Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Guilherme Moser Manerichi

Sistema de Classificação de Terrenos baseado em Redes Neurais e Visão Computacional

Blumenau
2019

Guilherme Moser Manerichi

Sistema de Classificação de Terrenos baseado em Redes Neurais e Visão Computacional

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Título de Engenheiro de Controle e Automação.
Orientador: Profª. Drª. Janaina Gonçalves Guimarães

Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Blumenau
2019

Guilherme Moser Manerichi

Sistema de Classificação de Terrenos baseado em Redes Neurais e na Visão Computacional

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação.

Comissão Examinadora

Profª. Drª. Janaina Gonçalves Guimarães
Universidade Federal de Santa Catarina
Orientador

Prof. Dr. Leonardo Mejia Rincon
Universidade Federal de Santa Catarina

Prof. Dr. Marcos Vinicius Matsuo
Universidade Federal de Santa Catarina

Blumenau, 1 de fevereiro de 2019

Dedico este trabalho a todos aqueles que me incentivaram
e sempre me motivaram durante todo este longo caminho:
A minha família e a todos os amigos que estiveram
comigo durante os bons e os maus momentos.

Agradecimentos

Agradeço a minha família por ter me apoiado nos momentos mais difíceis de minha graduação, e terem disposto de esforços incomensuráveis para que pudesse completar uma das mais importantes etapas de minha vida. Agradeço muito a Professora Janaina, que em todos os momentos que mais precisei de ajuda sempre esteve presente me dando conselhos e me guiou rumo à conclusão do curso.

"Even the smallest person can change the course of the future."
(Galadriel)

Resumo

O objetivo deste projeto foi implementar um protótipo de um sistema simples de classificação de terrenos para um pequeno veículo terrestre baseado em uma rede neural artificial multicamadas do tipo *feedfoward*. A rede neural serve como sistema de classificação e é vinculada a um sistema de tratamento de imagens, que captura imagens do terreno e retira descritores para a alimentação de dados do sistema com a rede. As técnicas de tratamento e a arquitetura final da rede neural foram implementadas e aperfeiçoadas através do MATLAB, pois mostra-se como uma ferramenta poderosa no processamento de matrizes e apresenta um conjunto grande de bibliotecas que facilitam a prototipagem.

O sistema de classificação foi embarcado a um veículo de testes, responsável por comportar todo o sistema além de obter as imagens. Por fim realizada uma análise e comparação com outros métodos de classificação de terrenos.

Palavras-Chave: 1. Rede Neural Artificial. 2. Classificação de terrenos. 3. Visão Computacional. 4. Perceptron Multicamadas 5.Descritores Estatísticos

Abstract

The objective of this project was to implement a prototype of a simple terrain classification system for a small terrestrial vehicle based on a feedfoward multilayer artificial neural network. The neural network is used as a classification system and is linked to an image processing system, which captures terrain images and removes descriptors to feed data to the system into the network. The treatment techniques and the final architecture of the neural network were implemented and improved through MATLAB, as it is a powerful tool in matrix processing and presents a large set of libraries that facilitate prototyping.

The classification system was embedded to a test vehicle, responsible for handling the whole system in addition to getting the images. Finally, an analysis and comparison with other land classification methods was made.

Keywords: 1. Artificial Neural Network. 2. Terrain classification. 3. Computacional Vision. 4. Perceptron Multilayer 5.Statistical Descriptors

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

RNA	*Redes Neurais Artificiais*
PMC	*Perceptron Multicamadas*
EQM	*Erro Quadrático Médio*
FoV	*Field of View*

Sumário

1 Introdução

Com a constante evolução dos carros inteligentes, questões como segurança e autonomia ganharam grande destaque. Com o intuito de propor soluções adequadas para essas questões, a análise apropriada do tipo de terreno que o carro se desloca é de suma importância para que se realize adaptações para o veículo, que resultará em benefícios tanto em relação ao consumo, e principalmente quanto em relação à segurança do veículo. Em 2013 a *Land Rover*, uma empresa com sede na Inglaterra, desenvolveu um sistema denominado como *Terrain Response 2*. Este sistema, a partir do reconhecimento do terreno em que o veículo se desloca, permite um ajuste automático de seu movimento para uma melhor adaptação ao meio em que está inserido, possibilitando ajustes na caixa de câmbio, na resposta do motor, na resposta do acelerador e no sistema de freios ABS [1].

Várias técnicas podem ser utilizadas para o reconhecimento e classificação de terrenos. Essas técnicas se diferenciam uma das outras principalmente considerando parâmetros como: o tipo de informação do terreno que será utilizada para identificá-lo; o tipo de sistema que será responsável pelo reconhecimento e classificação; e a forma como as informações serão utilizadas pelo sistema.

Para a obtenção de dados relativos ao terreno, as técnicas abordadas podem depender do grau de robustez e complexidade do sistema de reconhecimento, podendo partir da utilização de câmeras, sensores de aceleração, Lidar 3D e até mesmo utilizando fusão de dados onde mais sensores são utilizadas em conjunto [2, 3]. A Visão Computacional apresenta-se como uma ótima alternativa ao oferecer técnicas de pré-processamento capazes de realçar características importantes de cada terreno, possibilitando obter dados mais adequados para o reconhecimento.

Um dos procedimentos para classificação e reconhecimento bastante utilizada são as Redes Neurais Artificiais (RNA), cada vez mais popularizadas devido a quantidade de pesquisas na área, e com uma gama de aplicações bastante ampla. Uma das principais aplicações da RNA é no reconhecimento e classificação de padrões para classes previamente definidas [4].

Portanto, levando em consideração o contexto apresentado, reconhecer e classificar o tipo de terreno onde um veículo se encontra apresenta-se como uma ferramenta bastante útil na tomada de decisões, tanto para veículos tradicionais quanto para veículos que são usados na robótica móvel.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é criar um protótipo de um sistema simples de classificação de terrenos baseado em um veículo terrestre equipado com uma câmera e um computador portátil, capaz de classificar cinco tipos diferentes de terreno utilizando uma rede neural artificial multicamadas do tipo *feedfoward*. A rede neural estará vinculada a um sistema de tratamento de dados com o objetivo de retirada de descritores de imagens obtidas pelo veículo e que serão utilizadas para a alimentação da rede neural.

1.1.2 Objetivos específicos

Com o intuito de alcançar o objetivo geral os seguintes objetivos específicos foram executados:

- Revisar os métodos utilizados para a classificação de terrenos;
- Implementar técnicas de tratamento para a retirada de dados de imagens;
- Encontrar ótima arquitetura de rede neural para a aplicação;
- Embarcar uma rede neural em um *hardware* para realizar a classificação de terrenos;
- Desenvolvimento do protótipo do sistema de classificação.

2 Revisão de Literatura

2.1 Redes Neurais Artificiais - RNA

A ideia de construir um mecanismo dotado de certo grau de inteligência sempre foi um objetivo de pesquisadores de diversas áreas da tecnologia. Mesmo que o conceito de Redes Neurais Artificiais tenha surgido nos anos de 1940, quando Warren McCulloch e Walter Pitts apresentaram a primeira representação de um neurônio artificial, a primeira rede propriamente implementada veio a acontecer por intermédio de Frank Rosenblatt, entre 1957 e 1958, com o desenvolvimento do primeiro neurocomputador, idealizando a Rede Perceptron [4].

As redes neurais artificiais são modelos computacionais do sistema nervoso biológico de seres vivos. Podem ser definidas como um conjunto de unidades de processamento, caracterizadas por neurônios artificiais, que são interligados por um grande número de interconexões (sinapses artificiais) apresentadas como pesos sinápticos.

As RNA são dotadas da capacidade de aprender e realizar a manutenção do conhecimento, baseado em seu treinamento por meio da apresentação de informações relevantes à sua aplicação [4].

2.1.1 Neurônio Artificial

A estrutura do Neurônio Artificial foi construída baseada nos modelos conhecidos de sistemas nervosos biológicos. Podem ser denominadas unidades processadoras que representam os neurônios biológicos de forma simplificada.

Os neurônios artificias são unidades não-lineares, que fornecem na maior parte das vezes saídas contínuas. Os neurônios recebem uma informação de entrada, processam as informações de acordo com sua função operacional e fornecem uma saída. O modelo mais utilizado até hoje, é o modelo proposto por McCulloch e Pitts (1943), e pode ser representado pela Figura 1.

As variáveis x_i representam as informações de entrada da rede, w_i são os pesos sinápticos, que são os valores que ponderam cada uma das entradas da rede, θ é um limiar de ativação que determinará se o valor ponderado poderá gerar um valor de disparo para a saída do neurônio e, por fim, a função $g(.)$ que é uma função de ativação limitará a saída conforme a função escolhida. Na figura 1, $\mathbf{u}$ é o potencial de ativação, se o seu valor é positivo o neurônio produzirá um potencial excitatório, caso negativo será um potencial inibitório [4].

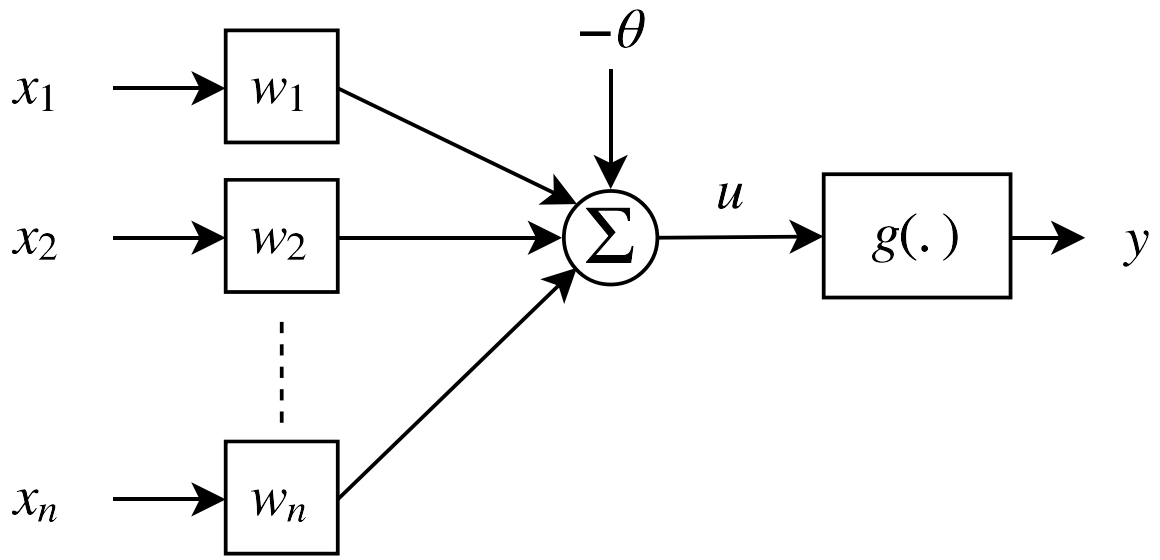

Figura 1 – Neurônio Artificial [5].

