UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MARCOS TIAGO ARAÚJO DE FRANÇA

APLICAÇÃO DE INTELIGÊNCIA ARTIFICIAL PARA PROTOTIPAÇÃO DE VEÍCULOS NÃO TRIPULADOS DE BAIXO CUSTO

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA

2019

MARCOS TIAGO ARAÚJO DE FRANÇA

APLICAÇÃO DE INTELIGÊNCIA ARTIFICIAL PARA PROTOTIPAÇÃO DE VEÍCULOS NÃO TRIPULADOS DE BAIXO CUSTO

Trabalho de Conclusão de Curso apresentado ao Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Bacharel em Computação".

Orientador: Prof. Dr. Arnaldo Candido Junior

Co-orientador: Prof. Dr. Pedro Luiz de Paula Filho

MEDIANEIRA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso de Ciência da Computação

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

APLICAÇÃO DE INTELIGÊNCIA ARTIFICIAL PARA PROTOTIPAÇÃO DE VEÍCULOS NÃO TRIPULADOS DE BAIXO CUSTO

Por

MARCOS TIAGO ARAÚJO DE FRANÇA

Este Trabalho de Conclusão de Curso foi apresentado às 15:50h do dia 22 de Novembro de 2018 como requisito parcial para a obtenção do título de Bacharel no Curso de Ciência da Computação, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Arnaldo Candido Junior
UTFPR - Câmpus Medianeira

Prof. Pedro Luiz de Paula Filho
UTFPR - Câmpus Medianeira

Prof. Alan Gavioli
UTFPR - Câmpus Medianeira

Prof. Fernando Schutz
UTFPR - Câmpus Medianeira

RESUMO

FRANÇA, Marcos Tiago Araújo. APLICAÇÃO DE INTELIGÊNCIA ARTIFICIAL PARA PROTOTIPAÇÃO DE VEÍCULOS NÃO TRIPULADOS DE BAIXO CUSTO. 65 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

A capacidade das máquinas aprenderem atualmente é aplicada na campo dos veículos autônomos, desenvolvidos por grandes empresas de tecnologia, seja para a locomoção urbana ou para trabalhos em chão de fábrica. Uma das vertentes mais utilizadas para o desenvolvimento desses veículos é o uso das redes neurais convolucionais profundas que representam uma evolução, tendo em vista que são capazes de aprender características de imagens é classificação utilizando-as seja para locomoção do ambiente ou detecção de objetos como um pedestre. Uma das grandes dificuldades para a produção desses equipamentos é o seu alto custo computacional e financeiro, por isso para estre trabalho foi optado por um equipamento de baixo custo objetivando a viabilidade de produção de um veículo autônomo simples, porém funcional. Se utilizando kits como o Raspberry PI e o Kit Lego Mindstorms para construção em conjunto com bibliotecas de processamento de imagem foi possivel montar um conjunto de treinamento primitivo e aplicar estes dados em uma Rede Neural Artificial, com base neste conjunto foi possivel se atingir uma taxa de acuracia otimizada de 40%. Com base nestes resultados e a capacidade do módulo capturar dados é demonstrado o grande potêncial de aplicação a longo prazo.

Palavras-chave: lego mindstorms, redes convolucionais, aprendizagem de máquina,robotica

ABSTRACT

FRANÇA, Marcos Tiago Araújo. THE APPLICATION OF ARTIFICIAL INTELIGENCE FOR LOW COST UNMANNED VEHICLE PROTOTYPING. 65 f. Trabalho de Conclusão de Curso – Curso de Ciência da Computação, Universidade Tecnológica Federal do Paraná. Medianeira, 2019.

The ability of machines to learn today is to apply in the market for autonomous vehicles, by large technology companies, either for urban locomotion or for work on factory floors. One of the most used slopes for the development of the indicators is the use of neural networks that convolve depths that represent an evolution, considering the capacity of expression of the image elements that are used to detect the locomotion or the detection of objects like a pedestrian . One of the greatest difficulties for the production of these equipments is its high computational and financial cost, therefore, the job is a low cost utility, with the feasibility of producing a simple, already functional standalone vehicle. You are using kits such as the Raspberry PI and the Lego Mindstorms Kit for building in conjunction with the image processing libraries, it was possible to install a set of primary exercises and apply this data in an Artificial Neural Network, based on this set. acuracy of 40 %. Based on results, the capacity of the data capture module is greater or greater.

Keywords: lego mindstorms, convolutional network, machine learning,robotic

AGRADECIMENTOS

Agradeço a todos que me acompanharam nessa longa jornada ao longo desses últimos anos me incentivando a todo momento a alcançar os meus objetivos. Assim agradeço a minha família principalmente aos meus pais e meus irmãos que me garantiram a possibilidade de focar integralmente nos meus estudos. Aos meus colegas de turma que caminharam nesta jornada tenho muito a agradecer além do corpo docente em especial a professora Alessandra Hoffman por ter me ajudado nos primeiros passos desta empreitada seguida pelos professores Arnaldo Candido Junior e Pedro Luiz de Paula Filho que me garantiram um suporte sempre me motivando a buscar e desenvolver coisas novas.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

M.A.R.C.	Módulo Autônomo de Reconhecimento Cognitivo
IA	Inteligência Artificial
RNAs	Redes Neurais Artificiais
MLP	Multi-Layer Perceptrons
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
S.C.D.	Sistema de Captura de Dados
RST	Robot Smartphone Tablet

SUMÁRIO

1 INTRODUÇÃO

Na sociedade moderna, a administração do tempo é essencial em várias áreas de atuação. Muitas dessas tarefas são complexas e para a sua automação é necessária a construção de modelos de inteligência similares ao do ser humano, chamados de sistemas inteligentes, que devem ser capazes de executar ações de forma racional e agir com capacidade de aprendizado. Os sistemas inteligentes devem ser capazes de realizar tarefas com o comportamento racional similar ao de um humano, demandando inteligência (MINSKY; PAPERT, 1969), esta área de estudo é conhecida como Inteligência Artificial.

A Inteligência Artificial é dividida em várias subáreas com objetivos distintos, por exemplo, a área de busca que visa o desenvolvimento de algoritmos através do uso de grafos para resolução de problemas, com aplicações em resolução de problemas relacionados a jogos de tabuleiros e otimização em geral. Outro exemplo em evidencia é o uso de lógica difusa, que é baseada em uma extensão da lógica na qual as proposições têm um grau de verdade associado e é utilizada para lidar com vaguezas e incertezas. Uma das subcategorias que mais está em evidência na Inteligência Artificial atualmente, devido a necessidade de automação, é o uso de agentes inteligentes com capacidade de aprendizado de máquina.

Os agentes inteligentes devem ser capazes de perceber o que acontece ao redor no ambiente, que pode ser determinístico ou não, com seus sensores e executar determinadas ações através dos seus atuadores. Um exemplo comum é no campo da robótica, onde com base em dados de sensores de proximidade, um maquinário pode executar determinado movimento através dos seus motores, que neste caso são os atuadores.

Um agente inteligente que pode perceber o que acontece no seu ambiente com base no seu aprendizado tornando-se mais eficiente ao longo do tempo é classificado como um agente com aprendizagem.

O ambiente influencia diretamente na forma que ocorre a execução do agente, se o próximo estado do ambiente for determinado com base no estado atual e as ações já executadas pelo agente este será determinístico, caso isto não ocorra, o mesmo será denominado estocástico ou não determinístico (RUSSEL; NOVIG, 1994), sendo o último o que geralmente ocorre no mundo real devido ao grande número de variáveis. O Aprendizado de Máquina (do inglês

Machine Learning) estuda a capacidade de um programa aprender de sobre um determinado conjunto de dados (PROVOST; KOHAVI, 1998), atualmente o campo de estudo possui diversas abordagens sendo o uso de Redes Neurais Artificiais (RNAs) um dos destaques no meio científico. Outra parte importante da Inteligência Artificial são as técnicas que se utilizam de modelos redes neurais artificiais que alta tem capacidade de aprendizado, garantindo alta velocidade e eficiência.

Entre os avanços que surgiram ao longo dos anos no uso dessas redes neurais artificiais pode-se citar o uso de técnicas de Aprendizado Profundo (do inglês Deep Learning), sendo um modelo de aprendizado de máquina que se inspira no conhecimento do cérebro humano, estatística e matemática aplicada (GOODFELLOW et al., 2016). A técnica abrange várias classes de Redes Neurais Artificiais com focos distintos, por exemplo as Redes Neurais Convolucionais. A popularidade do uso de redes de aprendizado profundo foi possível, pois diferentemente da época do surgimento dos primeiros modelos, houve a melhoria de técnicas para o uso destas redes e a melhoria em quesito de poder computacional.

A utilização de técnicas envolvendo o uso de Aprendizado Profundo está presente no cotidiano, alguns exemplos como assistentes virtuais presentes nos dispositivos móveis como a *Siri*[1], sistemas capazes de executar reconhecimento facial em câmeras fotográficas (ARSENOVIC; MARKO, 2017), sistemas de prevenções de fraude de cartão de crédito (PUMSIRIRATAND; YAN, 2018), entre outros. Uma nova área de aplicação destas técnicas que vem ganhando visibilidade atualmente é a criação de veículos autônomos.

Um veículo autônomo não tripulado pode ser definido como capaz de tomar decisões de direção com segurança de forma que não haja necessidade de intervenção humana (ÖZGÜNER et al., 2007),se utilizando de controladores, dados visuais e sensores, que recolhem dados do ambiente sendo estes processados em tempo real, assim navegando de forma segura. Porém, infelizmente, a criação destes protótipos envolve alto custo seja financeiro ou computacional. Os veículos de transporte autônomos atualmente utilizam-se de técnicas de aprendizado profundo envolvendo redes neurais convolucionais para o funcionamento. Redes convolucionais usam o reconhecimento de imagem, para a visualização do ambiente em relação ao veículo. Assim este trabalho visou o estudo de aplicações de técnicas de Aprendizado Profundo com o uso de redes convolucionais em conjunto com um hardware de baixo custo para a criação de protótipo de veículo autônomo viável com alto grau de segurança considerando que devido ao seu tamanho e a forma de sua construção o risco para acidentes é mínimo. Os resultados apresentados por este trabalho podem ser usados para definir os parâmetros necessários para a criação de veículos autônomos de diferentes classes e finalidades e encorajar

[1] https://www.wired.com/story/how-apple-finally-made-siri-sound-more-human/

maiores estudos na área, considerando que a maioria dos trabalhos existentes neste nicho são de tecnologia proprietária.

1.1 OBJETIVOS GERAL E ESPECÍFICOS

Realizar a montagem de um conjunto de treinamento que será aplicado em uma Rede Neural Convolucional, em seguida verificar a qualidade dos resultados. A montagem do conjunto de treinamento e a validação de resultados será feita com o auxílio de um modulo robótico de acordo com as condições referentes ao tempo para o desenvolvimento do trabalho. Para o experimento será utilizado o Módulo Autônomo de Reconhecimento Cognitivo (M.A.R.C.) (FRANÇA; FRANÇA, 2018), um protótipo de robô de baixo custo construído na plataforma Lego Mindstorms, um ambiente comportado será utilizado para ambas as etapas, além de uma área reservada jamais visitada, assim verificando a capacidade de generalização da rede. Com base nos resultados deve ser estabelecido padrões mínimos de qualidade para se obter um resultado positivo.

- Montar uma base de dados com auxílio de um modulo robótico;
- Realizar o treinamento desta base de dados em uma Rede Convolucional;
- Analisar a eficácia do treinamento realizado;
- Realização de melhorias em hardware de acordo com as necessidades para aplicação do modulo robótico.

1.2 JUSTIFICATIVA

A busca de automatização de tarefas avançadas, com objetivo de se evitar falhas humanas. Os modelos de locomoção autônomos como carros, ônibus, maquinários, são bastante visados no campo cientifico para buscar melhorias na automação e/ou mobilidade de forma geral, considerando que falhas humanas podem levar a acidentes ou mortes. Um empecilho para a produção em larga escala desses equipamentos de é que se mostram demasiado caros em sua concepção, por isso se faz necessário a elaboração de técnicas para baratear o processo

de construção, possibilitando a elaboração protótipos eficientes. Assim propomos elaborar um protótipo um veículo autônomo de forma eficiente e de baixo custo em conjunto com técnicas de IA. O uso de Técnicas de Deep Learning no processo se mostra necessário, levando em consideração que os modelos existentes no mercado se utilizam da mesma devido a sua capacidade de aprendizado e generalização. Apesar de atualmente existirem alguns protótipos no mercado como citado anteriormente, a grande maioria são de tecnologia proprietária (Google, Tesla, Uber, Apple), além de existir pouca informação de domínio público.

1.3 ORGANIZAÇÃO DO DOCUMENTO

Este trabalho é organizado como segue. O Capítulo 2 apresenta a fundamentação sobre redes neurais artificiais e discute trabalhos sobre veículos autônomos e quais as tecnologias utilizadas. O Capítulo 3 discute as metodologias que serão utilizadas no desenvolvimento do experimento. No capitulo 4 são discutidos os resultados com base em todas as etapas de experimentação. Para a finalização do trabalho são tiradas as devidas conclusões com base nas etapas de desenvolvimento e experimentação. Uma seção extra foi adicionada referente a trabalhos futuros considerando o potencial observado durante as etapas anteriores.

2 REFERENCIAL TEÓRICO

Será apresentado no referencial teórico um aprofundamento sobre tipos de aprendizado e a forma como eles são classificados, o uso de redes neurais artificiais para aprendizado de máquina, como sua história, características e subdivisões. O foco sobre a área de redes neurais artificiais será o uso de Redes Neurais Convolucionais considerando que estas são responsáveis por aprendizado com imagem. Com base nesses conhecimentos serão apresentados trabalhos que se utilizam de redes neurais artificiais com aplicabilidade em construção de veículos autônomos.

2.1 APRENDIZADO DE MÁQUINA

Aprendizado de máquina é definido como a capacidade de melhoria no desempenho de um programa durante a realização de alguma tarefa, por meio da experiência (MITCHELL, 1997). Este aprendizado é feito de forma indutiva, na qual, com base em determinado número de informações, é possível se concluir uma determinada hipótese.

Segundo Simon (2013), especialista na área de Inteligência Artificial com aplicabilidades nos jogos define como aprendizado de máquina "o campo de estudos que fornece a computadores a habilidade de aprenderem sem serem explicitamente programados". Segundo Richards et al. (2014) o aprendizado de máquina é considerado um ramo importante dada área de Inteligência Artificial, com especialização no estudo e concepção de sistemas inteligentes que tenham capacidade de aprendizagem com base em uma determinada entrada de dados.

O aprendizado indutivo pode ser dividido em três categorias, sendo elas o aprendizado supervisionado (preditivo), não supervisionado (descritivo) e o aprendizado por reforço (BATISTA, 2003). Existem alguns métodos para aprendizado, por exemplo, o aprendizado supervisionado pode ser feito através de métodos de classificação ou regressão. Enquanto o

aprendizado não supervisionado pode ser feito através de métodos de agrupamento, associação e sumarização, entre outras (BATISTA, 2003). Conforme pode ser visto na Figura 1 a hierarquia de aprendizado indutivo detalhada. Dentre as muitas técnicas de aprendizado de máquina, neste trabalho o foco principal foi o uso do Aprendizado Preditivo através da classificação, os motivos serão explicados mais à frente.

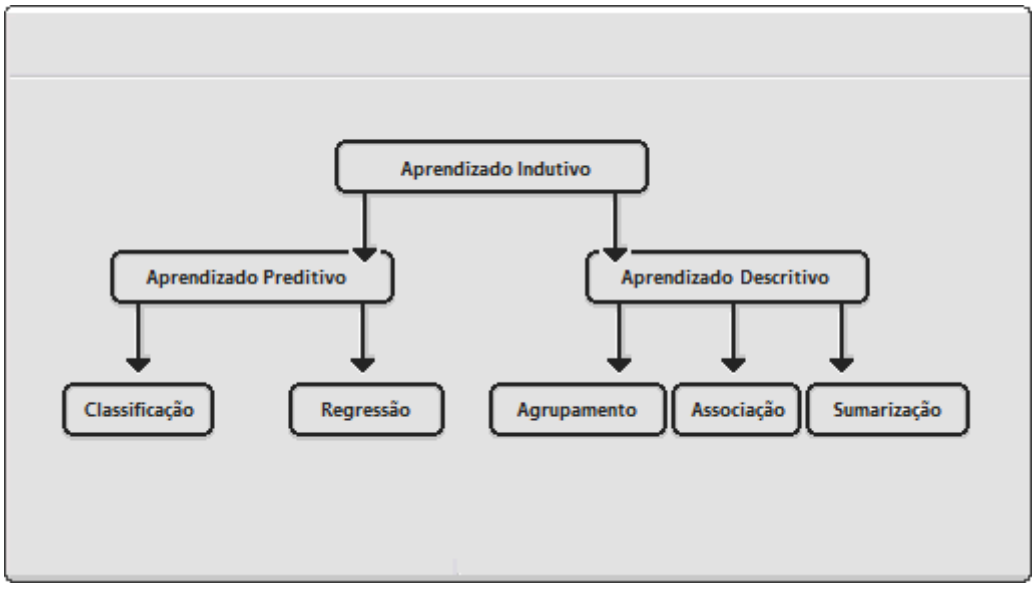

Figura 1 – Hierarquia de Aprendizado.
Fonte: Adaptado de Batista (2003)

2.2 REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais (RNAs) surgiram com a necessidade de resolução de problemas complexos que programas comuns não conseguiram resolver. Assim como outros inventos estes foram baseados em elementos presentes na natureza, a construção das redes neurais artificiais foi baseada no funcionamento do cérebro humano e seus neurônios biológicos.

> ”Uma rede neural artificial pode ser definida como processadores maciçamente paralelamente distribuídos constituído de unidades de processamento simples, que tem a propensão natural de armazenar conhecimento experimental e torna-lo disponível para uso (HAYKIN, 2001).”

2.2.1 Perceptron

O primeiro modelo de aprendizagem foi proposto por Donald Hebb baseado no comportamento do cérebro humano, no qual se os axônios de dois neurônios estão próximos o bastante e ocorre a coativação repetidamente, gerando um aumento das conexões sinápticas (HEBB, 1949). Com base na ideia de aprendizagem, os primeiros modelos de RNAs eficientes surgiram em meados dos anos 50, o perceptron, proposto por (ROSENBLATT, 1958). Sendo um classificador do tipo discriminador linear, ou seja, só consegue classificar padrões que sejam linearmente separáveis (HAYKIN, 2009).

O modelo foi criado para o reconhecimento de padrões e classificação, por exemplo, determinar se uma mensagem de e-mail é spam ou legitima. Porém, devido à complexidade dos problemas do mundo real, o uso do Perceptron para classificação se mostra ineficiente. Assim, para a resolução de diversos problemas complexos referentes ao mundo real, é necessário o uso de modelos mais eficientes como o MLP.

