração em Z
                    vAux[3] =
Double.parseDouble(vResult[contador]);
                }

                contador++;
                if (contador > 15)
                    contador = 0;

                if (passos > 60){
                    mediaPosA = somaPosA /60;
                    mediaAX = somaAX / 60;
                    mediaAY = somaAY / 60;
                    mediaAZ = somaAZ / 60;
                    passos = 0;
                    somaPosA = 0;
                    somaAX =0;
                    somaAY = 0;
                    somaAZ = 0;
                }
            }


        }
```

```
        //armazena as médias das variáveis posição do
acelerador, acelração em x,y e z
        vAux[0] = mediaPosA;
        vAux[1] = mediaAX;
        vAux[2] = mediaAY;
        vAux[3] = mediaAZ;

        //converte os valores obtidos para string
        for (int i = 0; i <=3; i++){
            vAuxStr[i] = String.valueOf(vAux[i]);
        }

        reader.close();
        fis.close();



    }
    catch(FileNotFoundException e){
        Log.e("Atenção!", " Arquivo não encontrado.");
    }
    catch(IOException ex){
        ex.printStackTrace();
    }
    //retorna o vetor com os dados obtidos do condutor
    return vAuxStr;

 }

 //método para testar se o valor de uma célula é numérico
 //para não correr risco de ler as células de título do
arquivo .csv
 public boolean isNumber(String num){
     boolean isNumber;
     try{
         Double.parseDouble(num);
         isNumber = true;
     }
     catch(NumberFormatException e){
         isNumber = false;
     }
     return isNumber;
 }
```

APÊNDICE B – Código-fonte da rotina que recebe os dados do aplicativo Android

```
#!/usr/bin/python

from SOAPpy import SOAPProxy
import socket

#cria um socket para receber os dados
s = socket.socket()
host = "187.108.64.177"
port = 5005

size = 1024
cont = 0
pa1 =""
ax1=""
ay1=""
az1= ""


#prepara-se para receber pacotes UDP
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((host,port))

#recebe os dados do aplicativo Android
while True:

    d = sock.recvfrom(1024)
    data = d[0]

    if cont == 0:
      pa1 = data
      print "Servidor recebeu posicao do acelerador: ",pa1,
"\n"

    if cont == 1:
      ax1 = data
      print "Servidor recebeu aceleracao em x: ", ax1, "\n"

    if cont == 2:
      ay1 = data
      print "Servidor recebeu aceleracao em y: ", ay1, "\n"


    if cont == 3:
```

```
      az1 = data
      print "Servidor recebeu aceleracao em z: ", az1, "\n"

    cont = cont + 1

    if cont > 3:
      cont = 0

      #grava os dados recebidos em um arquivo texto
    file = open('dados_socket.txt', 'w')
    file.write(pa1)
    file.write('\n')
    file.write(ax1)
    file.write('\n')
    file.write(ay1)
    file.write('\n')
    file.write(az1)
    file.close()
```

APÊNDICE C – Código-fonte da rotina que invoca a função de verificação do condutor no Web Service

```
#!/usr/bin/python

from SOAPpy import SOAPProxy

#abre o arquivo que contém os dados recebidos do aplicativo
Android
f = file('dados_socket.txt','r')
f_pa = f.readline()
f_ax = f.readline()
f_ay = f.readline()
f_az = f.readline()
f.close()


#estabelece uma conexão com o servidor que está executando
o WebService
server = SOAPProxy('187.108.64.177:8081')

#executa a função do web service que verifica a validade do
condutor utilizando a rede neural treinada
```

```
#e armazena a resposta na variável result
result = server.verifica(f_pa,f_ax,f_ay,f_az)

#mostra o resultado retornado pelo web service no browser,
no formato HTML
print "Content-type:text/html\r\n\r\n"
print '<html>'
print '<body>'
print '<h1 style=text-align:center;background-
color:lightgray>Provavelmente, o veiculo esta sendo
dirigido por um condutor... </h1>'
for saida in result:
      if saida[0] >= 0.5:
            print '<h3 style=text-align:center;color:green>
Saida: </h3>'
                print '<h3 style=text-
align:center;color:green>',saida[0],'</h3>'
            print '<h1 style=text-align:center;
color:blue>Autorizado</h1>'
      else:
            print '<h3 style=text-
align:center;color:green>',saida[0],'</h3>'
            print '<h1 style=text-
align:center;color:red>Desautorizado</h1>'
print '</body>'
print '</html>'
```

APÊNDICE D – Código-fonte do Web Service que contém a função de verificação do condutor

```
#!/usr/bin/python

from pybrain.datasets import SupervisedDataSet

from pybrain.supervised import BackpropTrainer
from pybrain.tools.shortcuts import buildNetwork
from pybrain.datasets import ClassificationDataSet
from pybrain.tools.customxml.networkreader import
NetworkReader
import pickle
import sys, os, os.path, tty, termios
import funcoes
from SOAPpy import SOAPServer
from array import *
```

```
import socket


#função de verificação do condutor por meio da rede neural
treinada
def verifica(pa,ax,ay,az):
      network =
NetworkReader.readFrom("treinamento_rede_neural");
      trainer =  BackpropTrainer(network,verbose=True);
      dados_entrada = SupervisedDataSet(4,1)
      dados = ([pa,ax,ay,az])
      dados_entrada.addSample(dados,[0]);


      avgErr, outputs, targets =
trainer.testOnData(dados_entrada, verbose=False,
return_outputs_and_targets = True);
      contador = 0;
      acertos = 0;
      erros = 0;

      return outputs

#registro do web service
if __name__ == "__main__":
   server = SOAPServer(('187.108.64.177',8081))

   namespace = 'snafv:namespace'

   #registro da função verifica que permite ser chamada
remotamente
   server.registerFunction(verifica)

   print "Servidor executando...."

   #servidor ouve a porta 8081 contantemente aguardando uma
solicitação à sua função verifica
   server.serve_forever()
```