Matematicamente o neurônio pode ser representado pelas seguintes equações:

$$u = \sum_{i=1}^{n}(w_i \cdot x_i) - \theta \tag{2.1}$$

$$y = g(u) \tag{2.2}$$

Apresentado o neurônio artificial, três itens são de extrema importância para definir o tipo de rede neural que pode ser implementada:

- *Arquitetura*: A forma como os neurônios serão organizados para a construção da rede, se será em camadas, totalmente interconectados, parcialmente interconectados, com ou sem realimentação entre outros aspectos construtivos;
- *Aprendizagem*: Qual o método que será utilizado na determinação dos pesos dos neurônios que representam o conhecimento da rede neural;
- *Função de ativação*: A função de transferência que determina o intervalo de valores da saída do neurônio, ou seja, quando o neurônio ativará sua saída.

2.1.2 Redes Neurais Artificiais Multicamadas

As redes Perceptron foram apresentadas no ano de 1958, pelo psicólogo americano Frank Rosenblatt, propondo um modelo computacional dos neurônios de McCulloch–Pitts. Inicialmente, o modelo proposto apresentava uma rede caracterizada por somente duas

camadas de processamento, uma de entrada que pode apresentar n valores e uma de saída. O neurônio perceptron recebe este nome por conta de sua função de ativação que é do tipo degrau, assumindo valores de 0 a 1. Foi inicialmente utilizada para a identificação de padrões geométricos.

Apesar de se apresentar um modelo interessante e simples, a configuração da rede Perceptron de duas camadas possui uma grande limitação, sua capacidade de diferenciar padrões é limitada a situações onde duas classes sejam linearmente separáveis. No exemplo abaixo, estão representadas respectivamente as combinações lógicas *AND* e *XOR*. Através da figura da combinação *AND* é facilmente perceptível que é possível separar suas saídas de forma linear, baseado nas entradas x_1 e x_2. Já para a combinação *XOR* não é linearmente separável, o que prova a limitação da rede Perceptron simples. [4]

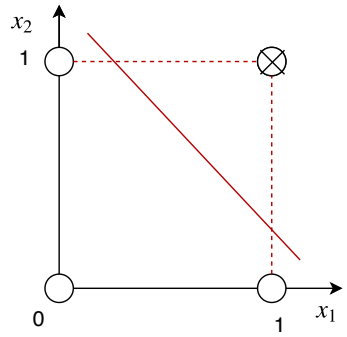

Figura 2 – Combinação lógica **AND**.

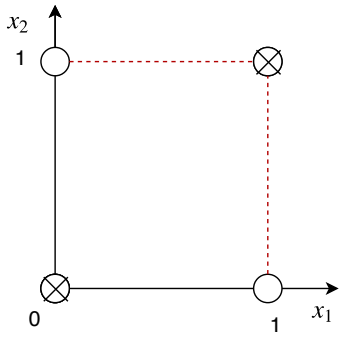

Figura 3 – Combinação lógica **XOR**.

Por conta da simplicidade e limitação da rede Perceptron proposta por Rosenblatt, foi formulada uma rede que pudesse realizar a identificação de padrões que não fossem linearmente separáveis. A rede criada para complementar a Perceptron foi a **Rede Neural Artificial Multicamadas**, normalmente do tipo *feedfoward*. Muitas vezes essa configuração é denominada de **Perceptron Multicamadas - PMC** mesmo que a rede não seja composta por neurônios perceptron. A rede multicamadas têm esse nome por possuírem pelo menos uma camada intermediária, ou seja, pelo menos três camadas de processamento contando com a de entrada e a de saída.

Possuem uma grande aplicabilidade e vêm sendo abordada por várias áreas do conhecimento. Algumas de suas aplicações são aproximação universal de funções, reconhecimento de padrões, identificação e controle de processos, previsão de séries temporais, otimização de sistemas entre muitas outras aplicações.

Um exemplo de PMC com duas camadas escondidas de neurônios pode ser visto na figura 4. Cada círculo representa um neurônio artificial e pode ser interpretado como um encapsulamento da figura 1. Cada flecha que liga os círculos da figura 4 representa um peso sináptico.

Uma rede *feedfoward* significa que a informação é apresentada na camada de entrada e passa por vários neurônios até a camada de saída, responsável por apresentar a resposta

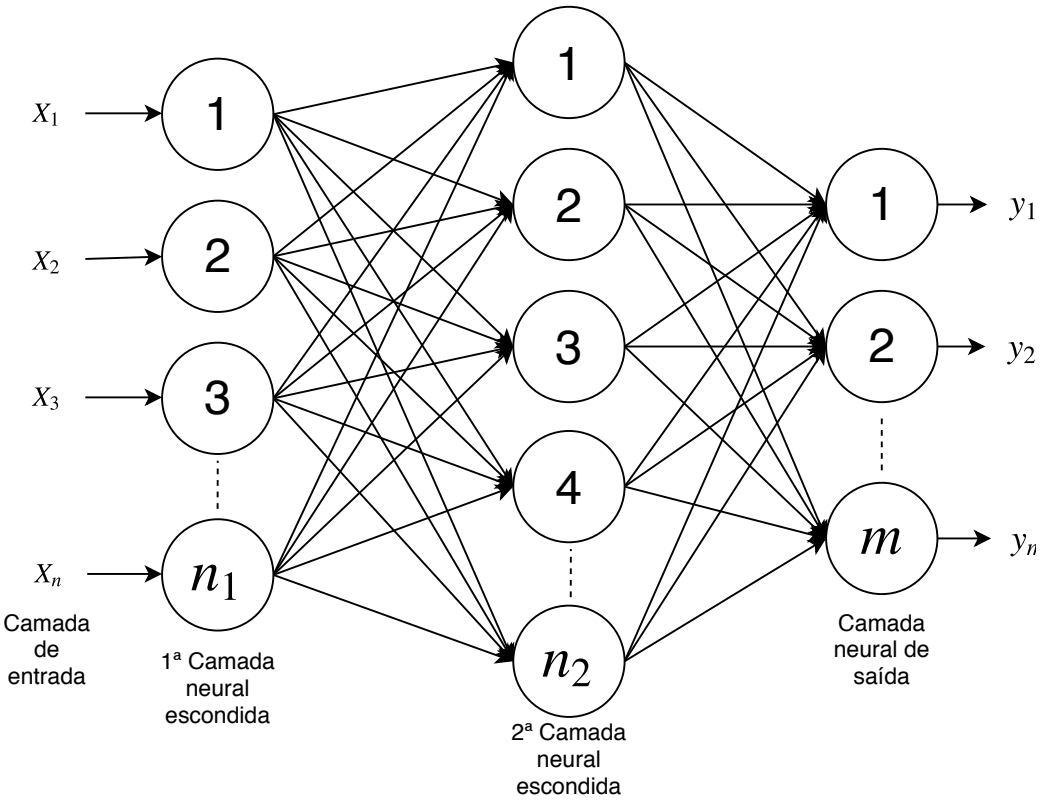

Figura 4 – Rede Neural Multicamadas.

desejada.

2.1.2.1 Processo de Aprendizagem

Para que a rede possa ser treinada é necessário o uso de um algoritmo de aprendizagem, sendo na PMC o mais utilizado é o método *Backpropagation*, também conhecido por **Algoritmo de Retropropagação do Erro**. O processo tem por objetivo a determinação dos pesos sinápticos e limiares de ativação de todos os neurônios inseridos na rede por meio de um processo recursivo.

O treinamento é supervisionado, pois ao ser apresentado o vetor de dados de entrada, também é fornecida a saída desejada para a rede, para que se calcule o erro, o qual é utilizado pelo algoritmo de retropropagação do erro para direcionar a adaptação/atualização dos pesos e do *bias*. Ao iniciar o treinamento, os pesos sinápticos normalmente são iniciados em valores aleatórios para aumentar a velocidade de convergência.

O processo se inicia apresentando um vetor de entradas x_i à rede, que percorre os neurônios até chegar à saída, ou seja, realiza o processo de *feedfoward*, onde o conjunto de entrada passa por cada neurônio até atingir a saída.

A próxima etapa é chamada de *backward*, onde o valor na saída da rede será comparado com um vetor desejado d_i de resposta apresentado para a rede. É nesta etapa onde ocorre a

correção dos pesos sinápticos e limiares, onde o valor do erro é ajustado em cada neurônio, sendo o processo iniciado da última camada de processamento para a primeira.

Esse processo de *feedfoward* e do *backward* realizado sucessivas vezes, tem por objetivo atingir valores de treinamento que possa tornar a rede adequada para a utilização.

É necessário que se utilize uma função custo para que se saiba quando o erro total da rede entrará em um intervalo ideal desejado. Esta função na maior parte das vezes é o **erro quadrático médio - EQM** e ele pode ser facilmente calculado pela equação 2.3:

$$E(k) = \frac{1}{2}\sum_{j=1}^{n}(d_j(k) - Y_j(k))^2 \tag{2.3}$$

Onde na equação $E(k)$ significa o erro considerando a k-ésima amostra de treinamento, $Y_j(k)$ é a saída do j-ésimo neurônio da camada de saída da k-ésima amostra e d_j é o respectivo valor desejado [5, 4].

A atualização das matrizes pesos sinápticos é realizada em duas partes, a primeira é o ajuste dos pesos na camada de saída e a segunda parte é o ajuste dos pesos nas camadas intermediarias. A primeira parte pode ser calculada pelos passos a serem descritos.

O algoritmo aplica o operador gradiente ao erro em relação à matriz W_{ji}. Utilizando-se da definição do gradiente e da regra da cadeia, têm-se:

$$\nabla E^{(m)} = \frac{\partial E}{\partial W_{ji}^{(m)}} = \frac{\partial E}{\partial Y_j^{(m)}} \cdot \frac{\partial Y_j^{(m)}}{\partial I_j^{(m)}} \cdot \frac{\partial I_j^{(m)}}{\partial W_{ji}^{(m)}} \tag{2.4}$$

Onde o índice **m** representa a última camada de processamento, ou seja, a de saída. O peso sináptico $W_{ij}^{(m)}$ é tratado como uma matriz que comporta todos os pesos de todos os neurônios na camada neural de saída, onde os índices representam o j-ésimo neurônio da camada calculada em relação ao i-ésimo neurônio da camada imediatamente anterior. $I_j^{(m)}$ representa um vetor cujo os valores são entradas do j-ésimo neurônio da camada m ponderado pelo peso sináptico correspondente. $Y_i^{(m)}$ é a saída do i-ésimo neurônio da camada m.

Por meio de definições matemáticas, podemos representar algumas fórmulas que ajudarão no entendimento. Matematicamente $I_j^{(m)}$ e $Y_i^{(m)}$ podem ser representados por:

$$I_j^{(m)} = \sum_{i=0}^{n_{m-1}} W_{ij}^{(m)} \cdot Y_i^{(m-1)} \tag{2.5}$$

$$Y_i^{(m)} = g(I_j^{(m)}) \tag{2.6}$$

Onde n_{m-1} diz respeito ao neurônio da camada anterior a **m**. A função $\mathbf{g(.)}$ representa a função de ativação aplicada.