Na Figura 2(a)e Figura 2(b) possível verificar um problema de classificação com duas classes (a positiva e a negativa), onde na Figura 2(a) é mostrada uma divisão sendo feita através de uma linha reta, porém na Figura 2(b) não possível.

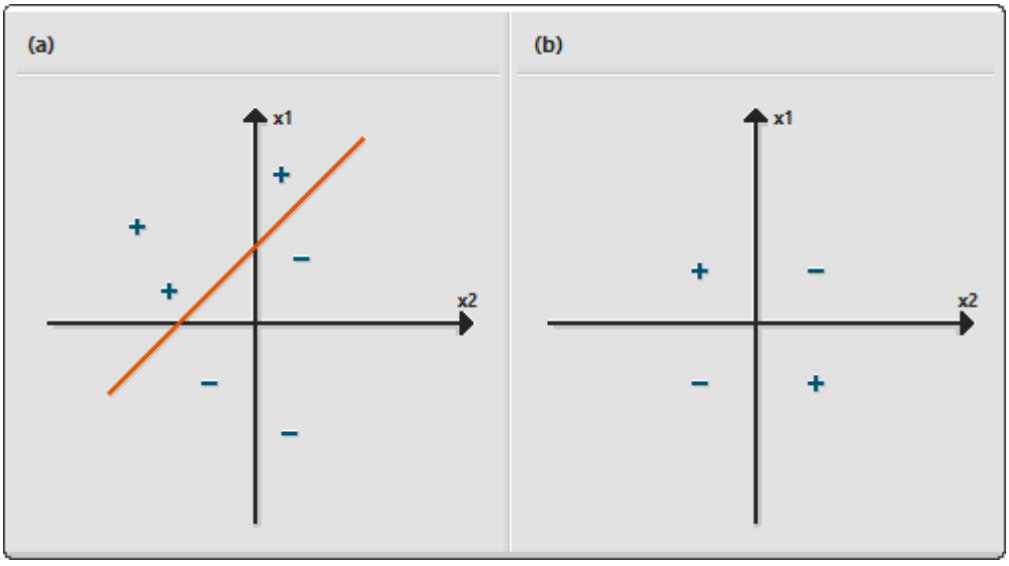

Figura 2 – Exemplo de Problemas: (a) Linearmente Separável (a); (b) Linearmente Inseparável.
Fonte: Adaptado de Mitchell (1997)

A Figura 3 demonstra a representação do Gráfico de Fluxo do Perceptron de uma única camada, onde de acordo com Mitchell (1997):

- **X1,X2 ... Xn:** Representa os atributos de entrada do neurônio;
- **W1,W2 ... Wn:** Representa os pesos da camada do neurônio;
- **Bias** (θ)**.:** Representa o Limiar de Ativação que será abordada mais à frente;
- **v:** Representa a soma ponderada das entradas por seus respectivos pesos. (Equação 1);
- **y(v):** Representa a função de ativação do neurônio;
- **z:** Sinal de Saída do Neurônio.

$$[H]v = \sum_{j=1}^{m} x_j w_j + \theta \tag{1}$$

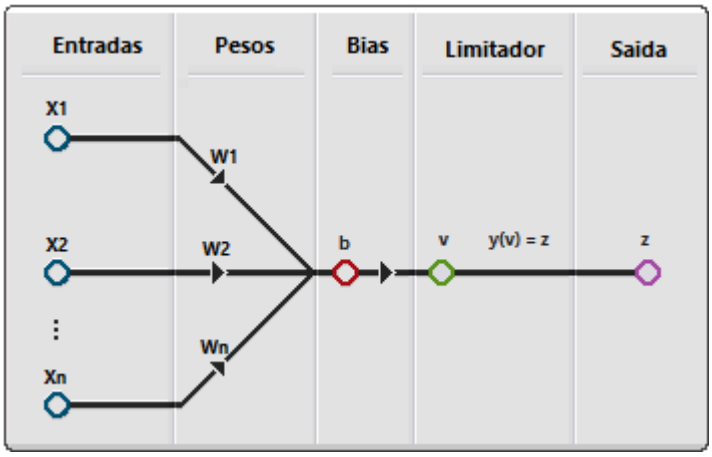

Figura 3 – Representação de um Perceptron de única camada.
Fonte: Adaptado de Mitchell (1997)

A função de ativação é responsável por limitar a saída do perceptron em um intervalo. Existem várias funções de ativação que são utilizadas de acordo a estrutura da rede, por exemplo a Função de Ativação Limiar (do Inglês Threshold), sendo g(x) representada pela equação 2 e parcialmente diferenciável, ou seja, não possuindo derivadas de primeira ordem. Se a saída do neurônio for positiva este será ativado, caso o contrário não (HAYKIN, 2009).

$$y = g(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{senão} \end{cases} \tag{2}$$

Segue na Figura 4 a representação gráfica da aplicação da Função de Ativação Limiar g(x) no intervalo de ativação [0,1]:

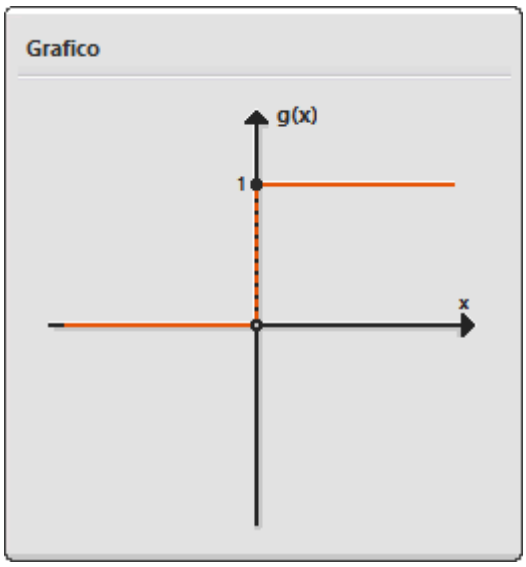

Figura 4 – Gráfico representativo da Função de Ativação Limiar.
Fonte: Autoria Própria

Para a execução do Modelo Perceptron foi construído o primeiro neuro-computador do mundo o Mark 1 Perceptron (Figura 5) na Universidade de Cornell entre os anos de 1957 e 1959, com o objetivo de reconhecimento de imagens.

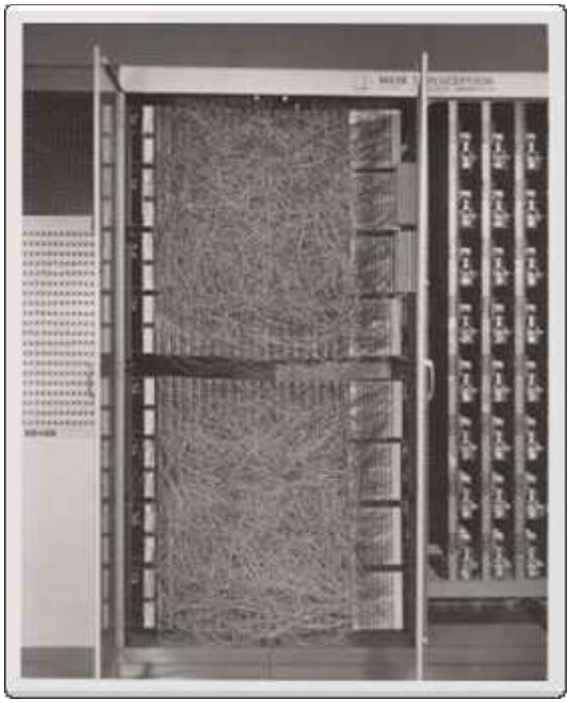

Figura 5 – O *Mark 1 Perceptron*.
Fonte: https://digital.library.cornell.edu/catalog/ss:550351

Porém, devido ao pouco entendimento do potencial dos modelos, o baixo poder computacional da época e a insuficiência de dados para aprendizado, as pesquisas na área estagnaram.

2.2.2 Perceptron de Múltiplas Camadas

O Perceptron de Múltiplas Camadas (MLP), (do Inglês *Multi-Layer Perceptrons*) foi um modelo de RNA proposto por Minsky e Papert com a publicação do seu Livro Perceptrons (MINSKY; PAPERT, 1969). Este modelo é superior ao Perceptron, pois é capaz de resolver problemas não linearmente separáveis, possuindo uma ou mais camadas escondidas onde o sinal e propagados para frente (Figura 6).

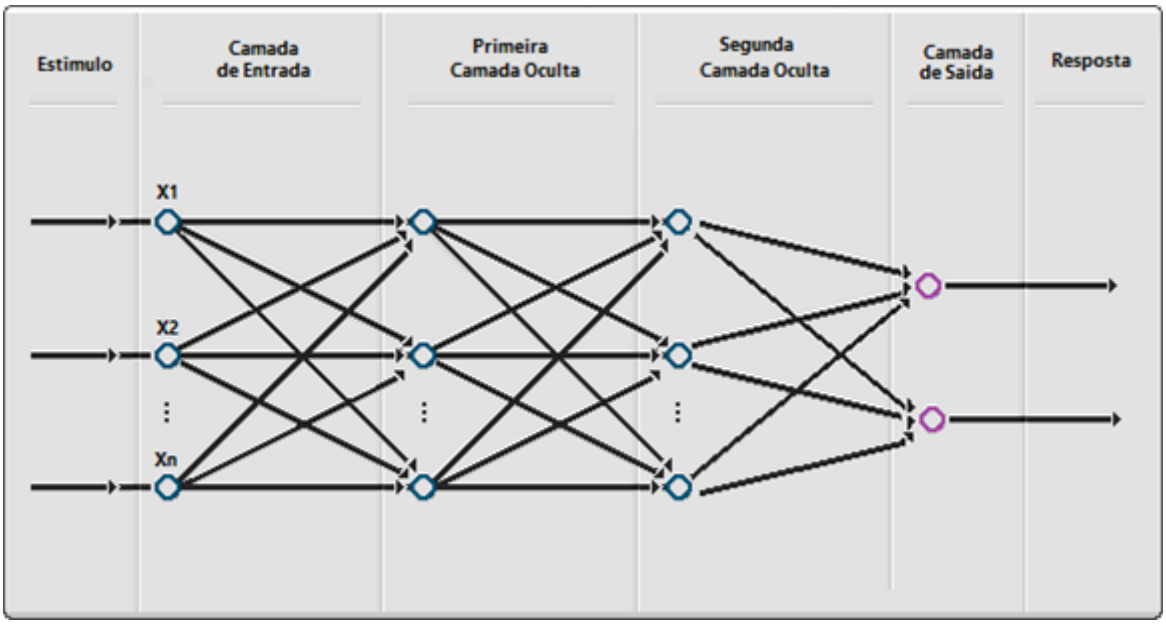

Figura 6 – Representação de uma rede de múltiplas camadas (MLP) com duas camadas ocultas.
Fonte: Adaptado de Mitchell (1997)

O Ajuste de parâmetros para o treinamento de uma rede MLP se mostra essencial na busca do resultado desejado, devendo ser ajustado manualmente até que os objetivos sejam atingidos. Os parâmetros principais para ajuste são a Taxa de Aprendizado, Momento e Erro (MITCHELL, 1997). A variedade de dados também se mostra bastante importante pois evita alguns vícios na sua rede como o Overfitting e Underfitting. O Overffiting consiste na

dificuldade da rede neural artificial generalizar, ou seja, o seu resultando esta tendendo a uma classificação, esta situação ocorre geralmente quando os dados estão viciados. Enquanto o Underfitting se reflete em uma taxa de erro considerável mesmo quando ocorre o teste no seu próprio conjunto de treinamento, podendo ser corrigido com uma alteração nos parâmetros para busca em uma taxa de erro inferior.

Um ponto importante a se levar em consideração é que as redes MLPs, diferentemente do Perceptron de uma única camada, necessitam de funções de ativação mais sofisticadas considerando a capacidade de criação de classificadores multiclasse (em vez de binários). Estas são funções de ativação preferencialmente diferenciáveis, ou seja, funções que possuem derivadas de primeira ordem para todos os pontos de seu domínio (HAYKIN, 2009).

Alguns exemplos de função de ativação diferenciáveis são a Sigmoide Logística (Equação 3) e Tangente Hiperbólica (Equação 4). A função Sigmoide Logística $s(u)$. esta já foi a mais utilizada para os treinamentos de RNAs devido seu intervalo real entre 0 e 1, ou seja, o neurônio ativava ou não. A função Tangente Hiperbólica $tanh(u)$. que atualmente se mostra mais capaz do que a Sigmoide Logística, esta possui um intervalo entre -1 e 1.

$$y = s(u) = \frac{1}{1 + e^{-au_j(n)}}, \qquad a > 0 \tag{3}$$

$$y = tanh(u) = \frac{e^u - e^u}{e^u + e^u} \tag{4}$$

Segue abaixo a representação gráfica das funções de ativação Sigmoide Logística (Figura 7a) e Tangente Hiperbólica (Figura 7b):

No ano de 1986, surgiu o algoritmo Back-Propagation que permitia o treino supervisionado de neurônios artificiais com a utilização de um Modelo de Múltiplas Camadas proposto por Minsky e Papert. Assim, as pesquisas na área sofreram grandes avanços.

O problema inicial para a utilização de redes MLP envolvia a concepção de um algoritimo de treinamento viavel , assim no ano de 1986 surgiu o algoritmo Back-Propagation que permitia o treino supervisionado de neurônios artificiais com a utilização de RNAs do tipo MLP (RUMELHART et al., 1986).

O treinamento de redes MLP é feito através do uso do algoritmo Back-Propagation, sendo comumente utilizado para o treino de Redes Profundas (do Inglês Deep Networks), uma variação de rede de múltiplas camadas que se mostra entre as mais poderosas atualmente, estes tópicos e suas características serão abordadas na subseção 2.2.4.

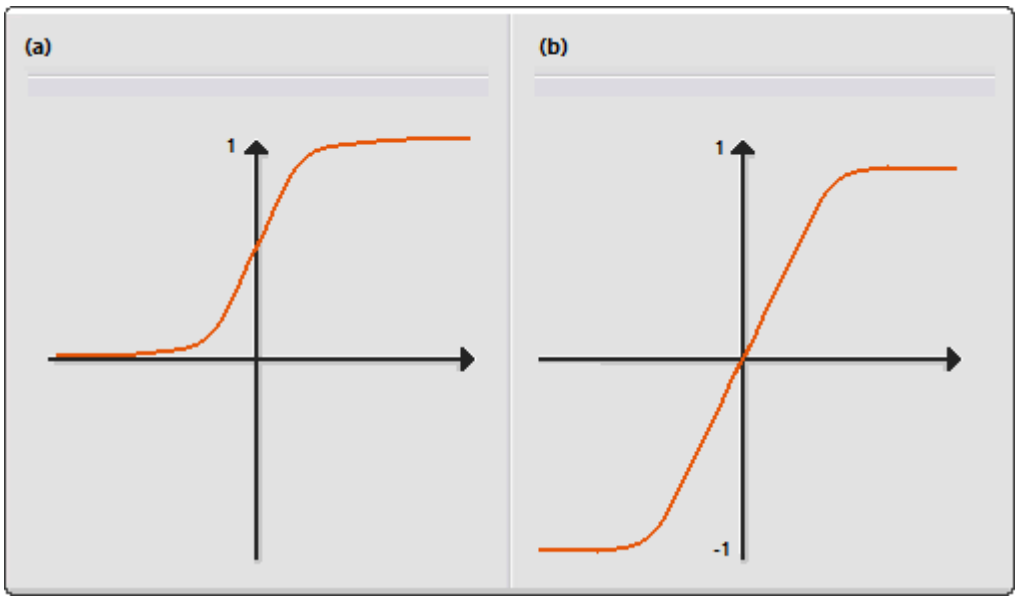

Figura 7 – Representação gráfica: (a) Sigmoide Logística (a); (b) Tangente Hiperbólica.
Fonte: Autoria Própria

2.2.3 Arquiteturas de Rede

Tanto o Perceptron e o MLP possuem uma camada de entrada e uma de saída, no caso do MLP possuindo ainda suas camadas escondidas. Existem várias arquiteturas de rede referentes a forma como ocorre a conexão destas camadas, por exemplo a arquitetura Feedfoward de camada simples (Perceptron), arquitetura Feedfoward de múltiplas camadas (MLP) e redes recorrentes (HAYKIN, 2001).

- **Feedforward de uma única camada:**
  Possuindo apenas uma camada de entrada e camada de saída, a conexão no qual corre o fluxo de informações entre as camadas segue uma única direção, sempre partindo da camada de entrada até a camada de saída.

- **Feedfoward de múltiplas camadas (MLP):**
  Possuindo também uma camada de entrada e camada de saída, porém além destas possui uma ou mais camadas escondidas. O fluxo de informações entre as camadas segue uma única direção, partindo da camada de entrada, navegando entre as camadas escondidas até a camada de saída. De acordo com Haykin (2001) existem 3 sub-divisões para esta arquitetura que divergem, nas quais ocorrem estas conexões podendo sendo estas completamente conectada onde todos os neurônios estão conectados seja na camada

anterior ou na seguinte, parcialmente conectada quando apenas alguns desses neurônios estão conectados na camada anterior ou na seguinte ou localmente conectada onde estes neurônios conectados pertencem a uma determinada região.

- **Redes Recorrentes:**
  Nas redes recorrentes o sinal de saída de um neurônio é realimentado na rede como se fosse um sinal de entrada, assim podendo armazenar informações para um novo processamento sendo ideais para o uso de dados de série temporal. Atualmente as redes profundas que serão abordadas mais à frente se utilizam de arquiteturas recorrentes.

2.2.4 Algoritmo de Treinamento Backpropagation

Algoritmo proposto por Rumelhart et al. (1986), é o método de treinamento de redes MLP mais utilizado (HAYKIN, 2001). Funcionando com ajustes de pesos, este método consegue solucionar o problema XOR (Ou-Exclusivo) (BORCHARDT, 2013), no qual busca encontrar o mínimo global na superfície com o objetivo dos valores dos pesos ideais para minimizar o erro da rede como um todo. Esses ajustes de pesos podem ser feitos de duas formas:

- **Online:** o método padrão de ajuste de pesos, onde a atualização deles é feita após cada exemplo ser apresentado à MLP;

- **Lote:** método onde a atualização é realizada após todos os exemplos do conjunto de treinamento serem devidamente apresentados à MLP.

Sendo o Backpropagtion um algorítimo elaborado para um treinamento supervisionado este é composto por duas fases, a fase Forward e a fase Backward. De acordo com Silva e Flauzino (2010):

- **Forward:** as entradas são inseridas na rede e em seguida ocorre a propagação do erro ao longo das camadas escondidas até a última camada gerando as suas respectivas saídas. Nesta etapa os pesos se mantêm inalterados;

- **Backward:** ocorre a propagação dos erros no sentido contrário com base naqueles encontrados na fase Foward, ou seja, da camada de saída até a camada de entrada. Assim, diminui-se a cada interação, a soma dos erros entra a resposta desejada e a resposta obtida.

Como citado anteriormente a aplicação do algoritimo backpropagation, exige que a função de ativação seja diferenciável, de modo que possa ser calculada a derivada parcial do erro em relação a um dado peso $w_i j$. Outros elementos a serem considerados é a Taxa de Erro(E), a saida do neurônio oj após a função de ativação e finalizando com a saída de rede (netj) que representa o potêncial de ativação em conjunto com o bias,como é descrito na Equação 5:

$$\frac{\partial E}{\partial W_{i,j}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \frac{\partial net\, j}{\partial W_{i,j}} \tag{5}$$

2.3 REDES PROFUNDAS

Um tipo de rede MLP de destaque é a arquitetura de rede profunda onde é feito o uso das camadas intermediarias de processamento de informação (DENG, 2014), sendo estas camadas intermediárias bastantes utilizadas (TUSHAR, 2015). Algumas variações de redes profundas são bastante conhecidas como as Redes Convolucionais que são amplamente utilizadas na área de visão computacional e serão abordadas na próxima seção.

2.3.1 Funções de Ativação para Redes Profundas

O aprendizado das Redes Profundas é feito normalmente de forma supervisionada, o algoritmo recebe um conjunto de características para responsável por classificar as entradas. Entre suas funções de ativação têm-se as funções ReLu e a Softmax (GOLDBERG, 2015).

A função ReLu é uma função não linear. Podendo ser calculada através da Equação 6. Caso a saída seja um valor positivo, o neurônio é ativado e, caso contrário, não, possuindo um intervalo entre [0, +inf], conforme pode ser visto na representação gráfica da Figura 8, podendo

facilmente ter ativações elevadas.

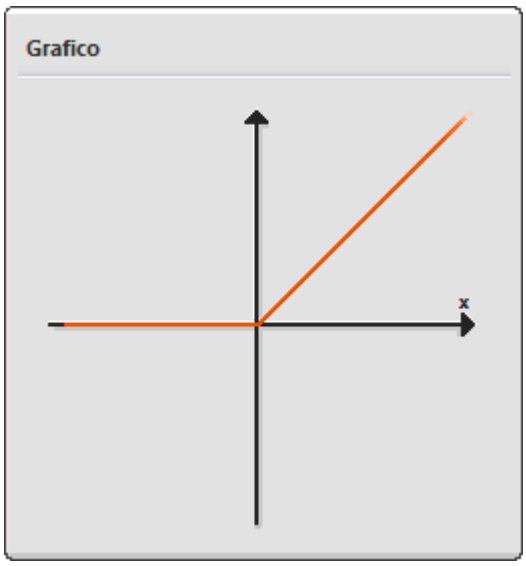

Figura 8 – Gráfico representativo da função de ativação ReLu.

Fonte: Autoria Própria

$$A(x) = max(0, x) \tag{6}$$

A função Softmax (Equação 7), assim como a Sigmoide, resulta em uma saída entre 0 e 1, porém no caso da Softmax é feita a divisão proporcional de todas as saídas de forma que a somatória das mesmas deve ser igual a 1, como pode ser mostrado na Figura 9.

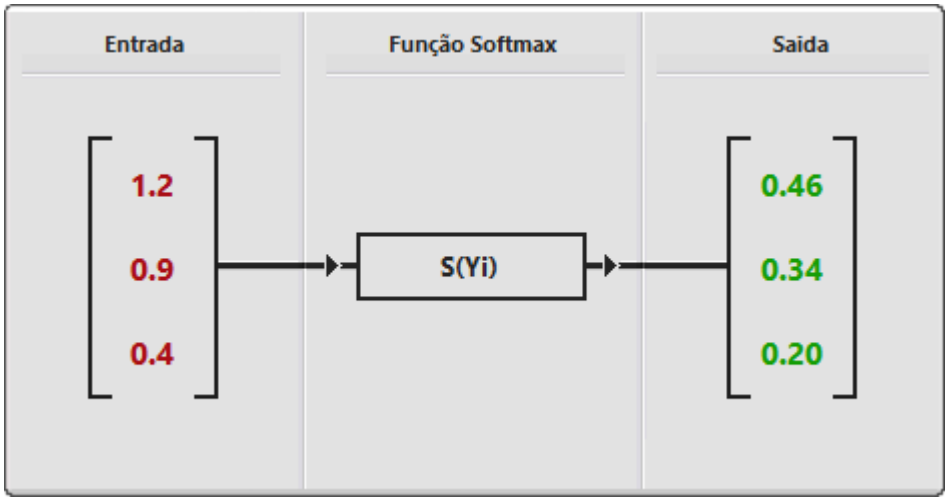

Figura 9 – Exemplo de aplicação da função Softmax sobre um conjunto de valores.

Fonte: Autoria Própria

$$S(Yi) = \frac{e^{Y_i}}{\sum e^{Y_i}} \tag{7}$$

2.3.2 Redes Neurais Convolucionais

Sendo uma variação de rede de Perceptrons de Múltiplas Camadas baseada no processo biológico no qual os seres vivos processam dados visuais.

Essas redes atualmente se mostram entre as mais poderosas no quesito qualidade, alguns exemplos como a Alexnet (KRIZHEVSKY et al., 2012) e GoogLeNet (SZEGEDY et al., 2014) ambas vencedoras do Concurso Imagenet, atingindo níveis elevados de acerto. Porém para atingir este resultado essas redes exigem grande poder de processamento para um treinamento adequado e grandes conjuntos de dados.

Uma rede convolucional é organizada em camadas, onde os dados de entrada serão processados ao longo dessas camadas (Figura 10). As camadas convolucionais da rede são responsáveis por extrair as características destas entradas, após cada camada de convolução vem as camadas de Pooling que tem a função de reduzir a dimensão desta entrada.

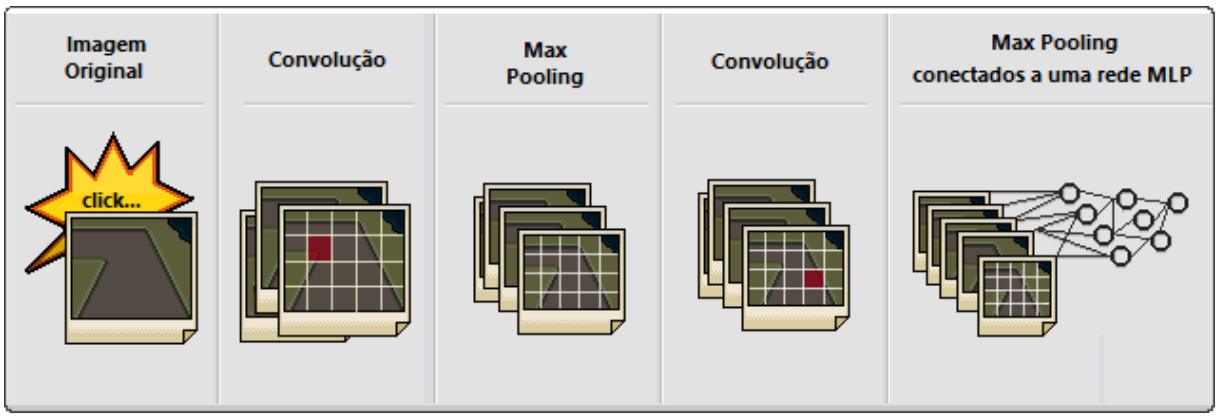

Figura 10 – Exemplo sequencial de entrada em uma rede convolucional
Fonte: Adaptado de Lecun et al. (1998)

A camada convolutiva é composta pelos filtros (NIELSEN, 2018), estes que são matrizes tridimensionais dimensionáveis e têm a responsabilidade de extrair as características de uma imagem, percorrendo a mesma como um todo ao longo do treinamento extraindo seus respectivos pesos (Figura 11).

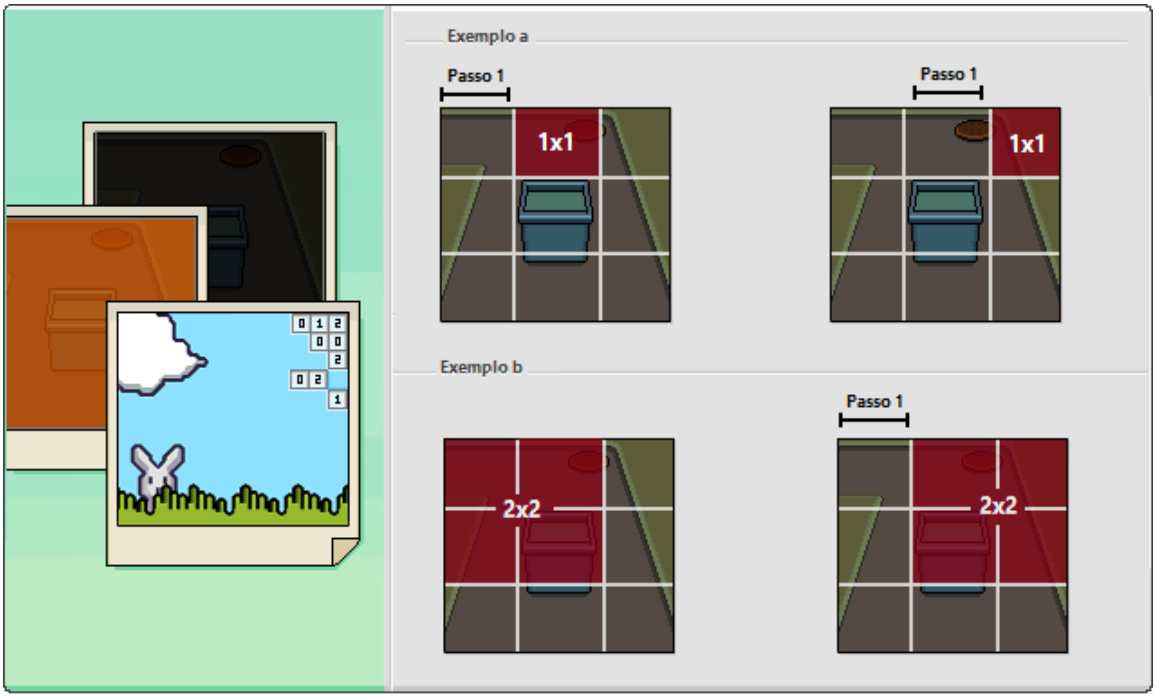

Figura 11 – Exemplo de Funcionamento do deslocamento de um Filtro.(a) Um Filtro 1x1 com passo 1 em 2 etapas sucessivas de convolução; (b)Um Filtro 2x2 com passo 1 em 1 etapa de convolução

Fonte: Autoria Própria

Para este aprendizado é necessário definir o passo, este representa o numero de casas entre as interações de filtro. No exemplo da Figura 11 é possível verificar que independente do tamanho do filtro, o passo respeitou corretamente os limites da imagem. Existem casos nos quais, ao deslocar o filtro, dependendo do passo este ultrapassa as dimensões da imagem, para se resolver é possível utilizar a técnica *Zero-Padding* (ZHAI et al., 2016). A técnica consiste em adicionar zeros a borda da imagem assim criando novas linhas e colunas de forma que o deslizamento seja possível como pode ser visto na Figura 12.

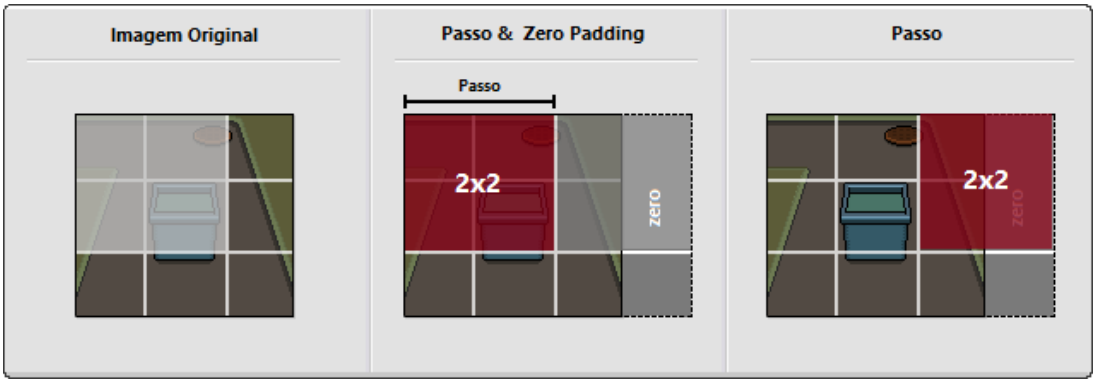

Figura 12 – Aplicação da Técnica *Zero-Padding* em uma imagem 3x3 com Filtro 2x2 e passo 2.

Fonte: Autoria Própria

A partir do filtro é possível se extrair os pesos das características da imagem toda vez que o mesmo é deslizado dependendo do passo como pode ser visto na Figura 13.

Figura 13 – Exemplo de Extração de Pesos de um pedaço da imagem com um filtro 1x1
Fonte: Autoria Própria

A uso da camada de Pooling evita desperdícios de processamento reduzindo a dimensionalidade da imagem. A forma mais comum de Pooling consiste em substituir os valores de uma região pelo valor máximo (Figura 14) (GOODFELLOW et al., 2016).

Figura 14 – O uso do algoritimo de Max-Pooling nos pesos de uma imagem
Fonte: Autoria Própria

2.3.3 Competição Imagenet

Desde do ano de 2010, com o objetivo de avaliar a capacidade de aprendizado de

modelos neurais foi proposto o Desafio de Reconhecimento Visual de Grande Escala do ImageNet (do Inglês *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC)), onde são avaliados algoritmos para detecção de objetos e classificação de imagens em larga escala. Permitindo por parte dos pesquisadores a comparação de resultado em relação ao progresso da visão computacional para indexação de imagens em grande escala para recuperação e anotação e compartilhamento de bases de dados (Stanford Vision Lab, 2015).

O concurso imagenet trouxe para a área de visão computacional modelos com taxas impressionantes de aprendizado como a Rede Alexnet, GoogLeNet e VGGNet. Sendo alguns destes modelos propostos por grandes empresas no ramo de tecnológia como a Microsoft e Google (Stanford Vision Lab, 2015).

2.3.4 Rede Alexnet

Uma rede convolucional que foi vencedora do concurso Imagenet no ano de 2012, possuindo uma arquitetura semelhante à rede LeNet (LECUN et al., 1989), porém tendo maior profundidade, a arquitetura da Alexnet engloba oito camadas como pode ser visto na Figura 15.

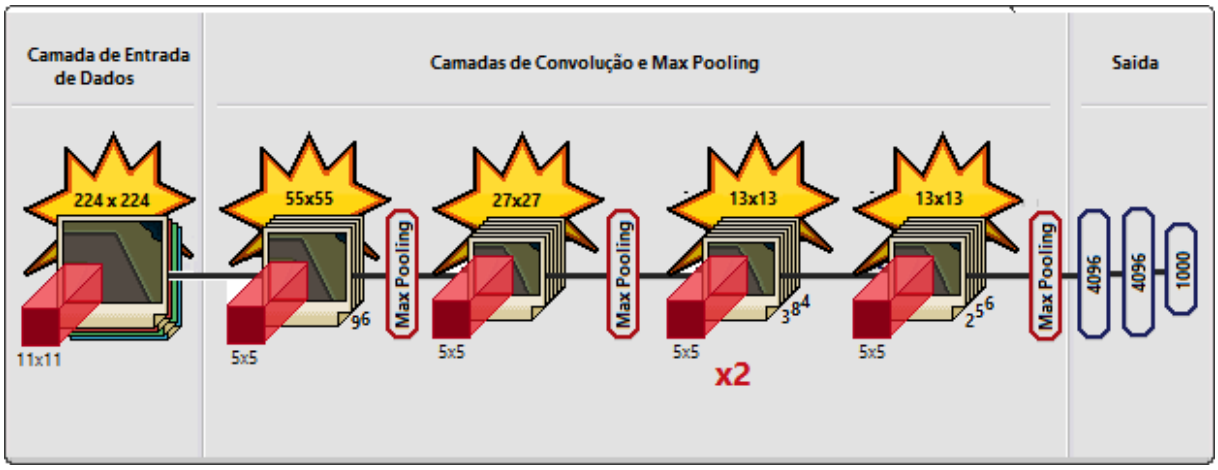

Figura 15 – Topologia da Alexnet
Fonte: Adaptado de (SILVA, 2017)

Destas oito camadas, cinco são de camadas de convolução, algumas seguidas por camadas de Max-Pooling e por fim três camadas completamente conectadas no final da rede incluindo uma camada Softmax com 1.000 saídas, sendo estas respectivas a cada uma das classes, conforme e descrito abaixo:

- **Primeira Camada Convolutiva:** os dados de entrada estão em formato RGB onde é carregada uma imagem 224x224 pixels de dimensão, onde estão são tratadas separadamente possuindo um Filtro de 11x11 e um passo de 4 para extração de características assim gerando 96 saídas com resolução 55x55;
- **Segunda Camada Convolutiva:** as 96 saídas de características são analidades com um Filtro de 5x5, resultando em 256 novas saídas de características e estes passam por um Max-Pooling em sequência;
- **Terceira Camada Convolutiva:** as 256 saídas de características são analisadas com um Filtro de 5x5 resultando em 384 novas saídas de características e estes passam por um Max-Pooling em sequência.;
- **Quarta e Quinta Camada Convolutivas:** as 384 saídas da camada anterior são analisadas com um Filtro de 5x5 resultando em 384 saídas, em seguida, esses dados entram na na quinta que possui os mesmos parâmetros da camada anterior com um filtro 5x5, resultando em outras 384 saídas de características.
- **Sexta Camada Convolutiva:** as 384 saídas de características da quinta camada são analisadas com um Filtro de 5x5 resultando em 256 novas saídas de características e estes passam por um Max-Pooling em sequência;
- **Camadas Completamente Conectadas:** após os dados passarem pelo Max-Pooling da sexta camada, são utilizados nas completamente conectadas com 4.096 entradas seguida por uma camada de Softmax com 4.096 entradas resultando no total de 1.000 saídas.

Figura 16 – Alguns exemplos de classificação com uso da Alexnet

Fonte: Adaptado de (KRIZHEVSKY et al., 2012)

A Alexnet apresentou bons resultados com uma taxa de erro no TOP 5 reduzida a

apenas 15.3% (KRIZHEVSKY et al., 2012), exemplos de classificação que foram executados pela Rede podem ser vistos na Figura 16. Nos exemplos de classificações vistos na Figura 16, os valores presentes na taxa de classificação de cor vermelha representam que aquela classificação corresponde a real, enquanto os valores em azul representa outras classificações.

Um exemplo de classificação que vale ressaltar é referente a imagem Cereja, no qual a rede a classificou como um Dalmata, não estando completamente errada se levada em consideração a imagem.

2.3.5 Rede GoogLeNet

RNA que foi vencedora do concurso Imagenet no ano de 2014 com uma taxa de erro de classificação de apenas 0.06656, cuja a Topologia Formada por 22 camadas (Figura 17), algumas seguidas de camada de Pooling, o seu grande diferencial e o uso de Inceptions.