Disto podemos obter os termos da equação 2.10, baseados na diferenciação das equações 2.5, 2.6 e 2.3:

$$\frac{\partial I_j^{(m)}}{\partial W_{ji}} = Y_i^{m-1} \tag{2.7}$$

$$\frac{\partial Y_j^{(m)}}{\partial I_j^{(m)}} = g'(I_j^{(m)}) \tag{2.8}$$

$$\frac{\partial E}{\partial Y_j^{(m)}} = -(d_j - Y_j^{(m)}) \tag{2.9}$$

Onde $g'(.)$ denota a derivada primeira da função de ativação e a equação 2.9 representa o erro na última camada. Substituindo as equações **??** em 2.10 chegamos a uma representação do erro em relação aos pesos sinápticos:

$$\frac{\partial E}{\partial W_{ji}^{(m)}} = -(d_j - Y_j^{(m)}) \cdot g'(I_j^{(m)}) \cdot Y_i^{m-1} \tag{2.10}$$

Logo o ajuste da matriz pode ser definido por 2.11, e deve ser realizado na direção oposta ao gradiente, pois pretende-se reduzir a equação de erro:

$$\Delta W_{ji}^{(m)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{m}} \tag{2.11}$$

$$\delta_j^{(m)} = (d_j - Y_j^{(m)}) \cdot g'(I_j^{(m)}) \tag{2.12}$$

$$\Delta W_{ji}^{(m)} = \eta \cdot \delta_j^{(m)} \cdot Y_i^{(m-1)} \tag{2.13}$$

Onde η é uma taxa de aprendizagem do próprio algoritmo, $Y_i^{(m-1)}$ é a saída do i-ésimo neurônio da camada anterior e δ_j é o gradiente local em relação ao j-ésimo neurônio da camada [4].

Por fim pode ser escrito de forma iterativa na equação 2.14

$$W_{ji}^{(m)}(k+1) = W(m)_{ji}(k) + \eta \cdot \delta_j^{(m)} \cdot Y_i^{(m-1)} \tag{2.14}$$

Por fim, enquanto o **EQM** da equação 2.3 não estiver no intervalo desejado, o algoritmo recalculará recursivamente os pesos sinápticos, a essa recursividade dá-se o nome de **épocas**.

O ajuste dos pesos sinápticos nas camadas intermediárias, ocorre de forma parecida com o ajuste da camada de saída. A diferença é que os neurônios das camadas intermediárias não tem acesso à um valor desejado de saída, por isso utilizam de estimativas dos erros de saída dos neurônios da camada imediatamente posterior, que já foram ajustados nesta etapa.

Afim de demonstrar isto, a equação 2.10 é aplicada para a camada anterior, gerando a equação 2.15:

$$\nabla E^{(m-1)} = \frac{\partial E}{\partial W_{ji}^{(m-1)}} = \frac{\partial E}{\partial Y_j^{(m-1)}} \cdot \frac{\partial Y_j^{(m-1)}}{\partial I_j^{(m-1)}} \cdot \frac{\partial I_j^{(m-1)}}{\partial W_{ji}^{(m-1)}} \tag{2.15}$$

Da mesma forma que a anterior, calcula é realizada a substituição das derivadas parciais, onde existe a variação em relação a equação 2.9, e pode ser entendida como:

$$\frac{\partial E}{\partial Y_j^{(m-1)}} = \sum_{k=1}^{n_m} \frac{\partial E}{\partial I_k^{(m)}} \cdot W_{kj}^{(m)} \tag{2.16}$$

Onde a parcela $\frac{\partial E}{\partial I_k^{(m)}}$, pode ser entendida como o gradiente $-\delta_k^{(m)}$ da camada posterior, sendo reescrito da seguinte forma:

$$\frac{\partial E}{\partial Y_j^{(m-1)}} = -\sum_{k=1}^{n_m} \delta_k^{(m)} \cdot W_{kj}^{(m)} \tag{2.17}$$

Substituindo-se em 2.15, obtém-se:

$$\frac{\partial E}{\partial W_{ji}^{(m-1)}} = -(\sum_{k=1}^{n_m} \delta_k^{(m)} \cdot W_{kj}^{(m)}) \cdot g'(I_j^{(m-1)}) \cdot Y_i^{m-2} \tag{2.18}$$

Desta forma atinge-se da mesma forma que para o cálculo para a saída, as equações a seguir:

$$\Delta W_{ji}^{(m-1)} = -\eta \cdot \frac{\partial E}{\partial W_{ji}^{(m-1)}} \tag{2.19}$$

$$\delta_j^{(m-1)} = (\sum_{k=1}^{n_m} \delta_k^{(m)} \cdot W_{kj}^{(m)}) \cdot g'(I_j^{(m-1)}) \tag{2.20}$$

$$\Delta W_{ji}^{(m-1)} = \eta \cdot \delta_j^{(m-1)} \cdot Y_i^{(m-2)} \tag{2.21}$$

Por fim pode ser escrito de forma iterativa na equação 2.22

$$W_{ji}^{(m-1)}(k+1) = W_{ji}^{(m-1)}(k) + \eta \cdot \delta_j^{(m-1)} \cdot Y_i^{(m-2)} \tag{2.22}$$

Este procedimento possibilita o calculo dos pesos para as camadas ocultas, sempre aplicando da última para a primeira, até chegar-se à camada de entrada. Esse processo todo acontece de forma iterativa até que os pesos atinjam um erro mínimo.

2.1.3 Aplicações

As redes neurais são aplicadas nas mais diversas áreas do conhecimento, na computação, matemática, engenharia, medicina, química, economia entre muitas outras áreas. Alguns exemplos podem ser citados como grandes aplicações [4]:

- **Classificação de padrões:** Uma das aplicações mais antigas em redes neurais, a classificação de padrões se faz uma ótima aplicação, pois pode ser aplicada à quase qualquer área do conhecimento humano. Um exemplo na eletrônica de potência, que as redes são utilizadas para classificar a fonte de correntes harmônicas no sistema de distribuição de energia elétrica;

- **Previsão de ações no mercado financeiro:** Na área de finanças e economia existem muitos problemas que possuem um comportamento não-linear, e a partir disso redes neurais são largamente empregadas.

- **Identificação de anomalias em imagens médicas:** Esta é uma área muito importante em que as RNA estão inseridas. São utilizadas para classificações e predições de câncer baseando-se no perfil genético do paciente;

- **Produção de novos polímeros:** Na área química, as RNA são muito utilizadas para a obtenção de novos compostos poliméricos;

- **Controle de processos:** Identificar ações de controle que permitam o processo atingir requisitos de qualidade, eficiência e segurança na aplicação. Pode ser visto em várias áreas, aeronáutica, robótica entre outras;

2.2 Visão Computacional

Visão computacional é um campo da ciência da computação que desenvolve técnicas que permitem os sistemas computacionais verem, identificarem e processarem imagens de uma forma análoga a visão humana. A visão computacional tem um grande impacto na robótica móvel, pois permite que que os robôs possam reconhecer objetos e se locomover de forma otimizada [6].

Uma parte fundamental da visão computacional se chama **processamento de imagem**. O processamento de imagem traz várias técnicas que permitem que seja possível detectar o maior número possível de descritores de uma imagem. Os descritores podem ser utilizados para analizar, descrever e relacionar imagens, sendo um passo muito importante para a otimização de dados.

2.2.1 Processamento de Imagens e obtenção de *Features*

O processamento de imagem tem por objetivo convertê-la em uma forma que possibilite alguma análise ou para que se possa adaptá-la para determinado fim. É uma etapa fundamental na extração de *features*. Algumas operações que podem ser citadas é a correção de exposição e balanceamento de cor, redução de ruído, espaço de cores ou filtros entre outros [6, 7].

A imagem pode ser entendida como uma matriz de elementos de *pixel* que possuem um intervalo de valor de acordo com o espaço de cores e formato que se encontra. Através de operações matemáticas sobre as matrizes é possível realizar transformações para adequar as imagens ao necessário. Na Figura 6 pode-se observar a transformação do espaço de cores e o redimensionamento da imagem, demonstrando um pouco do processamento de imagem:

Figura 5 – Imagem Lena - 512x512 pixels.

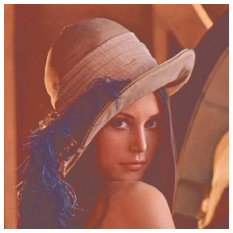

Figura 6 – Imagem Lena - 256x256 pixels.

Muitos livros e trabalhos na área acadêmica abordam diferentes tipos de pré-processamento. Na escolha das técnicas de processamento é importante ressaltar informações pertinentes

sobre as condições do processo a ser realizados, questões sobre variação de tamanho, luminosidade, rotação, dentre outros aspectos de suma importância para a otimização da escolha das técnicas. Em muitos casos também é necessário realizar a segmentação das imagens por conta de vários objetos diferentes em cena, em outros casos isso não se faz necessário já que o objeto a ser reconhecido engloba toda a cena.

No exemplo abaixo, é possível ver que o terreno de ambas as imagens é grama, porém para o reconhecimento da imagem da esquerda seria necessário realizar a segmentação e retirar o cenário ao fundo, já na direita o processo é mais simples.

Figura 7 – Grama com outros objetos.

Figura 8 – Grama.

As *features* são medidas obtidas das imagens, normalmente escalares que podem representar medidas como as coordenadas de um objeto ou tamanho, que resumem as informações antes expressas em vários pixels. As *features* reduzem a quantidade de dados e fornecem informações importantes na descrição de imagens [6].

Uma das técnicas bastante aplicadas para a obtenção de *features*, são as Matrizes de Co-ocorrência de níveis de cinza (GLCM) para o cálculo dos parâmetros de Haralick. Esta é uma técnica bastante antiga e que demanda certo grau de processamento por conta dos cálculos necessários. Os parâmetros de Haralick permitem que se retirem aproximadamente 14 descritores, sendo eles, energia, contraste, correlação, soma dos quadrados, momento diferencial inverso, soma da média, soma da variância, soma da entropia, entropia, diferença , diferença da entropia, medidas da correlação, e máximo coeficiente de correlação. Estes descritores são compostos em grande parte, por descritores estatísticos, que apresentam bons resultados e são de certa forma simples de serem calculados[8].

Outra técnica de extração de *features* é o *Local Binary Patterns* (LBP). A LBP é essencialmente um descritor local de textura. A sua combinação com filtros de Gabor é largamente reconhecida como um ótimo método de obtenção de features, porém o método é bastante sensível à variações de iluminação o que deve ser levado em conta na hora de sua escolha [9].

2.3 Métodos de Classificação de Terrenos

Existem vários métodos de obtenção de dados para o reconhecimento de terrenos, o que pode ser feito usando fusão de dados por meio de vários sensores, ou até mesmo baseado em somente um tipo de sensor, isso dependerá das técnicas utilizadas ou da proposta do trabalho. Os métodos listados servem para a obtenção dos dados pertinentes na classificação de terrenos, sendo necessário a escolha de um procedimento que permita relacionar os dados e encontrar padrões.

2.3.1 LIDAR 3D

O LIDAR é um sensor que gera uma nuvem de pontos 3D correspondentes a reflexão de um laser em um objeto, gerando assim a possibilidade da reconstrução do ambiente em 3D. Normalmente o LIDAR é utilizado em conjunto com algum outro sensor para classificar terrenos, como uma câmera RGB como complemento.

O LIDAR nesta aplicação pode ser utilizada para dividir o terreno em uma grade, e cada célula da grade retirar informações como altura dos pontos da nuvem, distribuição e intensidade dos pontos [10].