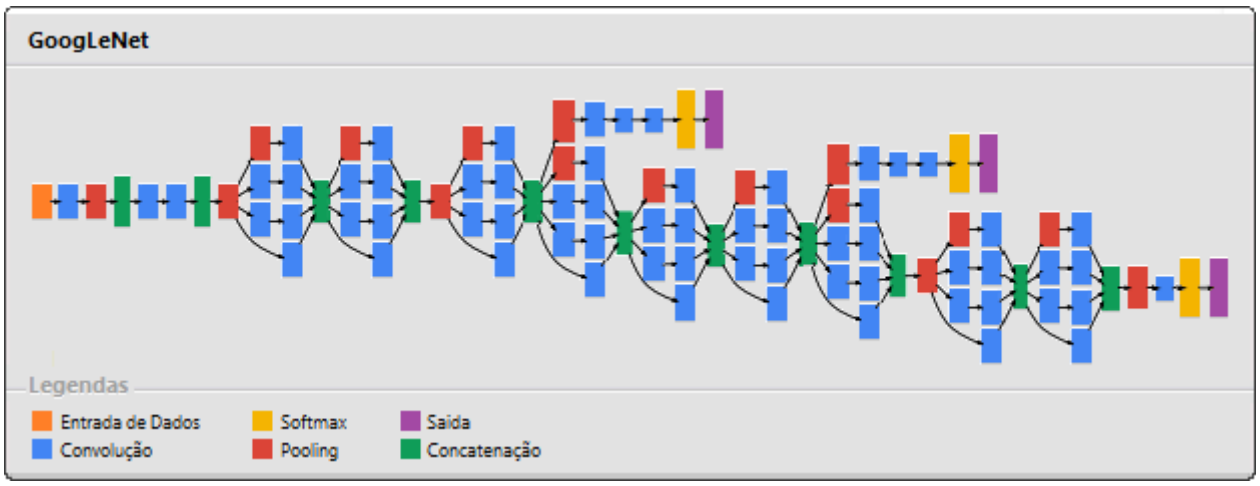

Figura 17 – Topologia GoogLeNet
Fonte: Adaptado de (SZEGEDY et al., 2014)

O modulo de Inception é formado por blocos de convoluções com filtros 1x1,3x3 e 5x5 que funcionam em paralelo como pode ser visto na Figura 18. Em alguns casos o Modelo Inception da Googlenet segue com uma camada de Max-Pooling 3x3.

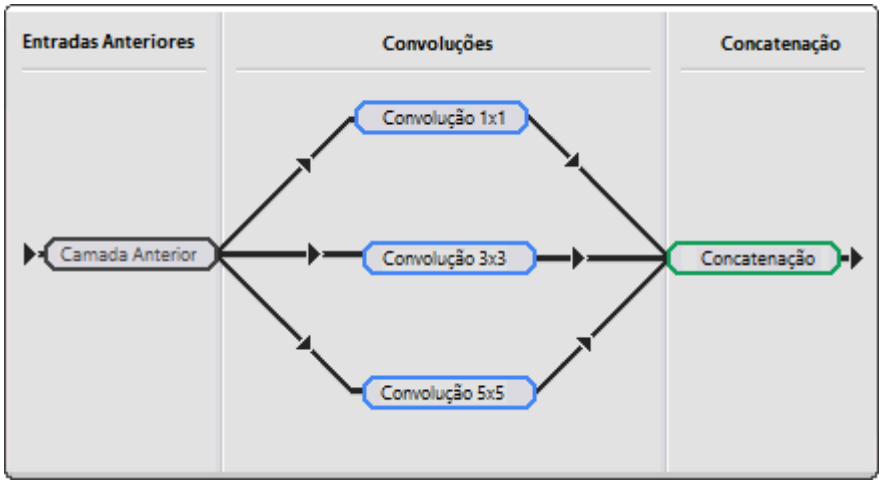

Figura 18 – Topologia Primária do Inception
Fonte: Adaptado de (SZEGEDY et al., 2014)

2.3.6 Inception v3

A rede GoogLeNet ao longo dos anos, foi sendo modificada de forma que houvesse aumento do seu desempenho, assim esta versão ficou conhecida como Inception v1. O Inception v3 (Figura 19), é um modelo de rede convulucional que foi o ganhador no concurso Imagenet no ano de 2015, com uma taxa de erro TOP 5 de apenas 0.03581 em comparação com o modelo original de 0.06656, sendo assim o primeiro vice-campeão da competição. Contando com um total de 42 camadas evolutivas possuindo como entrada imagens RGB com resolução 299x299 tendo como saida um lote de 2048 imagens em 8x8 pixels.O Inception v3 surgiu do resultados de testes de fatoração em relação ao Inception v2, onde a segunda versão teve um foco em normalização de lotes (SZEGEDY et al., 2015).

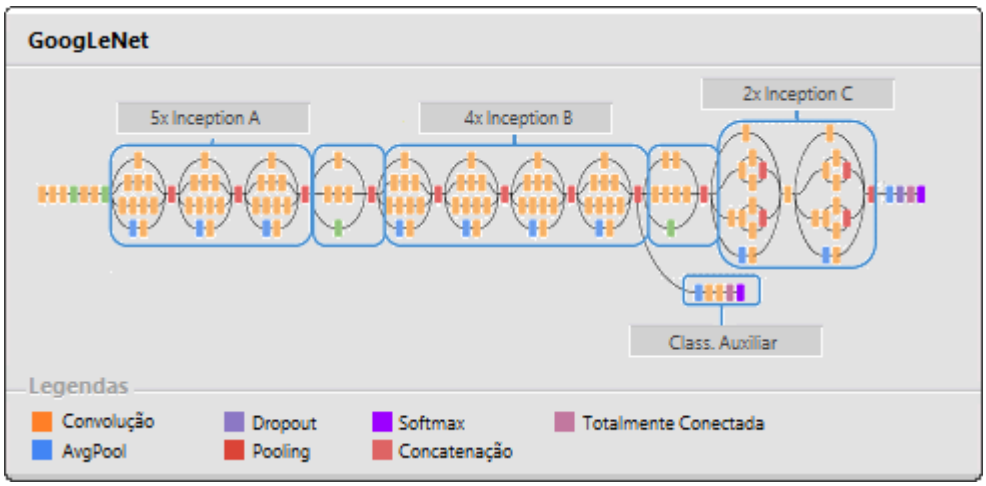

Figura 19 – Topologia Primária do Inception v3

Fonte: Adaptado de (SZEGEDY et al., 2015)

O objetivo por trás das fatorações de convolução descritas como o grande diferencial entre a segunda e terceira versão do Inception. A fatoração de convoluções trata-se de reduzir o numero de conexões e parâmetros definidos na arquitetura de rede sem que necessariamente haja uma redução de desempenho.

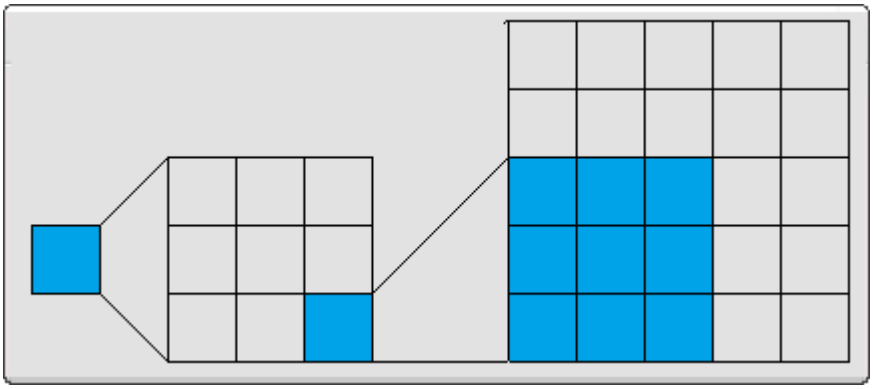

Figura 20 – Fatoração no Inception v3. Onde uma camada 5x5 totaliza 25 parametros, em comparação com duas camadas 3x3 totalizando 18 parametros, assim trazendo uma redução no total 28%.

Fonte: Adaptado de (SZEGEDY et al., 2015)

Um conceito semelhante já foi aplicado na rede VGGNet (SIMONYAN; ZISSERMAN, 2014) que foi a ganhadora do concurso Imagenet no ano anterior. Exemplos da eficacia de fatoração de convoluções pode ser visto na Figura 20

Outro ponto importante em relação ao modelo Inception v3, refere-se a utilização de classificadores auxiliares. Os classificadores auxiliares, que foram utilizados no primeiro modelo da GoogLeNet/Inception v1, no Inception v3 são utilizados apenas uma única vez

no topo da ultima camada 17x17, de forma que ajude a regularizar a classificação da rede (SZEGEDY et al., 2015).

2.3.7 Controle robótico baseado em visão

O campo da robótica se mostra uma das áreas mais promissoras no meio acadêmico, considerando a necessidade cada vez maior de automação de tarefas. Com o objetivo de garantir a maior eficácia destes procedimentos, utiliza-se de técnicas de retroalimentação visual em conjunto com sensores, que garantem uma maior precisão para a sua realização. Este campo ficou conhecido como Controle robótico baseado em visão (do Inglês, Vision-Based Robot Control ou Visual Servoing) (HUCHINSON; CASTAñO, 1994).

Exemplos de usos do controle robótico baseado em visão envolvem o uso de captura de dados visuais para a movimentação de um braço robótico para o recolhimento de determinado objeto. Outro exemplo é a sua aplicabilidade, em um ramo que se mostra bastante promissor, envolvendo a concepção de veículos autônomos. Existem varias formas de se interpretar essas retroalimentações visuais para a realização de ações, desde o uso de técnicas clássicas de processamento de imagem, ao uso de inteligência artificial (BATEUX et al., 2017). O ramo de inteligência artificial utilizado para o controle robótico baseado em visão envolve o uso de técnicas de aprendizado profundo baseadas em convolução.

2.3.8 Exemplos de Veículos Autônomos com aplicação de Técnicas de Deep Learning

O trabalho desenvolvido por Al-Qizwini et al. (2017) faz a utilização de modelos de Redes Profundas Convolucionais, neste exemplo ele utilizou a rede GoogleNet devido a sua baixa taxa de erro no conjunto Top 5%. De acordo com o autor a rede adaptada para o trabalho ficou conhecida como GLAD (GoogLenet for Autonomous Driving). Para este exemplo foi analisado dois parâmetros de affordance (reconhecimento) como podem ser vistos na Figura 21.

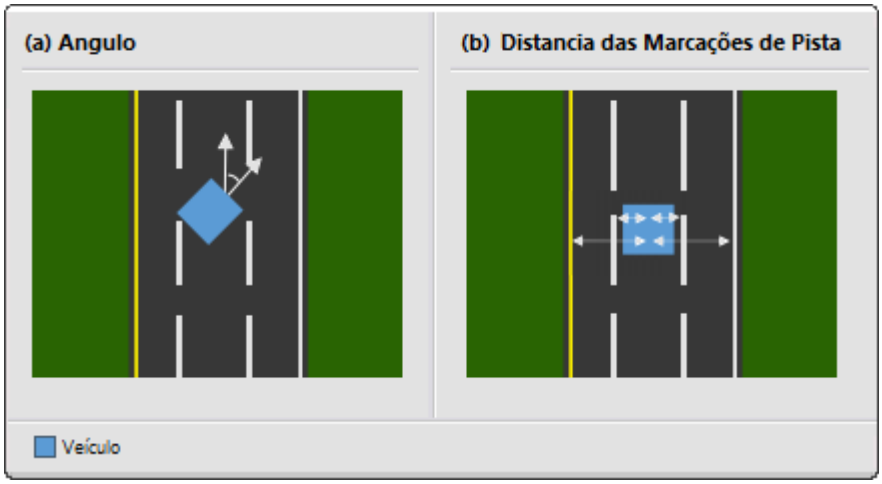

Figura 21 – Parâmetros de Affordance Utilizados no modelo. (a) Angulo da Frontal do Veiculo relativo a tangente da pista ; (b) Distâncias entre o centro do veiculo e as marcações da Pista

Fonte: Adaptado de (AL-QIZWINI et al., 2017)

O treinamento da rede foram realizados em Lotes de 64 imagens cada, para assim atingir os resultados necessários para uma locomoção segura do veículo.

Já o trabalho desenvolvido por Bojarski et al. (2016) funcionários da NVIDIA, utilizando modelos de redes profundas para o aprendizado da forma como os pilotos humanos realizavam ação, este ficou conhecido como DAVE.

Para o hardware a NVIDIA desenvolveu um equipamento de alto desempenho utilizado no experimento, o DRIVE PX2, garantindo um bom processamento em tempo real em conjunto com uma Rede Neural Convolucional.

Para o treinamento desta rede, eles utilizaram 3 conjuntos de câmeras na frontal do carro, esquerda e direita, e como dado de classificação utiliza-se o controle de direção como pode ser visto na Figura 22.

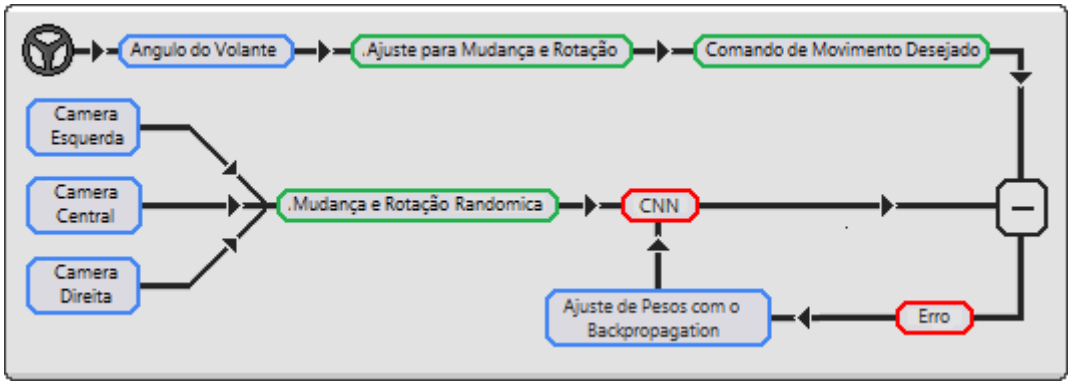

Figura 22 – Treinamento da Rede Neural do Nvidia DAVE

Fonte: Adaptado de (BOJARSKI et al., 2016)

Para a execução da rede em tempo real, o DAVE utiliza apenas a câmera frontal como argumento de entrada alimentando a Rede Neural Convolucional de forma que ela retorne o movimento de volante que será utilizado. A arquitetura da rede convolucional utilizada no projeto consiste em 9 camadas, incluindo uma camada de normalização, 5 camadas convolucionais e 3 camadas totalmente conectadas.

2.4 SISTEMAS EMBARCADOS

Um dos grandes problemas envolvendo a prototipação de equipamentos robóticos vem da capacidade técnica e financeira do desenvolvedor produzir o hardware. No mercado atualmente já existem soluções de baixo custo para resolver este problema como a plataforma Arduino(Figura 23) ou Raspberry PI. Sendo estes microcontroladores de baixo custo que permitem a interação entre hardware e software sem grandes dificuldades ou a necessidade de programação com o uso de linguagem de baixo nível.

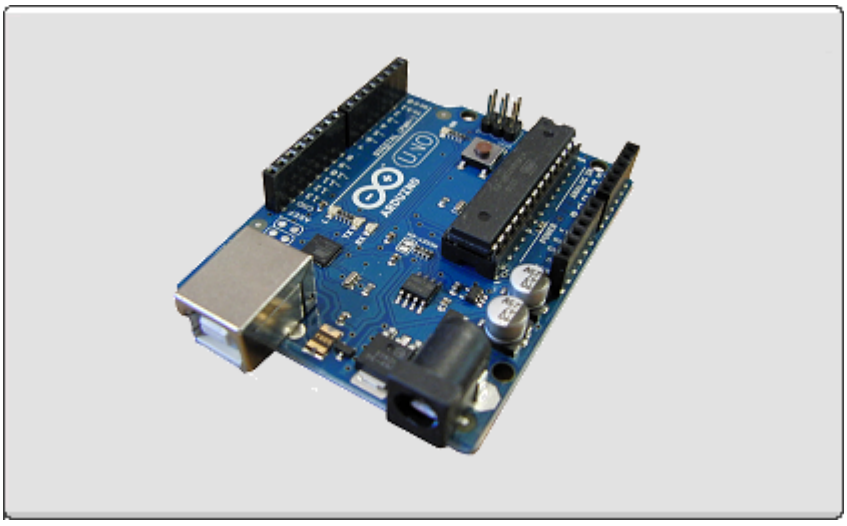

Figura 23 – Imagem referênte ao microcontrolador embarcado Arduino.
Fonte: `https://store.arduino.cc/usa/`

Além das plataformas citadas, existem outros kits de desenvolvimento como o Kit Lego Mindstorms proposto pela Lego Group com foco na educação, sendo este dotado de motores e sensores e possuindo um método de programação intuitivo que envolve blocos de código se mostram uma alternativa para o mercado contando atualmente com três versões sendo a mais recente introduzida no final de 2013.

Figura 24 – Versões do Microcontrolador do Kit Lego Mindstorms: (a) RCX; (b) NXT; (c) EV3
Fonte: Autoria Própria

2.4.1 Considerações

A área de visão computacional como um todo continua se mostrando bastante promissora, considerando as novas tecnologias envolvendo imagem com o uso de inteligência artificial, na qual a cada ano têm-se novos modelos neurais com taxas de erro cada vez menores. Contudo, para a resolução de problemas desta rede, existe a necessidade de um conjunto de treinamento de alta qualidade e quantidade.

Problemas como a criação de veículos autônomos, no qual existe o nível de periculosidade para a vida do ser humano, exigem cuidados especiais, o número de variáveis do ambiente é extenso, assim existe a necessidade da utilização de múltiplos parâmetros para a utilização efetiva com algoritmos de Inteligência Artificial.

3 MATERIAIS E MÉTODOS

Neste capítulo serão apresentados os materiais e métodos empregados no desenvolvimento deste trabalho, que segue desde da captura de dados até o treinamento e validação dos mesmos assim avaliando tanto o software como o hardware.

3.1 HARDWARE UTILIZADO

Com o objetivo de se criar uma plataforma viável de baixo custo, foi optado pelo uso do kit Lego Mindstorms EV3[1], que possui fácil montagem e não requer grandes conhecimentos em Hardware, além de já estar disponível na universidade para uso. O Kit Lego Mindstorms é formado por motores e sensores que serão utilizados no experimento para a captura de dados em tempo real.

A captura de imagens foi realizada através de um vídeo utilizando um dispositivo móvel Android acoplado, no qual, com base nos tempos de captura, os quadros deverão ser extraídos de um arquivo no formato MP4.

Para evitar a necessidade de construção de um módulo autônomo, foi utilizado o M.A.R.C. (FRANÇA; FRANÇA, 2018) para a aplicação do modelo neural, o qual atende os pré-requisitos necessários para o projeto sem necessidade de grandes modificações de construção. Sendo um veículo do tipo *rover*, este já estando na sua segunda versão, com capacidade de acoplar um dispositivo móvel e possuindo alguns sensores de leitura em tempo real em sua construção como pode ser visto na Figura 25.