2.3.2 Sensor de Aceleração

O outro método que pode ser utilizado, é o uso de sensores de aceleração. Sensores de aceleração permitem a obtenção da variação da velocidade de um corpo. Ao fixar um sensor de aceleração à um veículo, é possível descobrir-se as variações de velocidade ao qual está submetido em qualquer um dos eixos. Estes dados possuem características muito distintas dependendo de qual terreno estão localizados, e a partir disto é realizada a classificação [11].

2.3.3 Câmera

Câmeras RGB são utilizadas no reconhecimento e classificação de terrenos de forma bastante ampla. Existem diversas técnicas de extração de *features* ao se trabalhar com imagens, o que permite uma vasta quantidade de descritores [2].

3 Metodologia

Neste capítulo será apresentada a metodologia utilizada no desenvolvimento do projeto. Estará organizado nas seções da seguinte forma; na seção 3.1 é apresentada uma topologia que descreve o desenvolvimento do trabalho, na seção 3.2 descreve como foi montado o banco de dados para o trabalho, na seção 3.3 a escolha das técnicas de pré-processamento serão apresentadas, na seção 3.4 mostra arquitetura para a rede e por fim na seção 3.5 ilustrará o processo de seleção dos códigos para a implementação.

3.1 Topologia do Trabalho

O desenvolvimento do trabalho pode ser divido em três partes principais, a obtenção do banco de dados, o tratamento e retirada de descritores das imagens para por fim o treinamento e utilização da rede neural. A figura 9 apresenta o caminho percorrido pela imagem capturada pela câmera até a sua classificação.

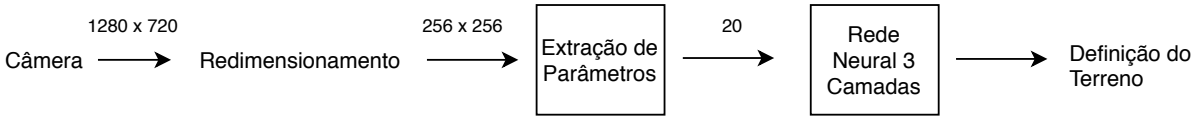

Figura 9 – Topologia do projeto.

Sobre as flechas da figura 9, é representado a quantidade de dados de uma etapa para a outra.

3.2 Banco de Dados

A montagem do banco de dados foi realizada com base nos terrenos propostos para o reconhecimento. O banco de dados construído para o treinamento da rede, foi capturado por uma câmera RGB posicionada a uma altura de aproximadamente 30cm do chão e formando um ângulo de 90°com o solo restringindo o ângulo da imagem para o tratamento.

Todas as imagens contidas no banco foram inteiramente coletadas para este trabalho. Com o auxílio de uma câmera as imagens foram capturadas procurando manter-se uma variabilidade nos terrenos.

Esta posição para qual foi escolhida para as imagens do banco de dados, deve-se justamente por conta do posicionamento do sistema de visão no veículo de testes, fazendo com que as imagens do banco sejam próximas das que são capturadas no veículo. Isto deve ser feito para que o treinamento da rede seja o mais confiável que se possa ser, pois

ao alterar a posição da captura no veículo ou do banco de dados, existirão inconsistências nos descritores de imagem e por consequência no treinamento das imagens.

Foram escolhidos cinco tipos de terreno para implementar o sistemas de classificação: grama, areia, asfalto, pavimento e pedra. O critério utilizado para determinar os cinco diferentes tipos de terreno foi exatamente a diferença presente um em relação ao outro, principalmente suas características distintas em relação a cor. A escolha de somente cinco tipos de terrenos partiu de uma tendência encontrada nos artigos utilizados como fonte de informações para o desenvolvimento do projeto.

A grama apresenta-se como um terreno predominantemente verde, com uma textura bastante presente. A areia é predominantemente amarela e apresenta uma baixa quantidade de textura. As pedras são mais variadas em relação a cor, porém continuam sendo na maior parte azul, quanto a textura também apresentam um certo grau, porém não tanto quanto a grama. O asfalto fica entre uma cor cinza e a cor preta, quase não apresenta textura assim como a areia. Por último está o pavimento com uma cor mais próxima do cinza. Possui uma textura parecida com a pedra.

No total, o banco gerado possui 383 imagens, sendo que o objetivo foi obter número de imagens bem próximo de cada tipo de terreno. Além disso, para cada tipo de terreno foram coletadas imagens variadas com intuito de aumentar a capacidade de generalização da rede. Por exemplo, as figuras 10 e 11 mostram dois tipos de imagem de grama, as figuras 12 e 13 representam a areia, as figuras 14 e 15 são de pedras e, finalmente, as figuras 16 e 17, respectivamente de pavimentação e asfalto.

Figura 10 – Grama Alta.

Figura 11 – Grama Baixa.

As imagens foram capturadas todas em dias limpos e ensolarados, num intervalo das 13h até as 16h, isso faz com que exista uma constância na questão de iluminação do ambiente, sombras do veículo de testes não interferem de forma incisiva no reconhecimento dos terrenos.

Figura 12 – Areia de praia.

Figura 13 – Areia de construção.

Figura 14 – Pedra Brita Grande.

Figura 15 – Pedra Brita Pequena.

Figura 16 – Pavimento Hexagonal.

Figura 17 – Asfalto Acinzentado.

3.3 Técnicas de Pré-processamento utilizadas

3.3.1 Técnicas de Cores

As classes propostas para o reconhecimento possuem características de cores bastante particulares, porém sabe-se que o espaço RGB é muito afetado por condições ambientais, sendo a principal o efeito da luminosidade da cena. Esse problema poderia ser contornado por meio da adicão de imagens com diferentes características de luminosidade no treinamento da rede neural, porém aumentaria muito o banco de dados a ser utilizado.

Sabendo-se dessa limitação causada pela luminosidade do ambiente, optou-se por utilizar além de parâmetros RGB, o espaço de cores nRGB, que divide em intensidade e luminosidade. Isso aumenta as chances de a rede conseguir acertar o terreno mesmo com pequenas variações de luz.

A escolha do espaço de cores nRGB foi devido a simplicidade da conversão para este espaço de cores. Outros espaços de cor foram aplicados com esta utilidade, como por exemplo o espaço de cores HSV, porém não houve uma melhora significativa para o banco de dados proposto, além de que a conversão para o espaço de cor HSV acaba sendo mais custoso computacionalmente que a conversão para nRGB

Os descritores para a retirada de informação relacionado a cor, são os *Momentos de Cor* ou *Momentos Estatísticos*, como o nome mesmo já sugere, são descritores estatísticos [12].

A média pode ser entendida como o valor médio da cor presente na imagem, e pode ser calculado pela seguinte equação:

$$E_i = \sum_{j=1}^{N} \frac{1}{N} \cdot p_{ij} \tag{3.1}$$

O parâmetro N diz respeito à quantidade de pixels da imagem e p_{ij} é o valor do pixel na i-ésima coluna e na j-ésima linha da matriz da imagem.

O desvio padrão amostral é calculado pela raiz quadrada da variância, e é uma medida de dispersão probabilística que indica a variabilidade das amostras, ou no caso imagens. Pode ser calculado como:

$$\sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^{N} \cdot (p_{ij} - E_i)^2} \tag{3.2}$$

O último momento estatístico é o *skewness*, que gera valores interessantes para a descrição da imagem. Isto porque baseado nos terrenos propostos, existe uma grande diferenciação em relação a este parâmetro. Neste contexto, superfícies mais brilhantes e

escuras tendem a serem mais positivas enquanto superfícies mais claras e foscas tendem a ser mais negativa:

$$s_i = \sqrt[3]{\frac{1}{N}\sum_{j=1}^{N} \cdot (p_{ij} - E_i)^3} \tag{3.3}$$

As imagens coloridas possuem na maior parte das vezes três matrizes de cores, uma para cada canal de cor. Para isso é interessante que se aplique os momentos de cores para cada uma das três matrizes.

3.3.2 Técnicas de Textura

Apesar das cores serem um bom indicativo para a distinção de cenários, por vezes dois terrenos possuem muito em comum em relação a cores, citando como exemplo a areia de praia de cor amarelada e a grama queimada pelo sol, conforme figuras 18 e 19. Ambos os terrenos são muito parecidos ao se analisar suas cores, porém são terrenos totalmente distintos quanto a sua forma. Este é o motivo pelo qual se fez necessário uma abordagem em relação à forma e não somente cor.

Figura 18 – Areia de praia.

Figura 19 – Grama seca.

Para tal utilizou-se o filtro de Gabor que é um filtro passa-faixa, e a sua resposta ao impulso é obtida pela multiplicação de uma função Gaussiana por uma oscilação senoidal. A escolha dos parâmetros $\lambda, \theta, \phi, \sigma, \gamma$, deve ser realizada conforme a necessidade e qualidade da saída do filtro. Como os filtros de Gabor buscam por características de textura, bordas neste caso, em uma frequência e determinada orientação, deve-se avaliar quais os parâmetros que adequem os filtros para obter a maior quantidade de informações para o processo. Os valores de X e Y são argumentos que especificam a posição do impulso de

luz no campo visual. Pode ser entendida pelas equações abaixo:

$$g_{\lambda,\theta,\phi,\sigma,\gamma}(x,y) = exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \cdot cos(2\pi\frac{x'}{\lambda} + \phi) \tag{3.4}$$

$$x' = x \cdot cos\theta + y \cdot sin\theta \tag{3.5}$$

$$y' = -x \cdot sin\theta + y \cdot cos\theta \tag{3.6}$$

O valor de σ é o desvio padrão do filtro gaussiano, que é o fator que determina o campo de recepção do filtro, γ é a proporção espacial, ϕ é o deslocamento de fase, θ é a orientação no qual o filtro funcionará e por fim λ é o comprimento do filtro. Ver Figura 25.

A escolha do filtro de Gabor provém da grande aplicação do filtro nos trabalhos acadêmicos, sempre obtendo bons resultados em relação a outros filtros.

Depois de se obter o filtro de Gabor é realizada sua convolução com cada imagem do banco de dados, e para cada imagem são retirados mais dois descritores, um é estatístico, a média e o segundo é a energia do sinal, mesmo parâmetro utilizado por *Haralick* [8].

O processo de escolha do conjunto de técnicas utilizados, pode ser entendido pelo fluxograma mostrado na Figura 20.

3.4 Arquitetura da Rede Neural

As redes neurais multicamadas pode possuir uma quantidade grande de neurônios por cada camada e uma quantidade grande de camadas, contudo existe um número ideal de neurônios e camadas para cada aplicação necessária da rede para que se obtenha a melhor generalização possível.

É importante ter em mente que o tamanho da rede será de suma importância para que a rede funcione conforme desejado. Normalmente a determinação do tamanho da rede é realizada de forma empírica, executando alguns testes com a mesma base de dados, alternando os parâmetros que ajustam o desempenho da rede, como a quantidade de camadas, quantidade de neurônios ou a velocidade de aprendizagem da rede.

As redes podem enfrentar duas situações diferentes após seus treinamentos e que não são agradáveis, são elas o *underfitting* e o *overfitting*. Ambas são situações a serem evitadas e dizem respeito aos parâmetros escolhidos no treinamento. O *underfitting* diz respeito à situação onde o treinamento da rede ficou muito abaixo de um valor ideal, onde a rede pode errar muito por não ter atingido uma situação de generalização. Já o *overfitting* é o contrário, a rede decorou o conjunto de treinamento, ou seja, não será possível fazer uma generalização com banco de dados diferente [4]. Desta forma, muitas vezes é preferível

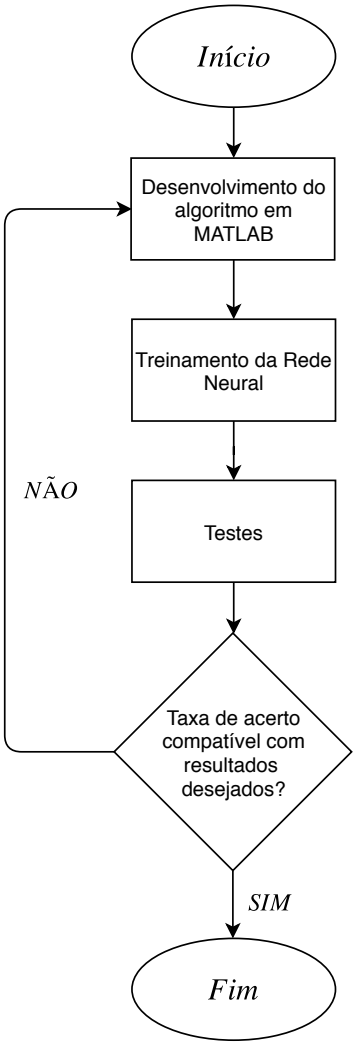

Figura 20 – Fluxograma da obtenção das técnicas de processamento

que a rede tenha um erro médio quadrático um pouco mais elevado, porém com uma generalização mais contundente.

Com base nestes aspectos, é necessário montar uma rede que possa se desvencilhar dessas situações indesejáveis. A escolha dos parâmetros da rede foi feita com base nos resultados obtidos com a alternância nos valores dos parâmetros, como o número de camadas e neurônios. O processo de treinamento pode ser entendido pelo fluxograma da figura 21.

O banco de dados é divido entre treinamento e validação, isso significa que parte das imagens do banco de dados é utilizado como informação de entrada da rede para seu treinamento, o que normalmente representa 75% das amostras. Os dados que serão utilizados como treinamento são obtidos de forma aleatória, afim de não gerar falsos positivos durante o treinamento.

Os outros 25% das amostras são utilizados na validação da rede, ou seja, tem por objetivo a minimização do *overfitting* da rede neural, neste ponto os pesos sinápticos não são ajustados mas sim servem como uma métrica.

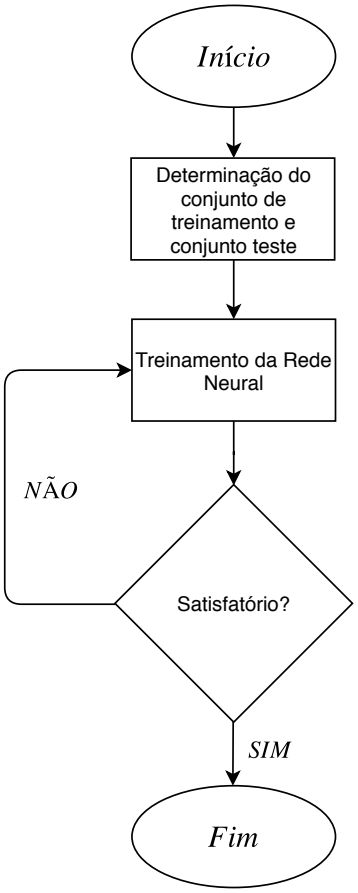

Figura 21 – Fluxograma da obtenção da Rede Neural

A rede foi exaustivamente treinada até chegar-se à arquitetura adequada. O ajuste da rede neural trata-se de um ajuste fino que deve ser feito baseado nos resultados obtidos com treinamentos anteriores. Na maior parte das vezes a escolha dos parâmetros ocorrerá de forma heurística, ajustando parâmetros como erro máximo permitido, número de épocas, taxa de aprendizagem dentre outros.

3.5 Código da Rede

A escolha do código para a implementação da rede ocorreu em duas etapas principais, primeiro escolhendo-se qual a linguagem de programação que seria utilizada e por último qual código atenderia a necessidade para eventuais alterações na rede.

A linguagem de programação escolhida foi a linguagem orientada a objetos C++, por facilitar a organização do código e sua compatibilidade com bibliotecas necessárias para o pré-processamento das imagens.

Após testes entre dois códigos abertos que permitem uma maior e mais fácil modificação, foi optado pelo projeto MLP - *Multilayer Perceptron* de autoria de David Nogueira, que além de ser escrito em C++ também possui uma maior documentação [13].

Para a validação da escolha, os parâmetros da rede no código em C++ foram escolhidos para ficarem bem próximos daqueles utilizados no MATLAB. A taxa de acerto foi a principal métrica de comparação entre os resultados do MATLAB com a da rede embarcada.

Para demonstrar como funcionam os códigos serão apresentados dois pseudo-códigos. Para o treinamento da rede neural, o pseudo-código parte do pressuposto que haja um banco de dados prontos para a retirada de informações. O processo é bastante parecido com o uso após o treinamento da rede neural, porém neste caso ele realiza o levantamento de todos os descritores de todas as imagens do banco, enquanto que no uso ele encontra os descritores para a imagem atual e já descobre qual a saída da rede, ou seja, já disponibilizando o terreno classificado.

```
1  inicio:
2
3  // \textit{While} para ler todas as imagens do banco e ao final, realizar o
       treinamento da rede
4
5  enquanto (tamanhodoBanco - imagensdoBancoLidas != 0){
6
7    // Tratamento inicial, abre imagem do banco de dados e redimensiona para
       256x256
8    imagem = abreImagemBanco();
9    imagem = redimensionaImagem(imagem);
10
11   // Retirar Descritores relativos a cor
12
13   descritorCor[i] = retiraMedia(imagem);
14   descritorCor[i+1] = retiraDesvioPadrao(imagem);
15   descritorCor[i+2] = retiraSkewness(imagem);
16
17   imagemNRGB = converteNRGB(imagem);
18   descritorCor[i+3] = retiraMedia(imagemNRGB);
19
20   // Retirar Descritores relativos a textura
21
22   filtro = criaFiltroGabor(parametrosGabor);
23   bancoFiltro = criaBancoFiltro(filtro);
24   imagemCinza = converteCinza(imagem);
25
26   gaborFinal = convoluiImagem(bancoFiltro, imagemCinza);
27
28   descritorTextura[i] = retiraMedia(gaborFinal);
29   descritorTextura[i+1] = retiraEnergia(gaborFinal);
30
31   // Unir os descritores
32
```

```
33   conjuntoRNA = descritorCor e descritorTextura;
34
35 }
36
37 // Treinar na Rede Neural Artificial
38
39 pesossinapticosRNA = treinarRedeNeural(conjuntoRNA);
40
41 fim
```

O código para o sistema segue o mesmo descrito para o treinamento da rede, com uma pequena variação vista no pseudo-código a seguir:

```
1 inicio:
2
3 // Tratamento inicial, captura e redimensionamento para 256x256
4
5 imagem = capturaImagem();
6
7 imagem = redimensionaImagem(imagem);
8
9 // Retirar Descritores relativos a cor
10
11 descritorCor[i] = retiraMedia(imagem);
12 descritorCor[i+1] = retiraDesvioPadrao(imagem);
13 descritorCor[i+2] = retiraSkewness(imagem);
14
15 imagemNRGB = converteNRGB(imagem);
16 descritorCor[i+3] = retiraMedia(imagemNRGB);
17
18 // Retirar Descritores relativos a textura
19
20 filtro = criaFiltroGabor(parametrosGabor);
21 bancoFiltro = criaBancoFiltro(filtro);
22 imagemCinza = converteCinza(imagem);
23
24 gaborFinal = convoluiImagem(bancoFiltro, imagemCinza);
25
26 descritorTextura[i] = retiraMedia(gaborFinal);
27 descritorTextura[i+1] = retiraEnergia(gaborFinal);
28
29 // Unir os descritores
30
31 conjuntoRNA = descritorCor e descritorTextura;
32
33 // Aplicar na Rede Neural Artificial
34
35 classificacaoRNA = redeNeural(conjuntoRNA);
```

```
36
37 fim
```

3.6 Protótipo

O protótipo para o teste da rede se trata de uma plataforma padrão móvel, com 4 rodas e chassi de material acrílico. Para o sistema de classificação dos terrenos foi optado por utilizar-se o *Raspberry Pi 3* modelo B, por tratar-se de uma plataforma compacta e dotada de um sistema operacional.

Para uma maior mobilidade o protótipo opera com baterias de lítio-íon, e a *Raspberry* por conta de seu consumo conta com uma *power bank* que fornece os 5V e 2A necessários para o seu funcionamento completo.

A câmera utilizada para a coleta das imagens trata-se de uma *Logitech C270*, que captura imagens com 720p e um campo de visão (FoV) de 60°. Para que o robô tenha maior mobilidade, utilizaram-se baterias para a alimentação da parte de locomoção do veículo e para alimentação da *Raspberry Pi*. A câmera foi fixada em duas barras roscadas, para que se obtivesse um grau de liberdade quanto ao posicionamento da mesma, para compensar seu baixo campo de visão.

4 Resultados

Neste capítulo serão apresentados os resultados obtidos durante o desenvolvimento do projeto. O capítulo será dividido em, seção 4.1 é descrita como a técnica de pré-processamento é realizada; na seção 4.2 é mostrado a evolução da configuração dos descritores e da rede desde seu início até a versão final; na seção 4.3 mostra a arquitetura final da rede neural e, por fim, na seção 4.4 é mostrado o teste realizado no protótipo com o sistema finalizado.

4.1 Procedimento de Pré-processamento

As imagens são capturadas pela câmera em uma resolução de 720p, que corresponde a uma imagem de 1280x720 pixels. O primeiro tratamento realizado é o redimensionamento da imagem para o tamanho de 256x256 pixels utilizando interpolação, tornando a imagem menor o que facilita e aumenta a velocidade de processamento da imagem.

Figura 22 – Imagem 1280x720.

Figura 23 – Imagem 256x256.

Com a imagem já redimensionada, começa-se o processamento para a retirada dos descritores de cor. O primeiro passo é a obtenção dos descritores estatísticos das imagens no espaço **RGB**. Como o formato da imagem capturada já está no espaço de cor **RGB**, basta aplicar as equações 3.1, 3.2 e 3.3. Estes descritores são armazenados em um vetor, totalizando nove descritores, três para cada camada de cor do **RGB**.

Como o **RGB** apresenta problemas com a incidência de luz e objetivando elevar um pouco a robustez do tratamento, é realizada uma conversão de espaço de cores **nRGB** ou cromaticidade, resultando na figura 24. Os valores absolutos da cromaticidade são utilizados para compor o vetor de descritores, já que possuem uma informação bastante única por si só e não levam em consideração a luminância.

Em relação ao tratamento em relação a cores, obtém-se doze descritores. O próximo passo é a retirada dos descritores de textura baseados em Gabor.