[1] http://www.lego.com/en-us/mindstorms/

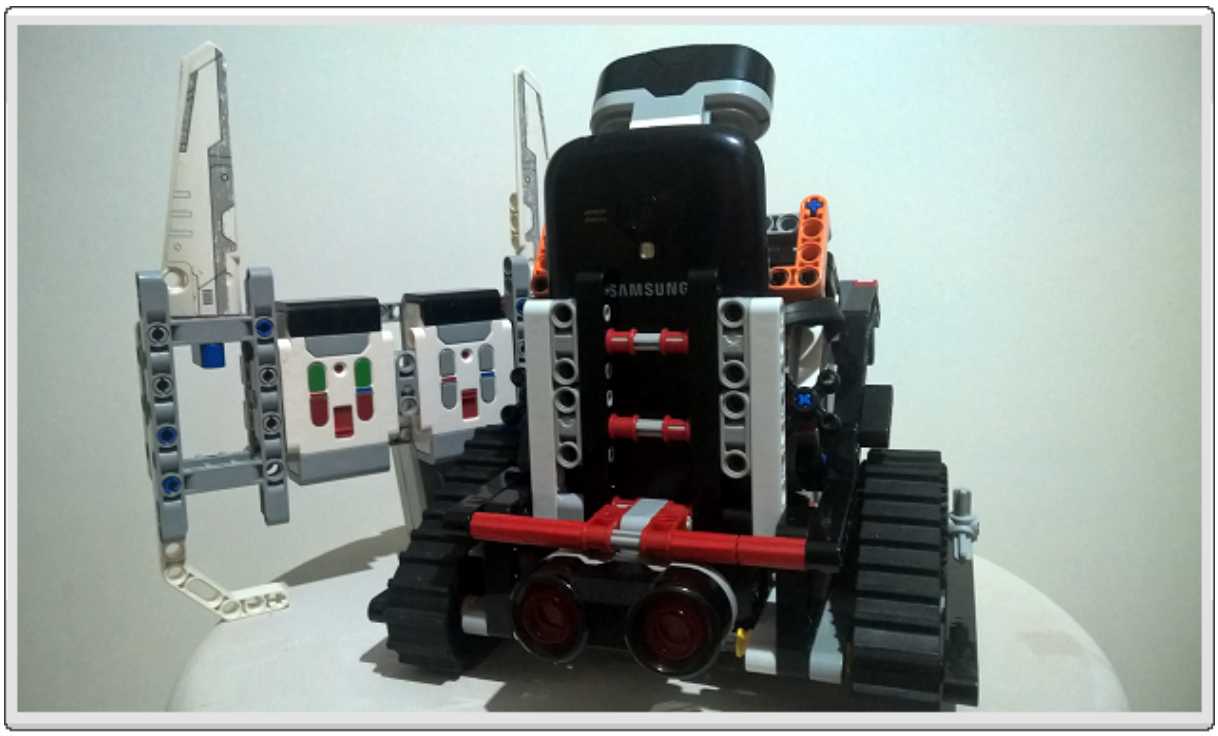

Figura 25 – Visão frontal do M.A.R.C 2 e o seu respectivo controle remoto
Fonte: Autoria Própria

Para o movimento do modulo autônomo foi utilizado um conjunto de controles remotos já mapeados Figura 26. Este controle utiliza-se do sistema infravermelho padrão do EV3, considerando a sua implementação em hardware e garantindo baixa latência. Esta escolha foi feita levando em consideração tratar-se de um módulo robótico não estacionário assim garantindo eficiência energética. Também levando em consideração o fato de ambiente de captura (abordado na Seção 3.3) causou interferência em outros protocolos de comunicação testados como WiFi (*Wireless Fidelity*), além da praticidade para o recolhimento desses dados (FRANÇA; FRANÇA, 2018).

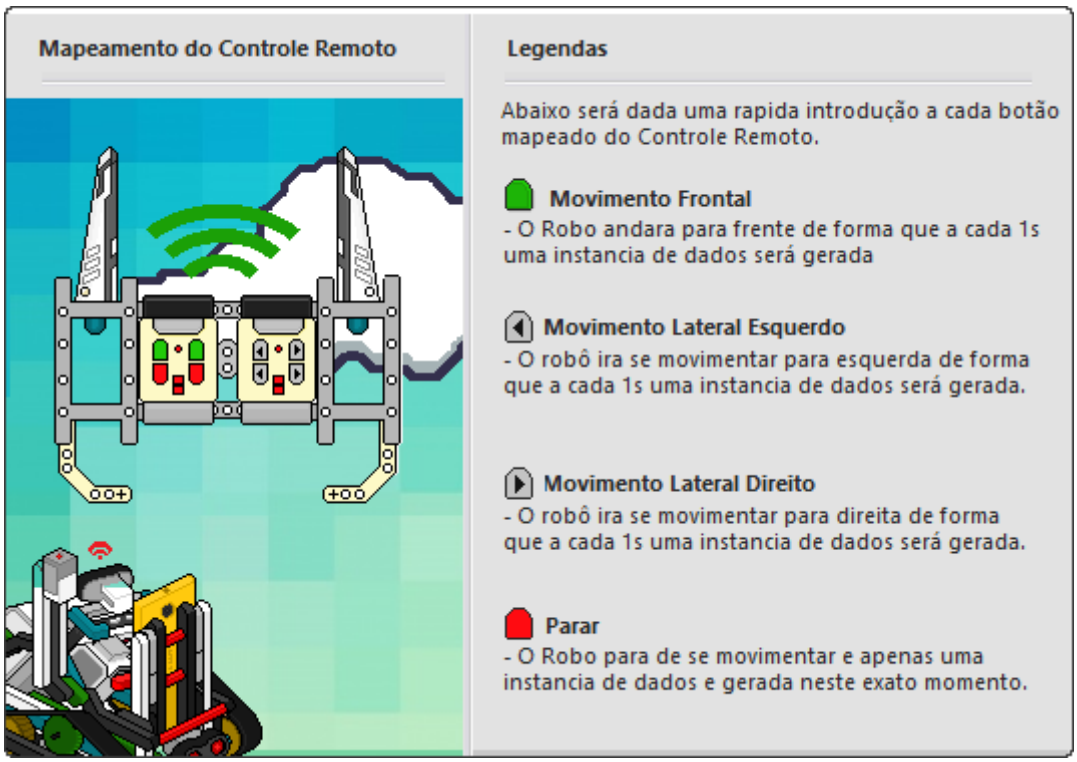

Figura 26 – Representação do controle remoto e o mapeamento dos botões e suas ações durante a captura de dados do ambiente para a geração das instâncias do futuro conjunto de treinamento.

Fonte: Autoria Própria

Além da captura em vídeo, foram utilizados os sensores acoplados ao equipamento, com o objetivo de se extrair múltiplas características do ambiente, gerando uma matriz de dados numéricos onde cada instância de dado deve ser referente a um segundo de captura e deve possuir as seguintes informações:

- **Marca de tempo:** *timestamp* referente ao momento em que a captura frontal em vídeo obtida foi realizada. Deve ser colhida uma captura a cada segundo sendo pós-processada logo em seguida;
- **Dados do sensor de proximidade:** os dados do sensor de proximidade frontal serão capturados retornando um valor representando a distância em centímetros no intervalo de [1-254]. Este sensor é um sensor padrão do Kit EV3;
- **Dados do sensor de luminosidade:** os dados de luminosidade do ambiente referentes à luz azul serão coletados com valores retornados intervalo de 0 a 100. Também é um sensor padrão do Kit EV3;
- **Movimentação desejada:** a rede receberá como informação de entrada sistema de IA se ela deve virar a esquerda, seguir em frente, virar a direita ou parar. Ela só deve tomar uma ação (por exemplo, virar a esquerda) se o ambiente assim o permitir, conforme detalhado

na imagem de entrada.

- **Direção de Movimentação:** a direção no qual o robô está efetivamente se locomovendo com base no botão pressionado no controle remoto para controlar a movimentação no robô. Este dado foi utilizado como a classificação da instância podendo ser (Frente, Esquerda, Direita, Parar).

Durante os primeiros testes do M.A.R.C., com o software que será abordado na seção 3.4, já adaptado para trabalhar com o problema proposto, foi detectado que devido a grande quantidade de dados a ser armazenada e lida em tempo real o equipamento não conseguia lidar corretamente com seu pouco poder de processamento. Para este trabalho foi necessária uma atualização do M.A.R.C onde foi objetivado a construção desta terceira versão alto desempenho com eficiência energética moderada.

Para se atingir os objetivos propostos foi realizada a substituição do Brick EV3, este sendo o controlador geral dos demais equipamentos. Para algo mais poderoso, optou-se pelo uso do Raspberry PI 3[2] , que é um microcontrolador utilizado para a prototipação de equipamentos. O comparativo entre as especificações do EV3 e Raspberry PI 3 é mostrado na Tabela 1.

Tabela 1 – Comparativo entre as especificações do Raspberry PI 3 e Lego Mindstorms EV3

Característica	Raspberry PI 3	Lego Mindstorms EV3
Ano de Lançamento	Fevereiro 2016	Setembro 2013
Memória RAM	1 GB	64 MB
Processador	1.2 GHz 64-bit quad-core ARM Cortex-A53	TI Sitara AM1808 (ARM926EJ-S core) 300 MHz

Fonte: Autoria Própria

Os sensores e motores do Kit EV3 não são compatíveis com o Raspberry Pi 3 considerando que estes se utilizam de entradas RJ12 para a comunicação. Assim foi adquirido o BrickPI3 [3], uma controladora compatível com o Raspberry PI 3, sendo dotada de 8 portas RJ12 sendo 4 para sensores e 4 para motores, alem de um kit de bateria para alimentar o equipamento. O kit se mostrou um ponto bastante positivo considerando que o Raspberry PI 3 possui suporte apenas para alimentação direta na corrente elétrica. Como resultado destas atualizações foi concebido o M.A.R.C 3.0 (Figura 27).

[2]https://www.raspberrypi.org/

[3]https://www.dexterindustries.com/brickpi3-tutorials-documentation/

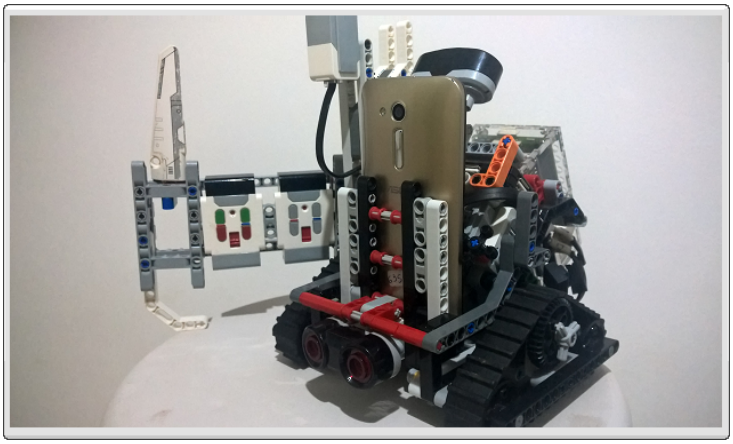

Figura 27 – M.A.R.C 3 já equipado com o Raspberry PI 3 e o BrickPI 3
Fonte: Autoria Própria

3.2 SOFTWARES UTILIZADOS

Devido às limitações de programação do Kit EV3, foi utilizado o Software EV3DEV[4] que é uma variação da distribuição Linux Debian 8[5], compatível com a plataforma que funciona através de um Cartão MicroSD de modo Dual-boot, que permite a interação de software com hardware com suporte a várias linguagens de programação como C++[6], Java[7] e Python[8].

3.2.1 Ambiente de Programação

Como linguagem de programação foi escolhido o Python, devido a sua simplicidade e ao fato de grande parte do material de código aberto referentes às ferramentas de Redes Profundas Convolucionais focadas em veículos autônomos estarem codificadas na linguagem.

[4] http://ev3dev.org
[5] http://debian.org
[6] http://www.cplusplus.com
[7] http://www.java.com
[8] https//www.python.org/

Com o objetivo de armazenar os dados capturados foi utilizado o Sistema de Captura de Dados (S.C.D.), software desenvolvido em conjunto com o M.A.R.C para a realização de captura de dados em tempo real, sendo capaz de ler os dados do ambiente a cada segundo através dos sensores acoplados no equipamento (FRANÇA; FRANÇA, 2018). Porém, foram necessárias algumas atualizações no código, como o armazenamento dos dados capturados em um arquivo de extensão .marcset (Figura 28), sendo este uma matriz contendo todas as informações capturadas em um formato legível para alimentar a RNA. As sessões de captura de dados foram limitadas a 3.600 instâncias que seriam o equivalente a uma hora de captura, o principal motivo é que com a atualização do hardware, o M.A.R.C. perdeu parte de sua eficiência energética, além da necessidade de economia de espaço para os vídeos gravados.

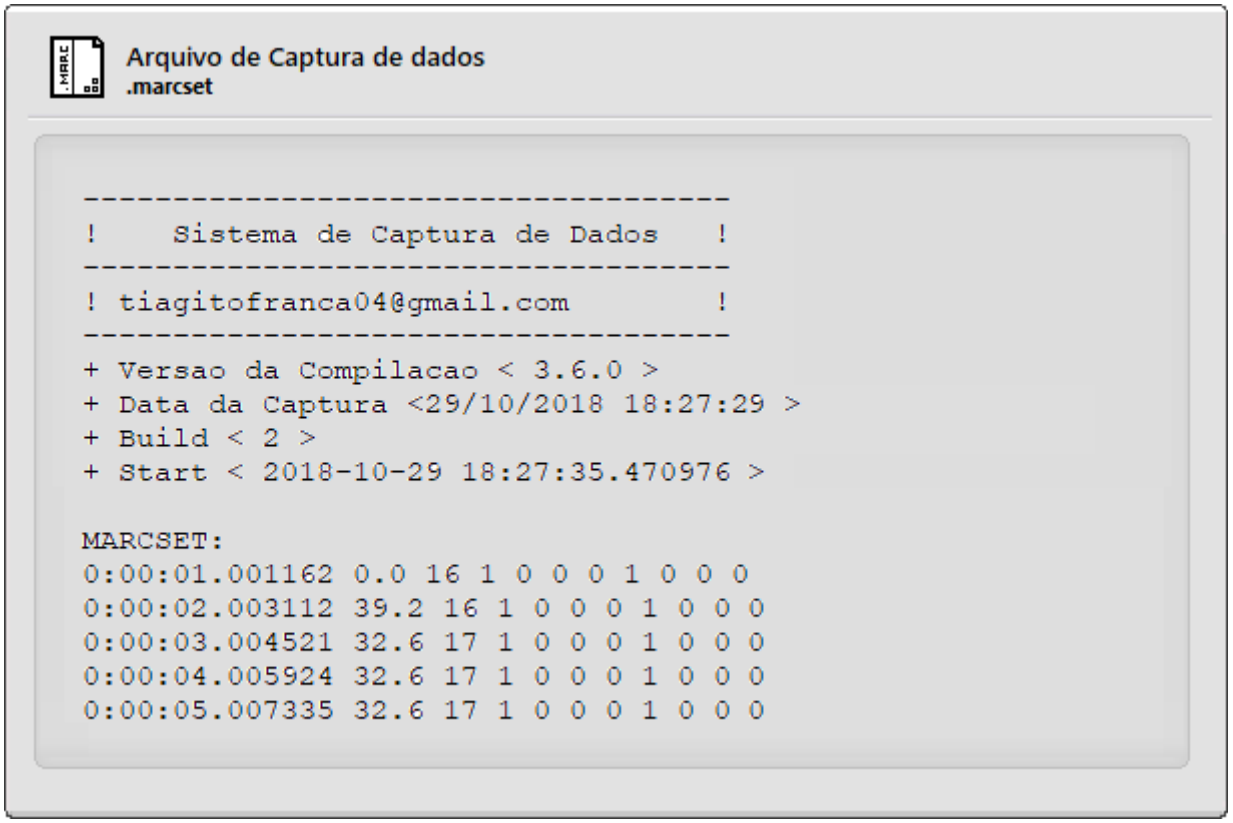

Figura 28 – Exemplo de arquivo salvo na extensão .marcset com um total de 5 capturas de dados
Fonte: Autoria Própria

Para a captura de dados visuais no dispositivo móvel, foi utilizada a biblioteca SL4A[9], sendo esta uma biblioteca responsável por permitir a execução de *scripts* executados na linguagem Python em dispositivos Android. Assim, foi possível portar o S.C.D. para a captura exclusiva de dados visuais. Devido à grande quantidade de dados armazenados, foi

[9]https://github.com/damonkohler/sl4a

necessária a realização do desbloquear o super-usuário (*root*) do Android no aparelho utilizado nos experimentos para que os arquivos de vídeo fossem armazenados no cartão de memória, de modo a atender a uma limitação do SL4A.

A atualização 3.0 no hardware do M.A.R.C. dispensou o uso do controlador do EV3. Porém, devido a plataforma EV3DEV possuir suporte ao Raspberry PI3 e outros dispositivos embarcados, não houve grandes dificuldades para a migração do software. Contudo, ocorreram dois problemas durante esse processo. Primeiramente o Rapsberry PI 3 não é capaz de ler os sensores e motores conectados automaticamente, assim existe a necessidade de configurar as portas antes do início do S.C.D. através de um *shellscript*. Devido a falta de documentação, este foi um processo trabalhoso.

O segundo problema encontrado é mais sério, e consistiu um erro no Kernel Linux do EV3DEV para o Raspberry PI 3. Verificou-se que o sensor de proximidade no modo antena não era capaz de ler múltiplos canais de comunicação exceto o primeiro. Considerando que o controle remoto do M.A.R.C é um conjunto de dois controles menores sendo estes no canal 1 e canal 4, respectivamente, o problema foi notificado ao desenvolvedor e uma correção do kernel foi liberada no dia seguinte[10].

Para o treinamento e execução da Rede Neural Convolucional esta prevista a utilização da plataforma Keras [11], considerando sua portabilidade e compatibilidade com a linguagem Python, o seu uso é ideal considerando a possibilidade de execução da rede em tempo real com o modulo robótico.

Devido à falta de equipamentos necessários, a aplicação dos dados recolhidos para o treinamento da rede foi feito através do uso de CPU e não GPU. Porém apesar deste não ser o método adequado, considerando que o foco do problema proposto é a análise de viabilidade e não um treinamento com baixa taxa de acurácia, este empecilho não se mostra um decréscimo para o experimento.

Para outras formas de comunicação com robô foi utilizado o M.A.R.C.O.S que é um software de autoria própria escrito em IronPython através de uma conexão Socket TCP, sendo este um servidor capaz de se comunicar com o S.C.D., assim sendo possível iniciar e terminar a captura ou testar a rede treinada quando necessário para recolher informações (FRANÇA; FRANÇA, 2018). Para o experimento proposto este foi executado em um Tablet.