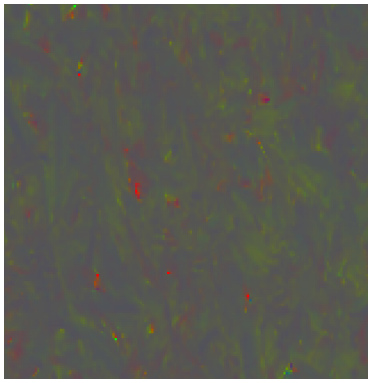

Figura 24 – Imagem no espaço de cores nRGB.

Os descritores de textura são obtidos através da convolução da imagem com um filtro de Gabor. Com o objetivo de aumentar a precisão dos descritores de textura, ao invés de utilizar-se somente um filtro com uma direção somente, foi construído um banco de filtros, mais precisamente quatro filtros com quatro direções diferentes conforme figura 25.

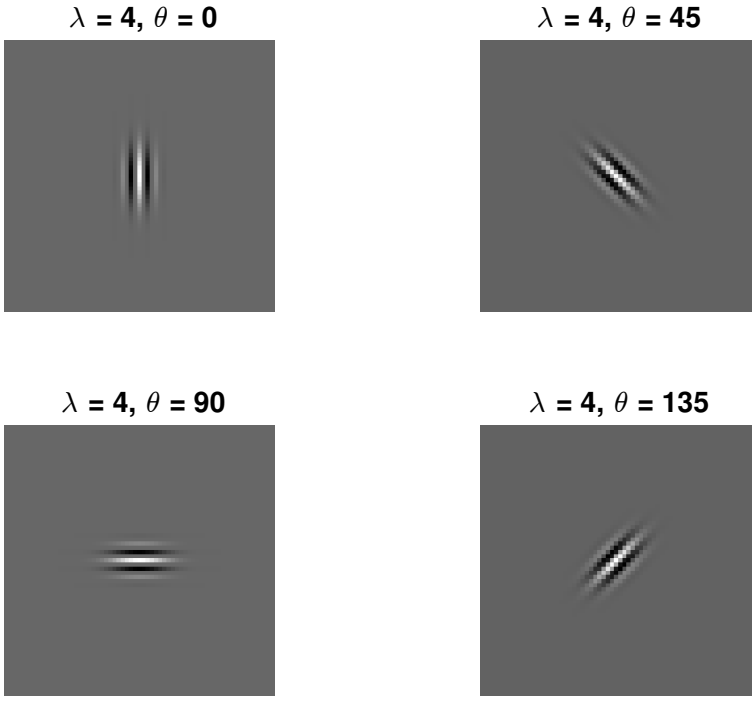

Figura 25 – Banco de filtros de Gabor.

Primeiramente, a imagem **RGB** é transformada em uma imagem em níveis de cinza, para que se possa fazer a convolução da imagem com o filtro em somente uma dimensão.

Após realizada a conversão da imagem, realiza-se a convolução com cada um dos filtros do banco, resultando assim em quatro imagens de saída para este tratamento.

Cada uma das saídas será diferente, pois cada um dos filtros é configurada de forma diferente para extrair a maior quantidade possível de informações em cada direção. Os descritores utilizados também são estatísticos, então é retirada a média e a energia do sinal, gerando dois descritores por filtro e oito no total por cada imagem capturada.

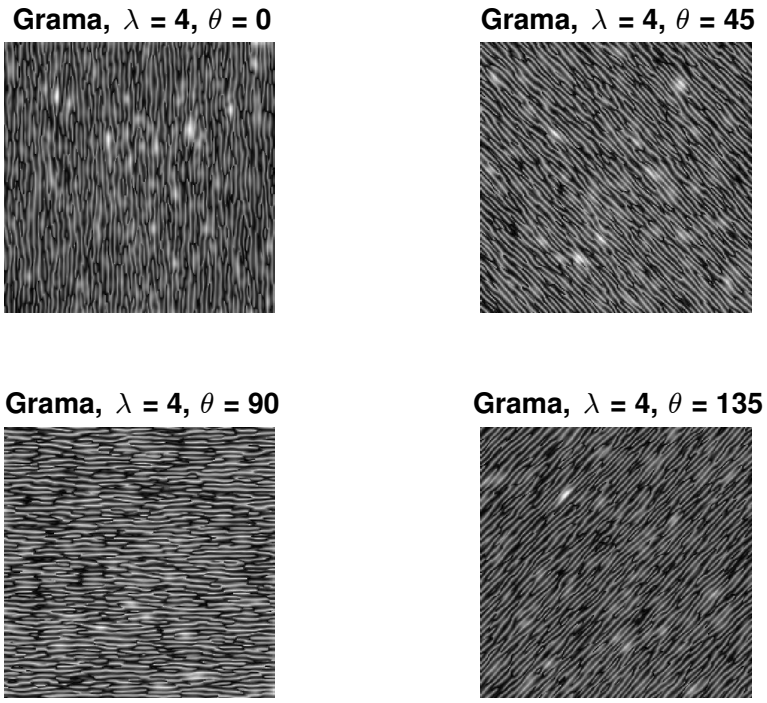

Figura 26 – Imagem após convolução com filtro de Gabor.

Ao final do pré-processamento, um vetor de vinte descritores é formado e está pronto para ser inserido na rede neural para o devido treinamento. A Tabela 1, apresenta os valores comuns resultantes das técnicas de processamento explicitadas, ou seja, dos descritores estatísticos para cada um dos diferentes solos em que foi proposto o reconhecimento. É possível observar a diferença entre os descritores de cada terreno, que possibilita com que a rede neural possa ser treinada com um conjunto de dados de representatividade relevante.

Tabela 1 – Tabela de descritores para cada terreno.

Descritor	**Grama**	**Areia**	**Pedra**	**Pavimento**	**Asfalto**
Média R	0.5760	0.5834	0.6076	0.5732	0.6014
Média G	0.5840	0.5444	0.6016	0.5690	0.5979
Média B	0.5758	0.4271	0.5874	0.5763	0.6084
Desvio Padrão R	0.1614	0.1305	0.1818	0.1851	0.0782
Desvio Padrão G	0.1664	0.1296	0.1827	0.1843	0.0790
Desvio Padrão B	0.1638	0.1158	0.1774	0.1860	0.0766
Skewness R	-0.1173	-0.1082	0.0345	0.1024	0.0861
Skewness G	-0.1283	-0.1055	0.0523	0.0945	0.0871
Skewness B	-0.1181	-0.0869	0.0683	0.1024	0.0839
Cromaticidade R	0.3318	0.3752	0.3382	0.3336	0.3327
Cromaticidade G	0.3364	0.3501	0.3270	0.3353	0.3308
Cromaticidade B	0.3317	0.2747	0.3349	0.3311	0.3366
Média Gabor 0 °	0.6999	0.7362	0.7065	0.7123	0.6476
Média Gabor 45 °	0.7277	0.6708	0.7277	0.7191	0.7457
Média Gabor 90 °	0.6997	0.6881	0.6958	0.6986	0.6575
Média Gabor 135 °	0.7316	0.7223	0.7076	0.7286	0.6973
Energia Gabor 0 °	0.5018	0.5541	0.5119	0.5185	0.4285
Energia Gabor 45 °	0.5434	0.4602	0.5433	0.5301	0.5679
Energia Gabor 90 °	0.5024	0.4844	0.4965	0.4985	0.4433
Energia Gabor 135 °	0.5490	0.5336	0.5135	0.5410	0.4964

4.2 Definição da arquitetura da Rede Neural

A configuração final da RNA atingida, passou por algumas modificações com o decorrer do projeto. A sua arquitetura foi alterada até atingir um ponto onde pode-se obter uma boa relação *custo/benefício*.

A métrica bastante utilizada para avaliações de redes neurais é a chamada *Confusion Matrix*. No total são quatro *confusion matrix* para cada rede neural treinada, a (*training confusion matrix*) que representa a métrica em relação ao treinamento, (*validation confusion matrix*) que diz respeito a métrica para a validação, a (*test confusion matrix*) que representa a fase de testes da rede e o resultado total final (*overall confusion matrix*), onde são somados os resultados de cada uma das matrizes anteriores. As duas matrizes mais importantes são as de treinamento e teste, pois indicam a qualidade final da rede.

Inicialmente utilizaram-se somente 143 imagens. A configuração da rede era de dezesseis neurônios e somente uma camada oculta de processamento. Com o objetivo de

aumentar a taxa de acertos, uma função custo diferente foi abordada, conhecida por *Cross-Entropy*. Entretanto mesmo aumentando o número de neurônios ou camadas alterando a taxa de aprendizagem da rede, as taxas não eram nada satisfatórias.

A *Confusion Matrix*, figura 27, gerada pelo MATLAB deixa isso claro.

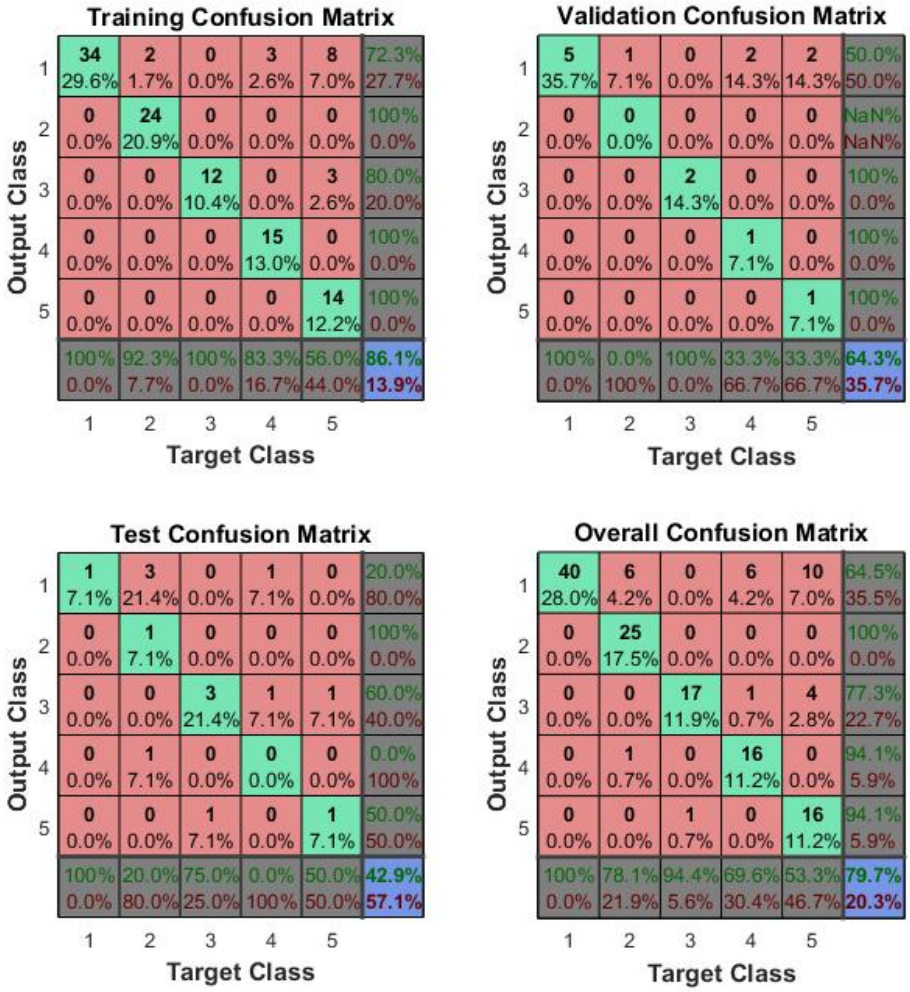

Figura 27 – Primeira RNA com banco de dados reduzido.

A *Confusion Matrix* da figura 27, apresenta uma taxa geral de 79.7%, o que não representa um número por si só ruim, porém a taxa de acerto do conjunto de teste está em 42.9% o que é um valor muito inferior ao esperado. Este valor significa dizer que a rede está errando mais do que a metade dos terrenos que deveria distinguir.

Por conta disto a primeira estratégia para a resolução do problema foi a ampliação do banco de dados. As imagens do banco de dados foram melhoradas e aumentadas para 383 imagens. Todos os testes foram repetidos, inclusive voltando-se a utilizar a função custo **EQM**.

Os resultados apresentados foram muito melhores, porém o treinamento estava sendo realizado levando em conta somente descritores de cor. O grande problema é quando

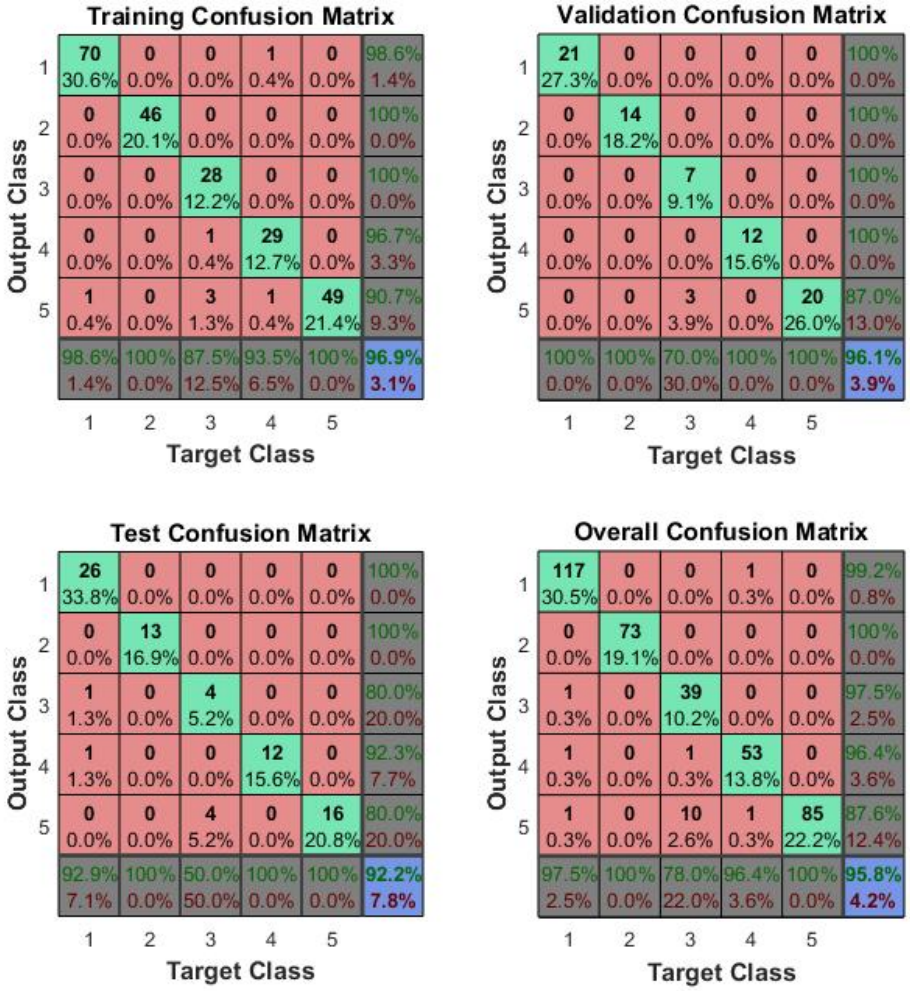

Figura 28 – Segunda RNA somente com descritores de cor.

ocorre variações de cor no terreno inserido e a rede não consegue distinguir com muita clareza. Pequenas variações nos aspectos de cor podem confundir a rede. Uma solução seria inserir imagens no banco de dados com o terrenos com variações, porém desta forma para cada situação se faria necessário um aumento no banco de dados.

Chegou-se então aos descritores de textura, utilizando-se de descritores estatísticos e filtros de Gabor. Os descritores de textura não representaram um aumento para a taxa de acertos em relação aos testes, conforme pode ser visto na figura 29, pois somente utilizando descritores de cor já seria possível o reconhecimento, porém permitiu o reconhecimento de imagens que contivessem uma certa diferenciação de cor em relação as imagens do banco de dados. O principal impacto no uso dos descritores de textura foi permitir um acréscimo no poder de reconhecimento da rede.

Realizadas as duas abordagens distintas para a retirada de descritores, uma em relação a cor e a outra em relação a textura, chegou-se ao momento de unir os descritores e obter o resultado que pode ser visto na figura 30. A taxa de acertos teve um pequena queda

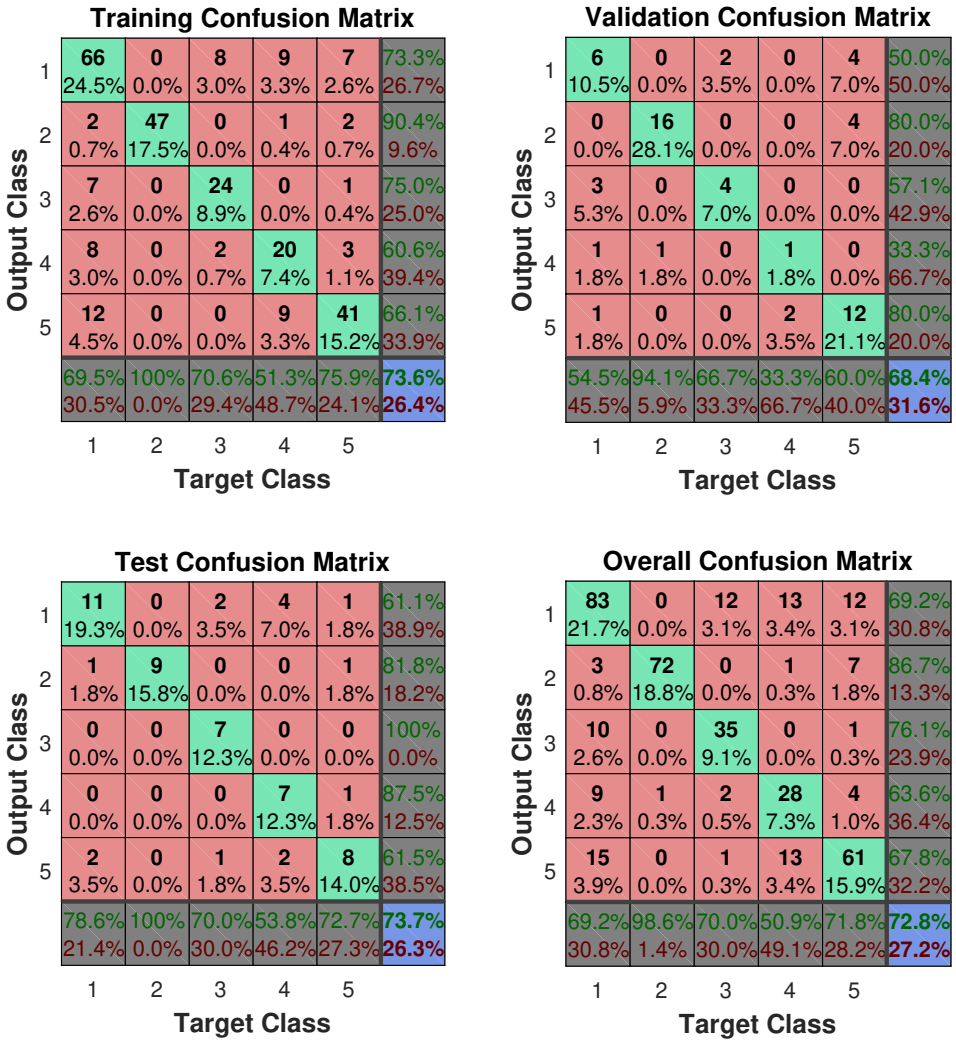

Figura 29 – RNA somente com descritores de textura.

em relação ao uso somente dos descritores de cor, porém ainda assim a taxa de acerto continuou bastante otimista com um valor de 91.6 %.

4.3 Rede Neural final

A arquitetura final da rede neural, foi determinada por meio de testes onde procurou-se a melhor combinação para a maior taxa de acertos. Determinou-se que a rede possui somente uma camada oculta e dezesseis neurônios nesta camada.

Neste caso foi constatado que um valor menor que dezesseis neurônios fazia com que a rede não permitisse a classificação de forma adequada. Valores maiores de camadas e neurônios não apresentaram uma melhora significativa para a taxa de acertos, então optou-se por manter uma taxa boa e uma rede menor, poupando processamento em treinamento e utilização.

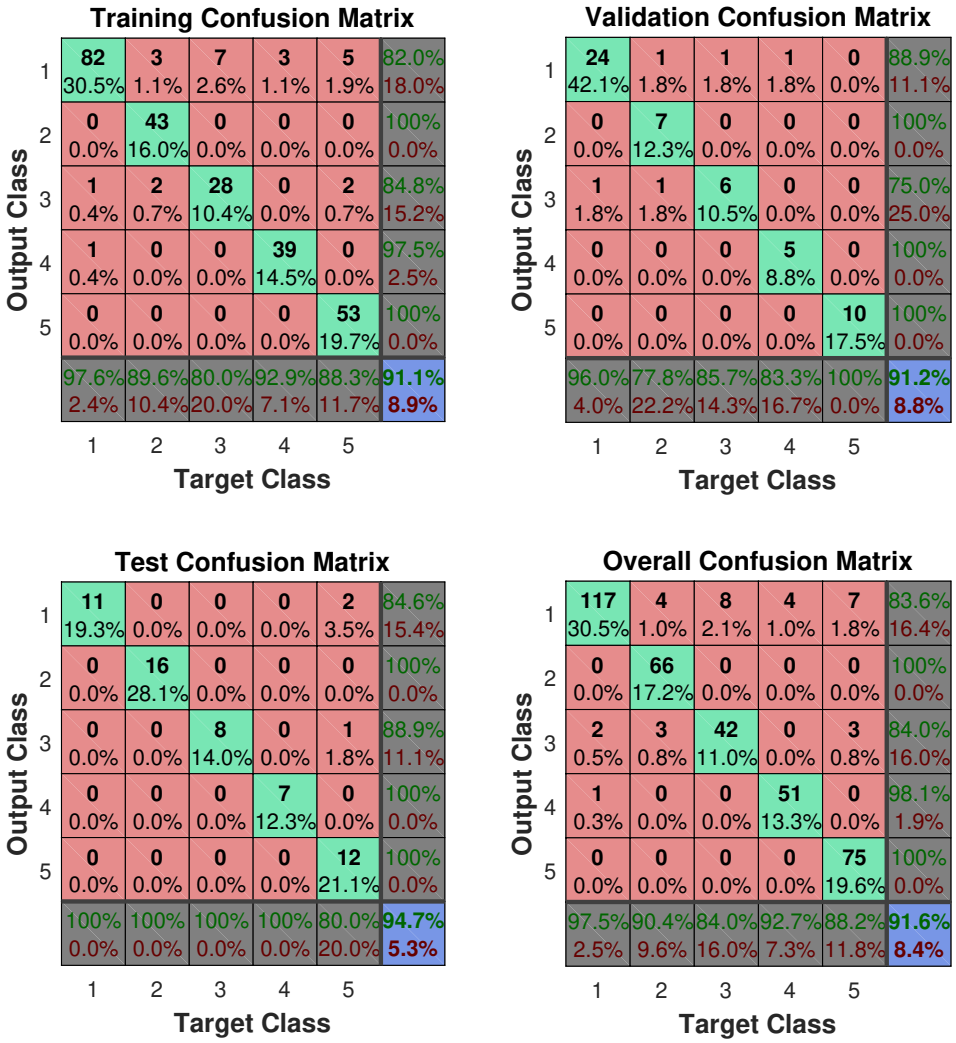

Figura 30 – RNA com descritores de cor e textura.