[10]https://github.com/ev3dev/ev3dev/issues/1176
[11] https://keras.io/

3.2.2 Protocolo de Comunicação RST

A comunicação entre o S.C.D. e o M.A.R.C.O.S. como cliente e servidor, respectivamente, foi inicialmente codificada de modo que era possível apenas iniciar e terminar a captura de dados. Contudo, houve a necessidade de criação de um novo protocolo de mensagens codificado para este trabalho que ficou conhecido como (RST). O protocolo RST é composto por uma sequência de 4 informações, três caracteres e uma cadeia de caracteres (*string*) seguidos por uma quebra de linha, o que permite depurar as mensagens através de do utilitário de linha de comando telnet. Os 4 valores informados podem ser vistos abaixo:

- **Identificador do dispositivo:** a primeira informação é um caractere que informa o remetente da mensagem, podendo assumir os valores R (Robô), S (*Smarthpone*) ou T (*Tablet*) controlador), sendo este último o servidor, enquanto os dois primeiros são clientes;
- **Tipo de Saída:** a segunda informação, caractere que descreve a quarta informação quando tratar-se de um comando de saída. Se tratar-se de um comando de entrada de dados, é utilizado o caractere ":". Exemplos para esse campo estão na Tabela 2;
- **Tipo de Entrada:** a terceira informação é um caractere que descreve o tipo de periféricos de entrada no qual serão passados dados;
- **Dado:** a quarta e última informação é uma cadeia de caracteres referente ao valor real de entrada ou saída, por exemplo, o valor do sensor de proximidade.

Com o objetivo de otimizar o processo, mais de uma mensagem pode ser passada ao mesmo tempo, desde que em uma única linha de texto separada pelo caractere ";", como exemplo "XXXX;YYYY;ZZZZ". Os tipos de mensagens podem ser vistas na Tabela 2.

Tabela 2 – Tabela referente as mensagens do protocolo RST

Valor	Tipo	Descrição	Exemplo
R	Mensagem de Origem , primeiro caractere a ser informado.	Mensagem referente ao dispositivo de origem neste caso o modulo robótico.	R:::
S	Mensagem de Origem, primeiro caractere a ser informado.	Mensagem referente ao dispositivo de origem neste caso o dispositivo móvel acoplado	S:::
T	Mensagem de Origem, primeiro caractere a ser informado.	Mensagem referente ao dispositivo de origem neste caso o dispositivo o tablet controlador.	T:::

Valor	Tipo	Descrição	Exemplo
A	Mensagem referente a dados de saída, segundo caractere a ser informado.	Mensagem referente a dados do motor conectado ao Robô na Porta A, onde X é um valor numérico.	RA:X
B	Mensagem referente a dados de saída, segundo caractere a ser informado.	Mensagem referente a dados do motor conectado ao Robô na Porta B.	RB:X
C	Mensagem referente a dados de saída, segundo caractere a ser informado.	Mensagem referente a dados do motor conectado ao Robô na Porta C.	RC:X
D	Mensagem referente a dados de saída, segundo caractere a ser informado.	Mensagem referente a dados do motor conectado ao Robô na Porta D.	RD:X
1	Mensagem referente a dados de entrada, terceiro caractere a ser informado.	Mensagem referente a dados do sensor conectado ao robô na porta 1.	R:1X
2	Mensagem referente a dados de entrada, terceiro caractere a ser informado.	Mensagem referente a dados do sensor conectado ao robô na porta 2.	R:2X
3	Mensagem referente a dados de entrada, terceiro caractere a ser informado.	Mensagem referente a dados do sensor conectado ao robô na porta 3.	R:3X
4	Mensagem referente a dados de entrada, terceiro caractere a ser informado.	Mensagem referente a dados do sensor conectado ao robô na porta 4.	R:4X
#	Mensagem referente a ações, ultimo caractere a ser informado	Mensagem referente ao inicio da captura	R::#
*	Mensagem referente a ações, ultimo caractere a ser informado	Mensagem referente ao fim da captura	S::*
#	Mensagem referente a ações, ultimo caractere a ser informado	Mensagem referente ao inicio da captura	R::#
*	Mensagem referente a ações, ultimo caractere a ser informado	Mensagem referente ao fim da captura	S::*

Fonte: Autoria Própria

3.2.3 Pre-Processamento dos Dados

Antes do pré-processamento de dados, foi necessário elaborar um método de extração dos dados visuais considerando que o S.C.D. presente no dispositivo móvel estava executando gravações em vídeo. Assim, foi utilizada a biblioteca FFMPEG[12] ,sendo esta capaz de extrair frames de arquivos de video e armazenar as mesmas em um arquivo de imagem correspondente com base em marcações de tempo informadas.

Durante as capturas serão gerados dois arquivos na extensão .marcset, sendo o primeiro gerado no módulo robótico, e o segundo no smartphone acoplado. Os dois arquivos contêm o horário de captura, com precisão de microssegundo, informado no seu escopo, assim é feita uma operação de subtração entre os tempos, com o objetivo de manter o sincronismo de captura com imagem como pode ser visto na Figura 29.

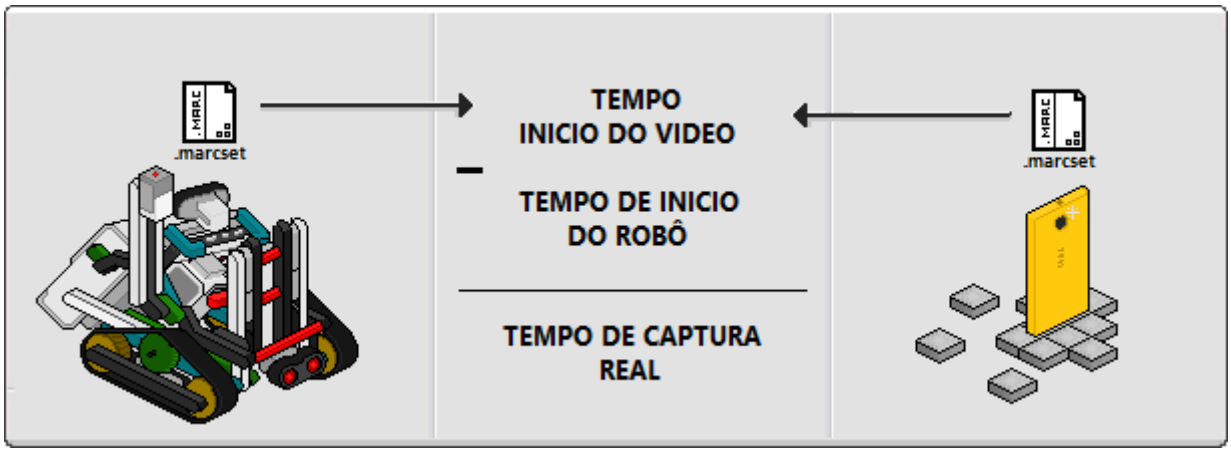

Figura 29 – Representação do método de sincronismo entre os diferentes dispositivos durante a captura de dados.

Fonte: Autoria Própria

Esta operação de mostrou importante considerando que foi observado que o dispositivo móvel começava a gravar o vídeo de captura antes do primeiro movimento ser executado.

Apos a extração para o pre-processamento e recolhimento de informações visuais foi utilizado o OpenCV[13]. No caso do M.A.R.C.O.S., para a visualização de vídeo em tempo real foi utilizado o EmguCV[14], este sendo um wrapper para C# compatível com a plataforma IronPython.

[12] https://www.ffmpeg.org/
[13] https://opencv.org/
[14] http://www.emgu.com

Foram realizados alguns testes de transformações de canais de cor, se utilizando da biblioteca OpenCV, durante o pre-processamento de dados foram feitas as seguintes transformações (Figura 30), para assim verificar a viabilidade de otimização desses dados para a rede neural. Estas imagens serão obtidas da extração dos quadros do vídeo correspondente a captura.

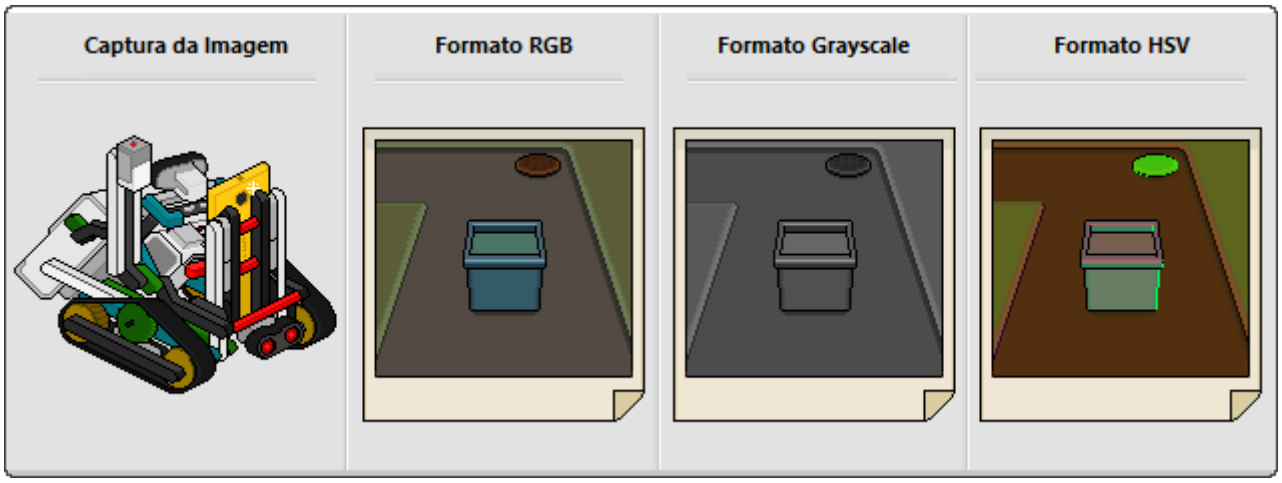

Figura 30 – Representação de conversão da imagem do ambiente para diferentes canais de cores.
Fonte: Autoria Própria

Entre os canais de cor, com o objetivo de otimizar o experimento, durante a etapa de pós-processamento foi optado pela criação de instâncias correspondentes das imagens no formato em tons de cinza (*Grayscale*) visando trabalhos futuros, considerando que os modelos de rede Alexnet,GoogleNet e Inception v3 pré-treinados que foram encontrados para o experimento exige uma entrada em RGB para a execução do algoritmo de treinamento.

3.3 CONJUNTO DE TREINAMENTO

Para a montagem do conjunto de treinamento foi realizado o treinamento diário de pelo menos uma hora, considerando que a cada um segundo uma instância deve ser gerada. A previsão de captura de instâncias é de no minimo 3.600 instâncias diárias considerando a limitação proposta anteriormente em cada sessão. Com previsão de trabalho de no mínimo 36.000 imagens até o fim deste trabalho. Segue na Figura 31 uma demonstração de como foi realizada a captura de uma instância a cada segundo.

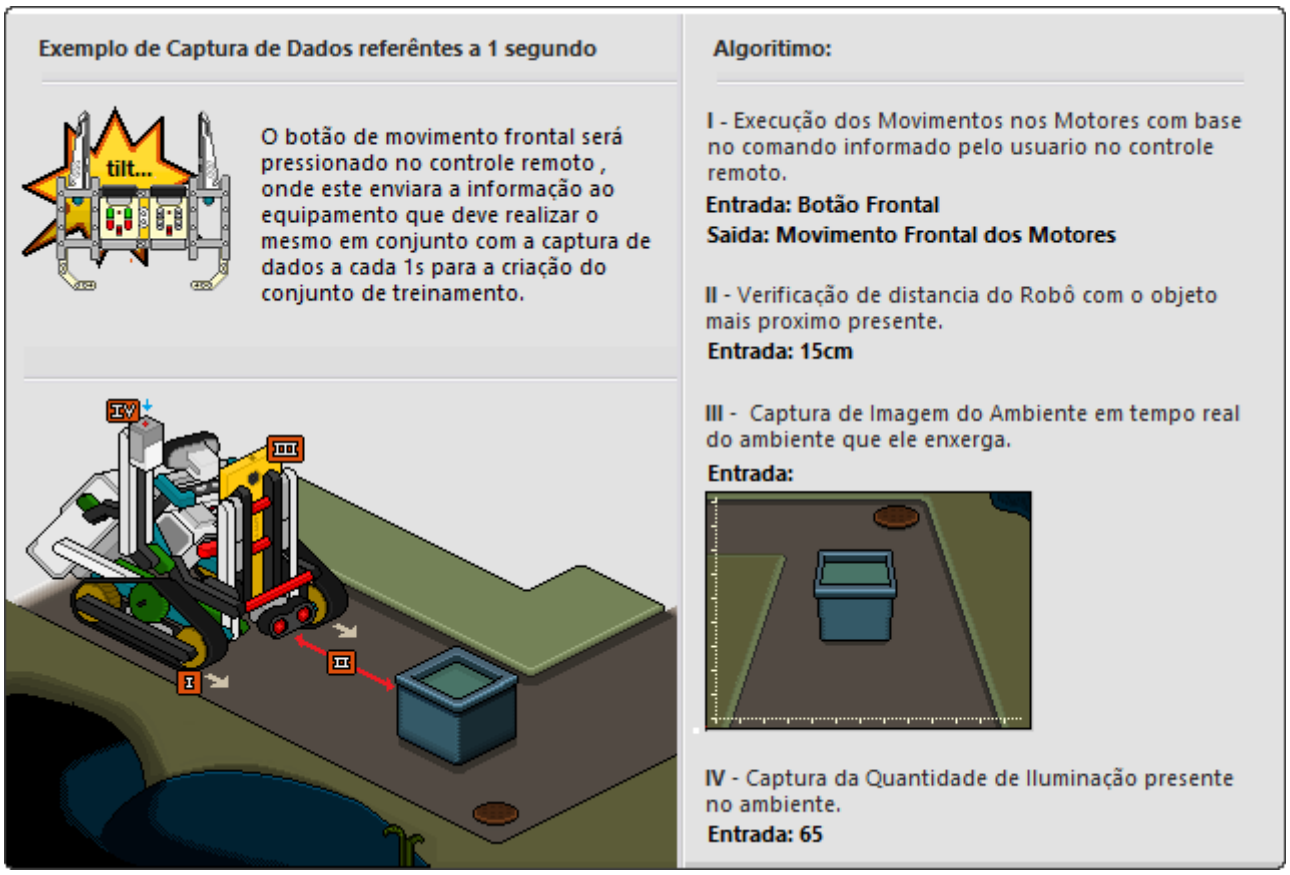

Figura 31 – Representação de captura de uma instância de dados referente a 1 segundo.
Fonte: Autoria Própria

Como rede de treinamento foi escolhida a Inception v3. Inicialmente foi proposto o uso da Alexnet, porém existiram alguns problemas, pois não foi encontrado um modelo pre-treinado compatível com a plataforma Keras, o que seria um empecilho para a execução da rede em tempo real a longo prazo. O ideal para o treinamento independente da rede neural é que este seja feito com uma GPU considerando que a própria Alexnet e modelos Inception foram otimizados para a execução em CUDA[15].

3.3.1 Captura em Campo

A rede foi pensada de modo que o aprendizado identifique que as paredes dos corredores atuem como guia, porém devido a instalação vertical do dispositivo móvel no modulo

[15] https://developer.nvidia.com/cuda-zone

robótico este campo de visão de mostrou bastante limitado, o que pode prejudicar o resultado final. Assim foram feitos testes com algumas lentes que foram acopladas ao dispositivo com o objetivo de expandir a área visível, conforme mostrado na Figura 32.

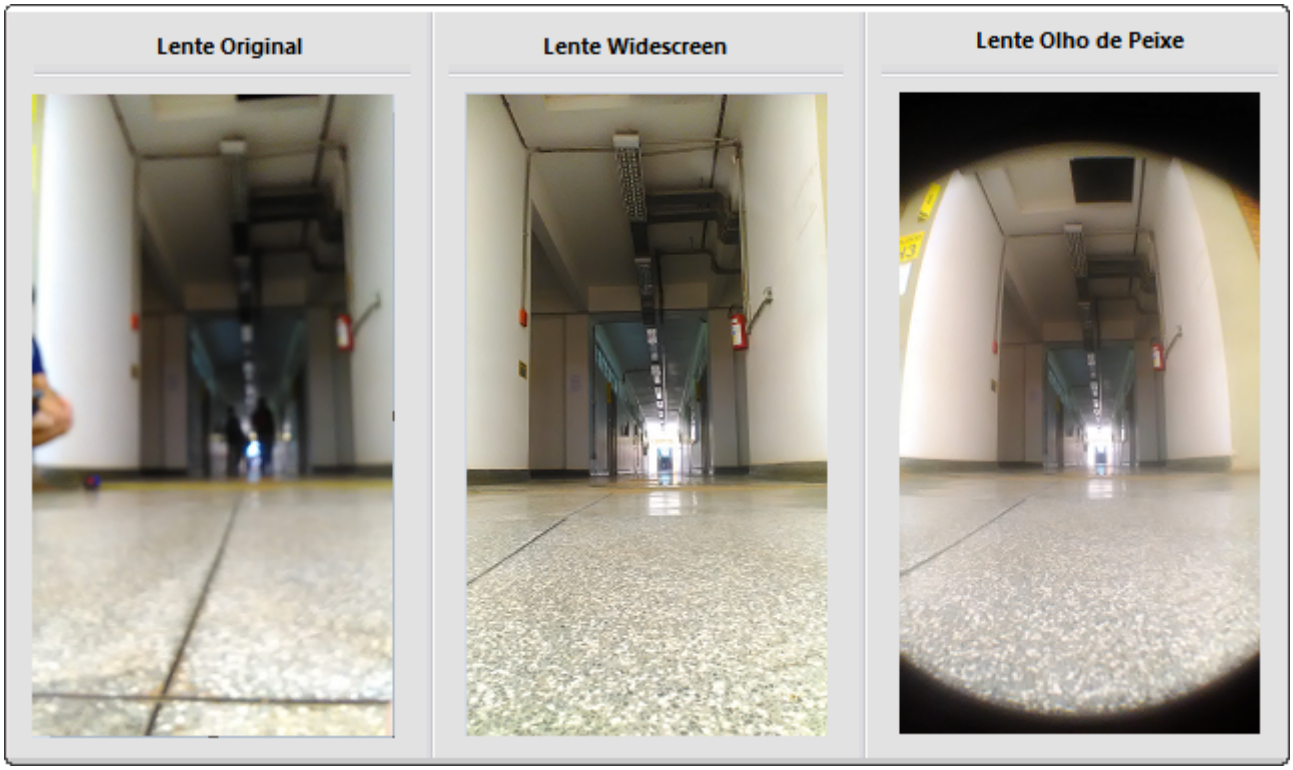

Figura 32 – Teste de abertura com diferentes lentes feitos nos corredores do campus.

Fonte: Autoria Própria

Entre as três situações testadas, a lente do tipo olho de peixe foi a que se mostrou com melhores resultados garantindo uma abertura maior para a visualização da parede conforme pode ser visto na tabela 3. Apesar da lente utilizada possuir um pequeno defeito de fábrica foi constatado que a rede foi capaz de ignorar o mesmo durante o aprendizado.

Tabela 3 – Tabela de Níveis de Qualidade para a Avaliação Extrínseca

Tipo	Abertura Aproximada
Lente Original	1.90m
Widescreen	1.80m
Olho de Peixe	4.80m

Fonte: Autoria Própria

Foram feitos testes de iluminação e foi verificado que a qualidade da imagem diminuiu com o uso do flash frontal do dispositivo móvel, gerando imagens borradas, assim o flash foi desativado durante os testes subsequentes.

Com o uso da lente olho de peixe, verifica-se uma área não utilizada da imagem. Considerando que essa área não interfere no aprendizado da rede neural é feita uma segmentação com auxílio da biblioteca OpenCV com o objetivo de otimizar o aprendizado considerando uma dimensionalidade menor da imagem.

Com o objetivo de satisfazer os critérios de captura e execução em campo previstos para este trabalho, os corredores da universidade foram mapeados em duas diferentes zonas:

- **Zona de Treinamento:** região onde serão feitos as capturas de dados para montagem do dataset e validações de acurácia com novas instâncias;
- **Zona de Execução em Tempo Real:** esta região nunca será utilizada durante o treinamento.Sendo reservada para aplicação em tempo real, com o objetivo de verificar se a rede é capaz de generalizar o seu aprendizado.

O mapeamento foi feito de forma que fosse possível reservar pelo menos um terço dos corredores para a execução em tempo real, outro critério levado em consideração foi a escolha de corredores para a execução do experimento que possuíssem características em comum. O Mapeamento dos corredores pode ser visto na Figura 33.

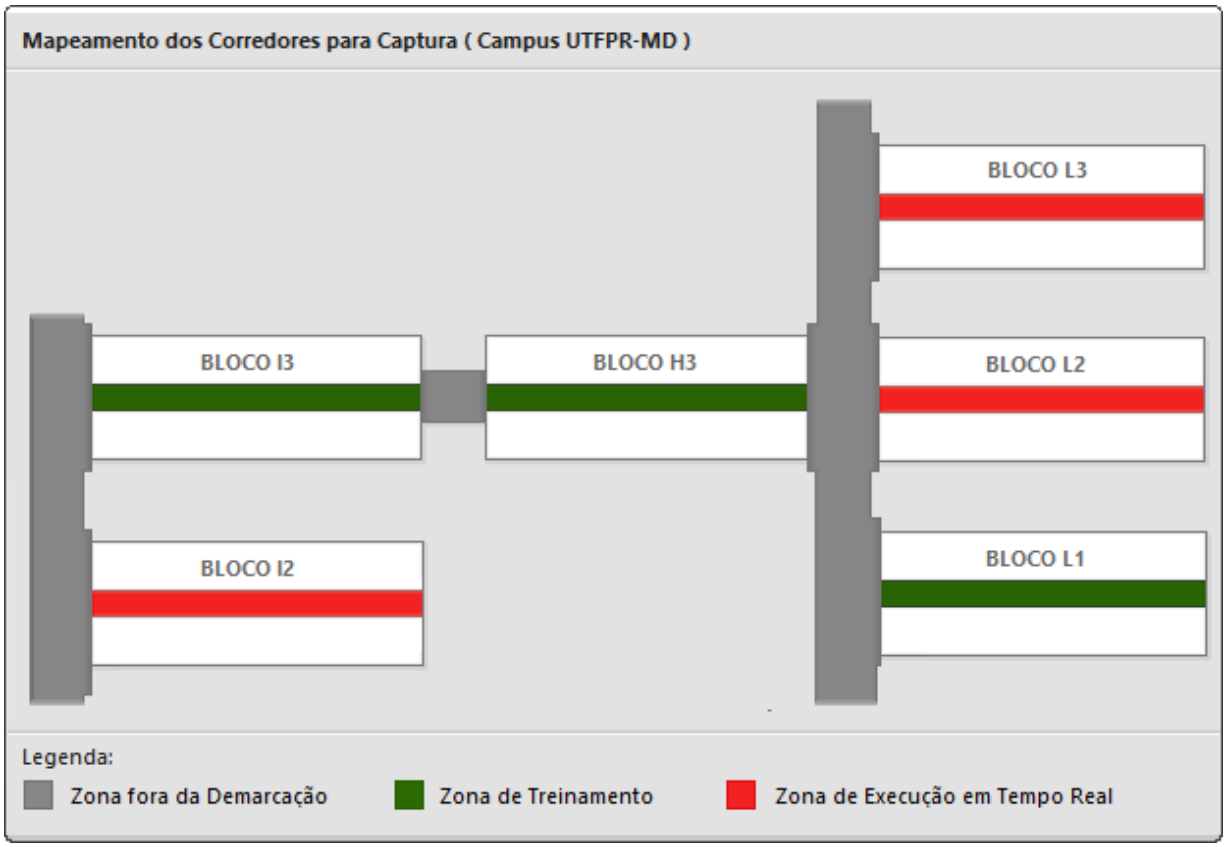

Figura 33 – Representação de Processamento de Dados na Rede Convolucional com sua Resposta .
Fonte: Autoria Própria

3.4 EXECUÇÃO DA REDE EM TEMPO REAL

Após executado o treinamento da rede, é previsto que sejam aplicados testes no modelo treinado em conjunto com o módulo autônomo, para assim validar os resultados do experimento. A previsão é de utilização dos dados em tempo real a cada um segundo conforme definido nos critérios de captura.

Para a execução destes testes foi desenvolvido um software, considerando que o S.C.D. era incapaz de realizar a tarefa, apesar de partes de código referentes a movimentação terem sido reaproveitadas. Nesta etapa, o dispositivo móvel funciona como uma câmera IP em tempo real, na qual os quadros são transmitidos através de rede sem fio para o modulo robótico, onde são extraídas e pre-processadas com auxilio da biblioteca OpenCV em conjunto com os dados dos sensores enviados em tempo real para a rede convolucional. A rede é responsável por retornar uma resposta referente ao movimento que a ser executado, conforme pode ser visto na Figura 34.

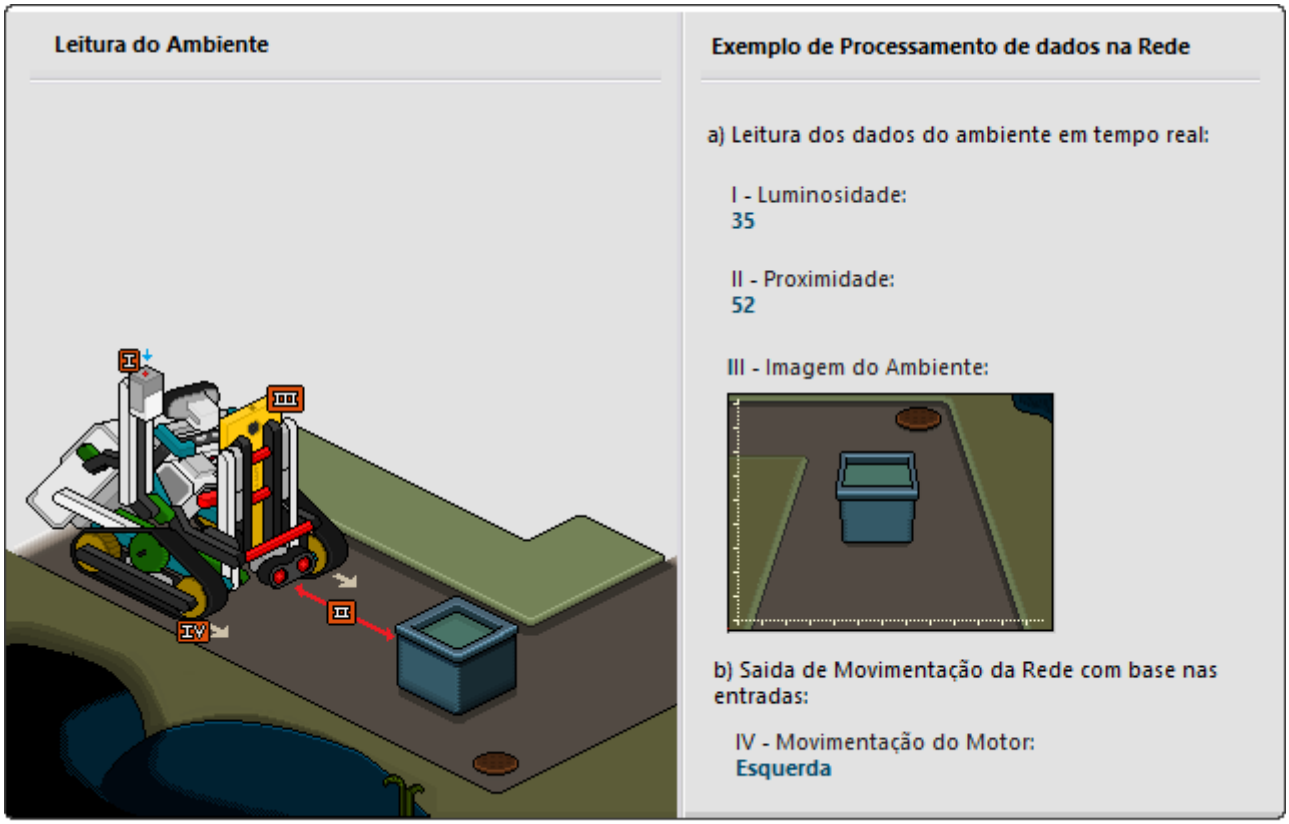

Figura 34 – Representação de Processamento de Dados na Rede Convolucional com sua Resposta .

Fonte: Autoria Própria

Esta metodologia para execução foi abordada considerando que não foi encontrada uma forma viável de executar no dispositivo móvel a Linguagem Python (SL4A) em conjunto com a biblioteca OpenCV.

4 RESULTADOS

Neste capítulo serão apresentados os resultados correspondentes das metodologias adotadas para o desenvolvimento deste trabalho.

4.1 COMPORTAMENTO DE HARDWARE

Durante as primeiras capturas, foi verificado que devido a atualização no seu microcontrolador, o módulo robótico era incapaz de se sustentar por mais de 10 minutos de capturas.

Inicialmente, utilizando-se de um conjunto de 12 Pilhas de 1.2v e capacidade de 1900mAh cada, foi verificado que devido a movimentação dos motores, os níveis de carga se mostravam irregulares. Assim com o objetivo de atender o módulo como um todo, foi adicionado um Powerbank com saída de 1.5V e um total 5000mAh.

Após esta atualização constatou-se que o módulo robótico era capaz de realizar capturas com um tempo total próximo a uma hora, sendo este resultado condizente ao valor limite delimitado por sessão durante a metodologia do trabalho.

Com o objetivo de otimizar a carga total do equipamento, foi feita uma pequena atualização no software, onde os sensores e motores desativados durante a etapa de pós-processamento de dados ou caso não haja nenhum captura em andamento.

4.2 PROCESSO DE CAPTURA

Nesta subsessão são abordadas os resultados referentes a etapa de captura, problemas e necessidades de adaptação em relação a metodologia proposta.

4.2.1 Interferência do Ambiente de Captura

Para a captura dos dados nos corredores da universidade foram encontrados alguns problemas. Diferentemente da aplicação teórica do S.C.D. em laboratório, foi necessária a criação de uma rede local sem fio móvel, de forma que a comunicação entre os dispositivos(Robô, Smartphone e Tablet) fosse realizada com confiança. Porém, foram encontrados alguns problemas de interferência devido a rede sem fio interna do campus, assim foram exploradas algumas alternativas:

- **Criação de um *hotspot* sem fio utilizando o *tablet***: como primeira tentativa foi efetuada a criação de uma rede sem fio através da ferramenta de *hotspot* embutida no sistema Microsoft Windows, porém foi constatado a instabilidade da rede levando em consideração que não era possível a escolha do canal de comunicação, além do pouco poder da placa embarcada do *tablet*;
- **Criação de um *hotspot* sem fio utilizando um *tablet* com um *dongle* USB**: com o objetivo de tentar manter a estabilidade do sinal, foi acoplado ao *tablet* um *dongle* USB no qual este seria o *host* da rede sem fio, porém, novamente, apesar do sinal de melhor qualidade, devido a interferência causada pela rede do campus, este se mostrou inviável;
- **Criação de uma rede *ad-hoc* utilizando o tablet**: como última tentativa baseada no *tablet* com o objetivo de manter todos os equipamentos conectados a rede foi optado pelo uso de uma rede *ad-hoc*, porém esta se mostrou incapaz de manter a estabilidade da conexão.
- **Criação de um *hotspot* sem fio utilizando o dispositivo móvel**: utilizando a ferramenta de *hotspot* sem fio dispnibilizada pelo sistema Android do dispositivo móvel, novamente encontra-se problema devido a incapacidade de escolha do canal de comunicação da rede, assim devido à grande quantidade de interferência, este foi descartado;
- **Pareamento USB entre os dispositivos**: com o objetivo de tentar reaproveitar as

funcionalidades disponibilizadas pelo sistema Android, foi experimentado o pareamento USB com o objetivo de criar uma rede local, este se mostrou parcialmente um sucesso, porém não foi possível efetuar uma conexão a três dispositivos (Raspberry, celular e *tablet*), assim esta opção foi descartada;

- **Pareamento Bluetooth entre os dispositivos**: outra possibilidade disponível pelo sistema Android é a capacidade de criação de um rede local através do Bluetooth, porém este se mostrou bastante instável, devido a diferentes versões de implementação hardware dos dispositivos, sendo mais um método descartado.
- **Criação de um *access point* utilizando o módulo robótico**: com o objetivo de tentar garantir mais confiabilidade, foi testada a criação de um *access point* utilizando o M.A.R.C. como roteador. Esta tentativa mostrou os melhores resultados, utilizando do software Create_Ap[1], sendo este uma ferramenta pronta para a criação de um ponto de acesso em sistemas Linux. Foi possível criar uma rede local de forma que permitiu escolha do canal de comunicação, assim evitando interferências. O canal 1 foi o que mostrou o melhor resultado. Devido a isso, esse foi método escolhido para a experimentação do trabalho de captura.

O software Create_Ap permite a configuração manual de uma rede local, com o objetivo de automatizar as capturas foi optado pelo uso do sistema DHCP estático. A grande vantagem do seu uso é não necessidade de atualização do IP do servidor toda vez que o ponto de acesso for inicializado. A topologia de rede pode ser visualizada na Figura 35.

[1] https://github.com/oblique/create_ap

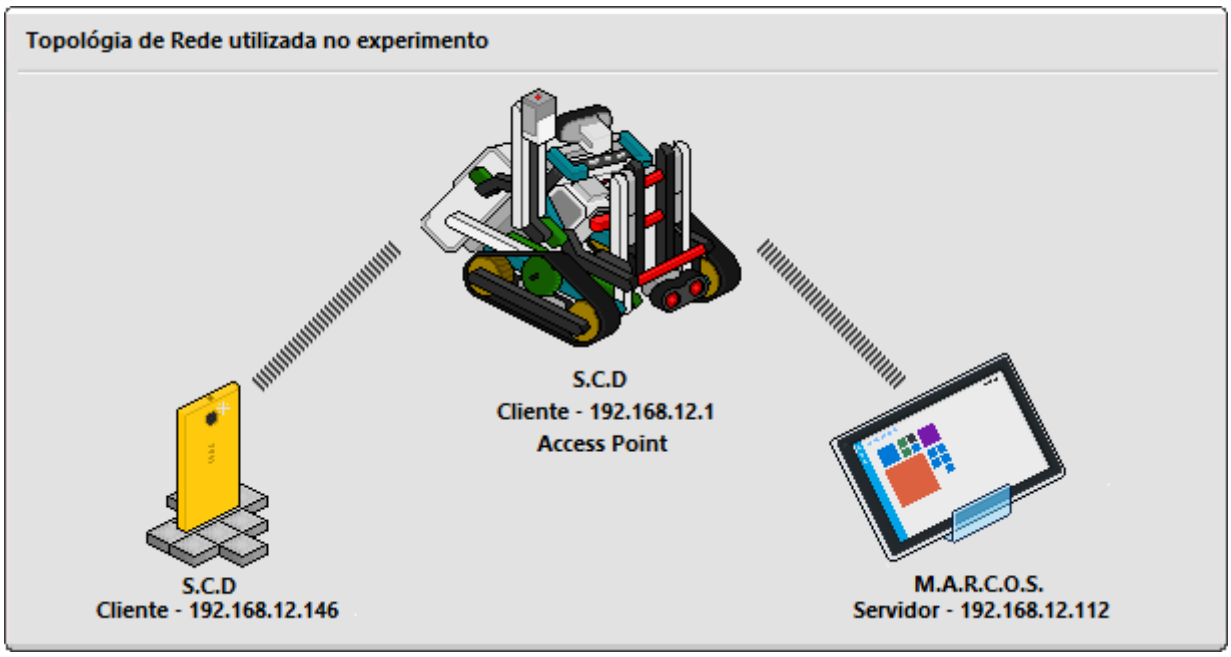

Figura 35 – Topologia de Rede sem Fio aplicada no Experimento entre os dispositivos (Robô, Tablet, Smartphone) utilizados no experimento com seus respectivos softwares.

Fonte: Autoria Própria

Após as configurações do método de comunicação, foram executadas as primeiras capturas, as imagens foram capturadas em formato de vídeo no tamanho de 840x400. Devido à baixa resolução, constatou-se a baixa qualidade em ambientes com mudanças bruscas de iluminação conforme pode ser visto na Figura 36.

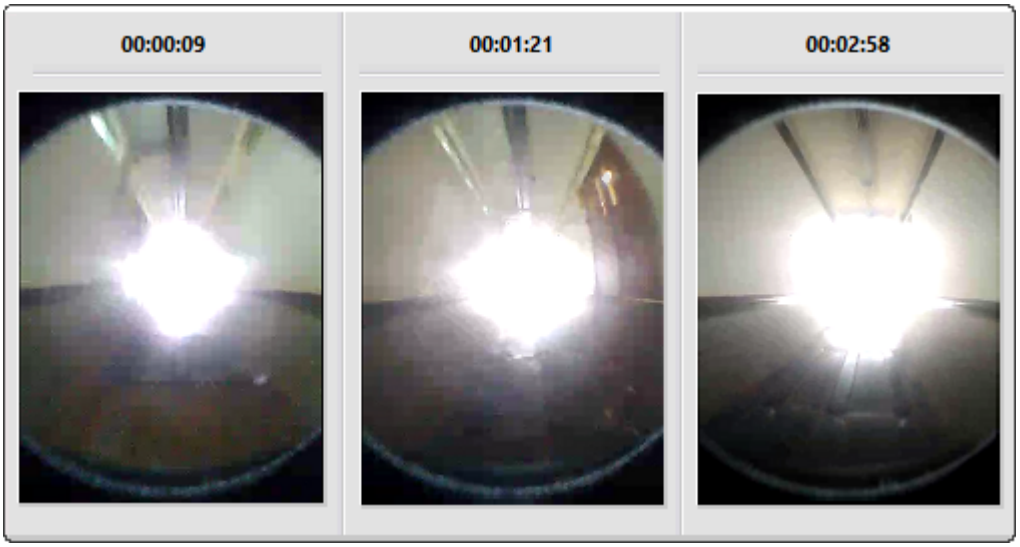

Figura 36 – Quadros extraídos da captura realizada na data 19/10/2018 com seus respectivos *timestamps*, onde é possível verificar ruídos causados pelo excesso de luz solar durante o processo.

Fonte: Autoria Própria

Estas primeiras capturas foram descartadas, considerando os artefatos gerados devido a luminosidade, assim para as próximas tentativas evitou-se a captura durante horários de maior incidência solar, garantindo a capacidade da câmera de visualizar o corredor corretamente.

4.2.2 Instâncias de Captura

Se utilizando dos arquivos gerados durante a captura de dados foi possível obter um total de 440 instâncias para os primeiros experimentos. Este número apesar de expressivo em um primeiro momento se mostra ineficiente para um treinamento adequado da RNA. Porém, o objetivo deste trabalho não é o de atingir o melhor resultado de treinamento possível e sim validar a possibilidade de aplicação de um módulo de baixo custo com técnicas de redes neurais. Os processos de captura executados podem ser visualizados na Tabela 4.

Tabela 4 – Tabela referente as capturas realizadas

Data	Build	Quadros	Status	Observações
15/10/2018	1	6	Descartada	Problemas relacionados a rede, onde resultaram na queda do servidor.
15/10/2018	2	4	Descartada	Problemas relacionados a rede, onde resultaram na queda do servidor.
15/10/2018	3	163	Descartada	Problemas relacionado ao sincronismo entre os dispositivos em relação ao Timestamp.
15/10/2018	4	186	Descartada	Problemas relacionado a falta de tempo de inicio no dispositivo móvel.
15/10/2018	5	112	Descartada	Problema relativo a movimentação do dispositivo influenciava na captura correta dos movimentos.
17/10/2018	1	78	Descartada	Problema de movimentação não foi corrigido da forma esperada
18/10/2018	1	216	Descartada	Faltou implementação adequada para o uso de SSH, e pre-processamento de dados.
19/10/2018	1	216	Descartada	Problemas de Iluminação Causados pelo sol.
21/10/2018	1	34	Válido	Teste executado com sucesso.
22/10/2018	1	234	Válido	Captura no corredor no bloco de computação
29/10/2018	1	172	Válido	Foram executadas a extração parcial de um total de 263 frames devido a falta de alimentação no modulo robótico.

Fonte: Autoria Própria

O principal motivo a ser levado em consideração que justifique o número insuficiente

de capturas foi o tempo para desenvolvimento do trabalho, a adaptação do material já construído para o problema, bem como os problemas apresentados detalhadamente na Seção 4.2.

O M.A.R.C. 2 se mostrou incapaz de lidar com tantas instâncias de capturas, principalmente devido a necessidade de armazenamento de novas colunas de dados para esta ser alimentada na rede. Assim, foi necessária uma atualização de hardware no qual como resultado levou ao M.A.R.C. 3. A atualização do módulo robótico trouxe alguns problemas, que tomaram bastante tempo de desenvolvimento.

Durante as primeiras capturas foi verificado que estas apresentavam alguns problemas, entre eles considerando que com o objetivo de balanceamento de carga entre os dispositivos o sistema funciona de forma distribuída, foi verificada algumas perdas de informações. Com o objetivo de garantir uma confiabilidade foi implementado um processo de validação suplementar onde é verificado se todos os arquivos necessários para o experimento estão intactos para a extração de características da rede.

Após o processo validado, foi verificado que com base nas imagens geradas, existe a possibilidade de *overfitting* do modelo neural, considerando que durante as capturas a maior parte das classificações foram referentes a movimentação frontal.

4.3 PROCESSO DE TREINAMENTO

Com o objetivo de validar a hipótese proposta para este trabalho, foi utilizado um modelo pré-treinado da rede Inception v3, considerando os motivos apresentados na seção 3.3. As imagens do conjunto de treinamento foram redimensionadas e cortadas no formato 139x139 pixels em RGB, assim acelerando o processamento. Após o redimensionamento foi realizada uma operação matricial sobre a imagem, zerando a primeira linha nas 3 camadas da imagem, onde nesta seriam armazenados os dados referentes à movimentação e aos sensores, como pode ser visto na Figura 37.

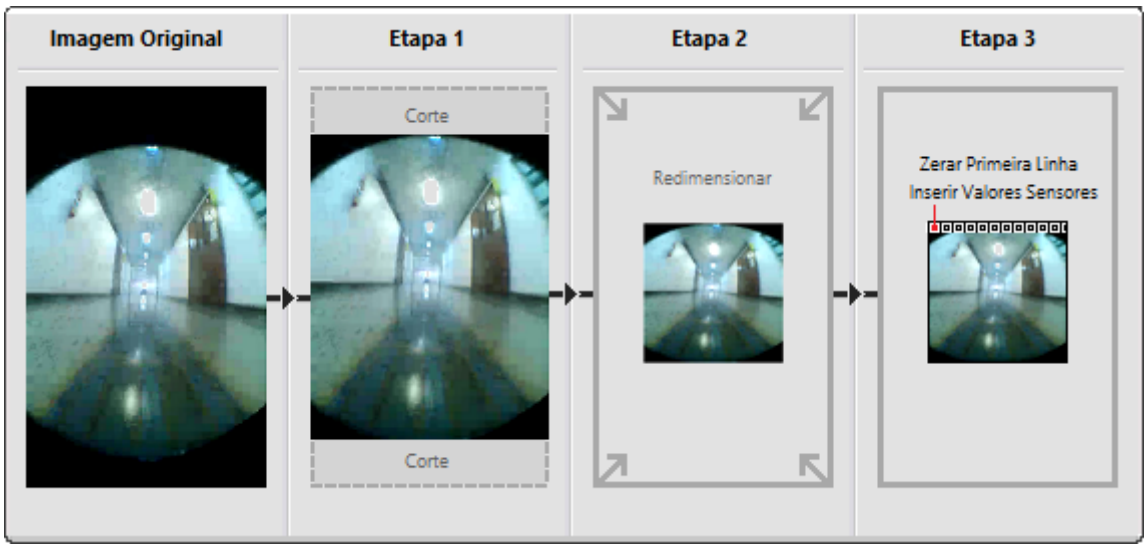

Figura 37 – Etapas de Pre-Processamento da Imagem para aplicação na Rede Neural
Fonte: Autoria Própria

Considerando que um pixel de imagem tem um valor inteiro entre 0 e 255, foi necessária a normalização dos elementos numéricos referente ao Sensor de Proximidade (Equação 8), Sensor de Luminosidade (Equação 9) e Movimentação (Equação 10).

$$Proximidade = round(Proximidade\_Original) \tag{8}$$

$$Luminosidade = round(\frac{255 * Luminosidade\_Original}{100}) \tag{9}$$

$$Movimento = \begin{cases} 255, & \text{se } x = 1 \\ 0, & \text{senão} \end{cases} \tag{10}$$

Considerando o uso do Keras para a aplicação, foi gerado dois arquivos no formato .numpyz, sendo o primeiro um vetor contendo todas as imagens já pré-processadas com auxílio da biblioteca OpenCV, e um segundo referente às classificações da imagem. Esta se mostrou a melhor opção, considerando a dificuldade de utilização das imagens no formato PNG.

A aplicação foi rodada na plataforma Google Colab[2], que é uma plataforma de Nuvem gratuita fornecida pela Google, utilizada amplamente para execução de scripts na área de I.A. A sua execução em conjunto com o Keras objetivando a busca de resultados se mostrou viável, assim foi realizado o treinamento inicial da RNA com um total aproximado de 350 instâncias pré-processadas. Inicialmente foram encontrados alguns problemas iniciais, envolvendo o uso Keras em conjunto com o modelo pré-treinado do Inception v3.

Apesar dos obstáculos, foram executadas as devidas correções no modelo. Assim um treinamento foi realizado com um total de 500 épocas, considerando o pouco tempo para desenvolvimento das etapas finais. Este treinamento foi realizado de forma experimental para definir-se os rumos para a realização a longo prazo de novos experimentos mais sofisticados.

Com base nos resultados, o modelo pré-treinado do Inception v3, foi obtida uma taxa de acurácia de 30%. Considerando o número de epocas e instâncias utilizadas, existe uma grande possibilidade de melhorias nestes resultados, tendo em vista a capacidade de ampla captura dos resultados, e alternando o foco entre as diferentes classes, evitando vícios na rede, principalmente referênte a curvas.

Após este resultado foi adotada uma nova abordagem utilizando o mesmo conjunto de treinamento, porém neste momento foi realizado um aumento na taxa de erro para 0.00001 e o número de épocas para 1000, neste momento foi possivel observar uma melhora no resultado chegando a taxa de 40%.

Para uma terceira abordagem se utilizando dos parâmetros do primeiro experimento foi realizada uma captura extra com o objetivo de balancear os resultados, adicionando um total de 400 instâncias ao conjunto de treinamento existente focando principalmente em paradas e curvas foi possivel alcançar uma taxa de acurácia de 44%.

[2]https://colab.research.google.com/

5 CONCLUSÃO

Neste trabalho, a questão de variáveis de ambiente se mostrou um fator importante, considerando a necessidade de garantir o melhor aprendizado possível, apesar do ambiente comportado e este trabalho propor um protótipo de baixo custo, se mostrou desafiador.

O limite proposto de 1 segundo foi utilizado considerando o poder de processamento limitado em conjunto com uma baixa velocidade, de forma que a leitura de dados dos sensores de iluminação e de ultrassom fossem confiáveis, porém considerando que o método de aplicação na rede neural exigiu que os seus respectivos pesos fossem inseridos na imagem, houve a necessidade de arredondamento dos valores do sensor de proximidade para um número inteiro, o que em uma situação real pode influenciar negativamente a resposta da rede neural.

A necessidade de captura de dados de qualidade se mostrou importante considerando os problemas que foram encontrados durante o desenvolvimento deste trabalhando, começando de interferências de ambiente até as limitações de hardware. Foi utilizado um sistema distribuído com o objetivo de otimizar o processamento entre as partes envolvidas no processo através de comunicação sem fio, porém o ideal seria comunicação física considerando que durante os experimentos foi observado certa latência, ou mesmo perda de pacotes através do protocolo TCP IP.

Os modelos de Rede Convolucionais pré-treinados encontrados se mostram preparados para trabalhar com imagens em RGB, porém neste problema o ideal seria trabalhar com o formato de escala em cinza para a otimização, considerando que as cores não se mostram um fator relevante.

Para o problema foi proposto que a rede utilizasse as paredes do corredor como um delimitador de aprendizado. Porém para um problema mais complexo como um veículo autônomo de rua, o ideal seria a utilização de múltiplas redes convolucionais encarregadas de diferentes tarefas, considerando que no experimento existiram alguns dúvidas como se a rede seria capaz com o treinamento adequado identificar a presença de pedestres e ao mesmo tempo as paredes.

O fator tempo de certa forma contribuindo de forma negativa para o desenvolvimento deste trabalho, aonde o ideal além de verificar a aplicabilidade dos dados capturados em uma

rede convolucional, se investir no treinamento adequado buscando assim atingir uma taxa de acurácia alta.

A falta de tempo hábil para o treinamento adequado da rede, também influenciou negativamente o teste de campo em tempo real, considerando que apesar do software estar pronto, o treinamento realizado ainda era muito precoce e longe do ideal para a execução em tempo real.

Outro fator relevante é a necessidade de utilização de um hardware potente para o treinamento do modelo, independente de qual rede for escolhida, pois o fator tempo para o treinamento se mostra bastante relevante, considerando que o modelo proposto captura um quadro por segundo, as 3.600 imagens por sessão de captura podem se mostrar um desafio, considerando o tempo demasiado gasto apenas na etapa de pré-processamento.

Considerando o treinamento da rede, graças a plataformas como o Tensorflow[1] e o Keras este se mostra mais simples e acessível ao usuário, considerando a capacidade de execução destes algoritmos em um Hardware de CPU. A grande dificuldade encontrada foi a necessidade de adaptar as entradas de imagem em um formato legível para a rede neural artificial. O resultado de 30% apesar de baixo, se mostra animador considerando o total de instâncias utilizadas e o número de épocas para treinamento considerando a grande capacidade de extração de dados do módulo robótico. Esta melhoria de resultado foi observada em testes subsequentes com mudanças nos parametros de treinamento ou no número de instâncias de dados.

Apesar de todos os obstáculos encontrados no desenvolvimento, mostrou-se viável a construção de um módulo autônomo de baixo custo se utilizando da metodologia proposta, levando em consideração o problema onde este deverá enfrentar devido a fatores como eficiência energética, o terreno onde este será executado e a capacidade de um treinamento adequado.

5.1 TRABALHOS FUTUROS

O fator tempo para desenvolvimento deste trabalho considerando sua interdisciplinaridade foi um empecilho para a busca de melhores resultados. Futuramente espera-se a continuação do processo de captura considerando o uso de melhores técnicas de

[1] https://www.tensorflow.org/?hl=pt-br

pré-processamento em tempo real com o objetivo de facilitar o seu processamento além da busca por melhores resultados.

Considerando que a metodologia prevê a capacidade de execução do problema em tempo real, com áreas especificas para o treinamento da RNA, assim garantindo testes como a capacidade de generalização da rede, não existe nenhum empecilho para que não seja feito um treinamento com uma grande base de dados.

Além da continuidade do trabalho proposto, existe a hipótese de utilização de mais de uma Inteligência Artificial em conjunto, com objetivo de observar determinadas características do ambiente, como a existência de pedestres, construindo assim uma hierarquia com foco em segurança.

REFERÊNCIAS

AL-QIZWINI, M.; BARJASTEH, H. A.-Q. I.; RADHA, H. Deep learning algorithm for autonomous driving using googlenet. In: **2017 IEEE Intelligent Vehicles Symposium (IV)**. [s.n.], 2017. Disponível em: <https://www.egr.msu.edu/˜alqizwin/IV2017_0203_FI.pdf>.

ARSENOVIC; MARKO. Facetime—deep learning based face recognition attendance system. 10 2017.

BATEUX, Q. et al. Visual servoing from deep neural networks. **CoRR**, abs/1705.08940, 2017. Disponível em: <http://arxiv.org/abs/1705.08940>.

BATISTA, G. E. de A. P. A. **Pré-processamento de Dados em Aprendizado de Máquina Supervisionado**. Doutorado, 2003.

BOJARSKI, M.; TESTA, D. D. D. D.; FIRNER, B. End to end learning for self-driving cars. **CoRR**, abs/1604.07316, 2016. Disponível em: <http://arxiv.org/abs/1604.07316>.

BORCHARDT, F. Neural network computing and natural language processing. **CALICO Journal**, v. 5, n. 4, 2013. ISSN 2056-9017. Disponível em: <https://journals.equinoxpub.com/index.php/CALICO/article/view/23553>.

DENG, L. A tutorial survey of architectures, algorithms, and applications for deep learning. **APSIPA Transactions on Signal and Information Processing**, Cambridge University Press, January 2014. Disponível em: <https://www.microsoft.com/en-us/research/publication/a-tutorial-survey-of-architectures-algorithms-and-applications-for-deep-learning/>.

FRANÇA, M. T. A. de; FRANÇA, M. G. A. de. A utilização de visão robótica para captura de dados do ambiente em tempo real com o protótipo de baixo custo. In: . [S.l.: s.n.], 2018.

GOLDBERG, Y. A primer on neural network models for natural language processing. **CoRR**, abs/1510.00726, 2015. Disponível em: <http://arxiv.org/abs/1510.00726>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning - Neural Networks and Learning Machines**. [S.l.: s.n.], 2016.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. [S.l.: s.n.], 2001.

HAYKIN, S. **Neural Networks and Learning Machines**. [S.l.: s.n.], 2009.

HEBB, D. **The Organization of Behavior**. [S.l.: s.n.], 1949.

HUCHINSON, S.; CASTAñO, A. Visual compliance: task-directed visual servo control. 1994.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>.

LECUN, Y.; BOSER, J. S. D. B.; HENDERSON, D. Backpropagation applied to handwritten zip code recognition. **Neural Computation**, v. 1, n. 4, p. 541–551, Dec 1989. ISSN 0899-7667.

LECUN, Y.; BOTTOU, Y. B. L.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, Nov 1998. ISSN 0018-9219.

MINSKY, M.; PAPERT, S. **Perceptrons**. Cambridge, MA: MIT Press, 1969.

MITCHELL, T. M. **Machine Learning**. [S.l.: s.n.], 1997.

NIELSEN, M. A. misc, **Neural Networks and Deep Learning**. Determination Press, 2018. Disponível em: <http://neuralnetworksanddeeplearning.com/chap6.html>.

PROVOST, F.; KOHAVI, R. **Machine Learning - On Applied Research in Machine Learning**. [S.l.: s.n.], 1998. 127–132 p.

PUMSIRIRATAND, A.; YAN, L. Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. 2018.

RICHARDS, J. W.; BRINK, H.; FETHEROL, M. **Real World Machine Learning**. [S.l.: s.n.], 2014.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, 1958.

RUMELHART, D. E.; MCCLELLAND, J.; RUMELHART. [S.l.: s.n.], 1986.

RUSSEL, S. J.; NOVIG, P. **Artificial Intelligence: a Modern Approach**. [S.l.: s.n.], 1994. 46 p.

SILVA, I. N. da; FLAUZINO, R. A. **Redes neurais artificiais para engenharia e ciências aplicadas**. [S.l.: s.n.], 2010.

SILVA, M. P. F. L. Detecção de instrumentos musicais com redes neurais profundas. 2017. Disponível em: <https://app.uff.br/riuff/handle/1/5810?mode=full>.

SIMON, P. **The Business Case for Big Data: The Business Case for Big Data.** [S.l.: s.n.], 2013.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. 2014.

Stanford Vision Lab. **ImageNet Large Scale Visual Recognition Competition (ILSVRC)**. 2015. `http://image-net.org/challenges/LSVRC/`. Acesso em: 09-12-2018.

SZEGEDY, C. et al. Going deeper with convolutions. In: . [s.n.], 2014. abs/1409.4842. Disponível em: <http://arxiv.org/abs/1409.4842>.

SZEGEDY, C. et al. Rethinking the inception architecture for computer vision. **CoRR**, abs/1512.00567, 2015. Disponível em: <http://arxiv.org/abs/1512.00567>.

TUSHAR, A. Making sense of hidden layer information in deep networks by learning hierarchical targets. **CoRR**, abs/1505.00384, 2015. Disponível em: <http://arxiv.org/abs/1505.00384>.

ZHAI, S.; CHENG, W. L. Y.; ZHANG, Z. Doubly convolutional neural networks. In: . [s.n.], 2016. abs/1610.09716. Disponível em: <http://arxiv.org/abs/1610.09716>.

ÖZGÜNER Ümit; STILLER, C.; REDMILL, K. Systems for safety and autonomous behavior in cars: The darpa grand challenge experience. 2007.