4.4 Protótipo

4.4.1 Versão final do protótipo

Uma imagem da versão final do protótipo pode ser visto em 31. A posição da câmera na base demonstra a necessidade da obtenção do banco de dados no mesmo formato.

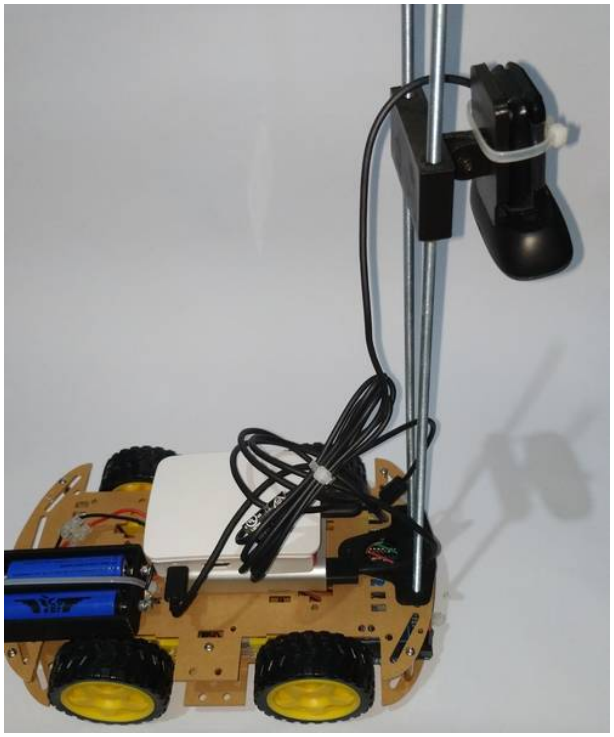

Figura 31 – Protótipo final do veículo.

4.4.2 Treinamento e teste

Ao obter-se o resultado final da seção anterior para a rede e as técnicas de processamento, foi iniciado a transição do código escrito em MATLAB para o código em C++. O código em sua essência continuou o mesmo, porém utilizando bibliotecas do *OpenCV 3.2.0* para o tratamento das imagens e o *MLP* para a rede neural. O sistema operacional utilizado na *Raspberry* foi o *Raspbian*. O treinamento da rede neural no protótipo, obteve uma taxa de acertos de 91.18%, o que é um número interessante, pois se encontra muito próximo do obtido através do MATLAB.

Para o teste, o protótipo foi inserido em cada um dos terrenos e retirados amostras de imagem de cada um deles. O cálculo dos descritores acontece logo após a captura da imagem, formando o vetor de vinte posições e sendo inserido na rede neural que já foi treinada. Conforme os testes foi possível obter a tabela 2.

Tabela 2 – Tabela de acertos para os testes do protótipo.

Terreno	**Grama**	**Areia**	**Pedra**	**Asfalto**	**Pavimento**	*Total*
Grama	10 (20.0%)	0	1(2.0%)	1(2.0%)	2(4.0%)	71.43%
Areia	0	10(20.0%)	1(2.0%)	0	0	90.91%
Pedra	0	0	8(16.0%)	0	0	100.0%
Asfalto	0	0	0	9(18.0%)	0	100.0%
Pavimento	0	0	0	0	8(16.0%)	100.0%
Total	100.0%	100.0%	80.0%	90.0%	80.0%	90.0%

Os testes foram realizados com imagens obtidas do protótipo em um total de dez imagens para cada terreno proposto. Pela limitação de se usar uma câmera *Rolling Shutter*, isto é, a imagem não realiza a todo o processamento de captura de uma só vez, o veículo precisa parar a cada imagem que ele retira. Pode-se observar uma certa robustez da rede, pois algumas imagens obtidas nos testes se diferenciam das imagens inseridas no banco de imagens.

Conforme tabela 2, a rede foi capaz de reconhecer as dez imagens de grama, as dez imagens de areia, oito imagens de pedra, nove imagens de asfalto e oito imagens do pavimento. A grama, areia e asfalto foram os três terrenos com melhor aproveitamento no reconhecimento, pois possuem características bem distinguíveis em questão de cor e textura, sendo a grama predominantemente verde, areia amarela e o asfalto com uma coloração bastante escura e quase sem textura. A pedra confundiu-se com a areia, pois o terreno que foi realizado o teste possuía uma quantidade de areia que pode ter provocado a alteração dos parâmetros de cor que foi predominante na escolha em relação a textura.

Outra dificuldade no reconhecimento, foi entre o pavimento e a pedra que pela composição do pavimento, pedra e cimento, confunde-se bastante o reconhecimento. Caso a imagem do pavimento seja obtida de uma certa distância, este problema não será tão visível graças ao formato hexagonal do pavimento.

5 Análises

Neste capítulo será realizada a análise do trabalho; na seção 5.1 é apresentada uma comparação da complexidade das técnicas e rede neurais utilizadas em trabalhos relacionados; na seção 5.2 é comentado sobre o aumento do banco de dados e na última seção, 5.3 é apresentado um parecer sobre o protótipo.

5.1 Comparação com outras técnicas utilizadas

No trabalho de Hata et al, os autores utilizaram de um sensor laser 2D, que com certa inclinação ao solo, obtém a distância em relação aos pontos capturados. Utilizando esses dados de distância como informações para alimentar uma rede perceptron multicamadas, obteve bons resultados com uma taxa de acerto de 99.36% [14].

Segundo artigo de Chang et al, utilizou-se de dados provenientes de um HSV quantizado. Para a extração de descritores é aplicada uma transformada de *Wavelet*, dividindo a imagem em quatro subimagens no domínio da frequência, sendo uma sequência de filtros passa-baixa e passa-alta. Para cada uma das imagens é calculada matriz de co-ocorrência de motif, uma alternativa para as matrizes de nível de cinza, e após isso, aplicam os descritores de *Haralick*. Seus resultados variam entre 93% até 100% no reconhecimento [15].

Para o trabalho de Wong et al, seu objetivo é o reconhecimento de diferentes tipos de plantas, muito semelhante ao reconhecimento de terrenos, utilizou técnicas de processamento para dar uma maior importância à cor verde por ser a predominante no seu banco de dados. Para a extração de descritores usou as matrizes de nível de cinza junto com os descritores de *Haralick*, onde obteu um resultado de 76.3% no reconhecimento das plantas [16].

Xu Ma et al, empreenderam um UAV para a captura de imagens de terrenos a uma certa altura. Utiliza de técnicas de cores, momentos de cor e para reconhecer a textura utiliza uma *wavelets* de Gabor. Suas amostras foram de 32x32 pixels com um banco de dados de 1000 imagens. Utilizaram uma rede neural artificial para comparação com o método de classificação proposto. Para a rede neural artificial obtiveram um acerto de 91.18% e no sistema proposto 96.11% [2].

Tabela 3 – Tabela de comparação entre trabalhos.

	Manerichi	**Hata**	**Chang**	**Wong**	**Ma**
Taxa de acerto	90.0%	99.36%	93% a 100%	76.3%	91.18%

A comparação dos métodos acima aconteceu por meio de diferentes bancos de dados.

5.2 Tamanho do banco de dados

Um ponto que deve receber bastante atenção ao iniciar a construção de uma RNA, é qual o tipo de dado que será introduzido nela. Para isso cria-se um banco de dados que contém informações que serão utilizadas no processo. Duas características são bastante importantes, a questão da qualidade dos dados, se os dados possuem características particulares de cada classe a serem determinadas, e a quantidade de dados, quantos dados serão utilizados com o objetivo principal de uma boa generalização da rede.

A quantidade neste trabalho apresenta um papel bastante importante, pois o objetivo é que se reconheçam não somente uma espécie de grama ou somente um tipo de areia, mas sim que se possa distinguir que se trata de grama ou areia. Para isso uma das soluções é construir um banco de dados com muitas amostras dos diferentes terrenos, pois assim a rede poderá calcular seus pesos sinápticos para atender a maior gama possível de terrenos. Em via de regra, quanto maior o banco de dados, mais informações estarão disponíveis e maior a chance de acontecer um bom reconhecimento de classes.

A qualidade dos dados/descritores também não pode ser ignorada, pois ter um banco de dados muito grande sem que se consiga retirar características próprias de cada terreno, fará com que a rede continue sem uma capacidade de generalização. Neste caso deve-se atentar para os descritores, prestando a devida atenção até que se encontrem os melhores para a devida aplicação.

5.3 Protótipo

A utilização da *Raspberry Pi* como uma plataforma embarcada atendeu as necessidades do projeto. Nenhum problema com demora no processamento pode ser detectado além de possuir um baixo consumo de energia, mesmo com uma câmera conectada e a rede *wireless* conectada todo o tempo. A base móvel poderia apresentar uma maior robustez para percorrer com maior facilidade os terrenos, além de poder ser mais alta para uma compensação do FoV da câmera.

A câmera poderia apresentar algumas características para aumentar a qualidade da captura, um exemplo seria o uso de uma *global shutter*, que permitiria a captura da imagem com o veículo em movimento, pois no caso desta câmera utilizada, que é *rolling shutter*, a imagem ficaria "borrada"caso capturada com o veículo se movimentando, impossibilitanto assim a aplicação das técnicas de processamento. Outro aspecto da câmera é seu FoV, que é relativamente baixo, fazendo com que o seu campo de visão seja bem reduzido.

No geral, o protótipo atendeu de forma satisfatória para o trabalho, podendo ser contornado aspectos que poderiam se apresentar como problemas.

6 Conclusões

O objetivo desde trabalho foi o de reconhecer diferentes tipos de terreno por intermédio de redes neurais artificiais (Perceptron Multicamadas), utilizando como obtenção das informações técnicas de visão computacional, mais especificamente as técnicas de pré processamento. A rede foi construída utilizando-se o MATLAB como ferramenta de testes, por conta da quantidade de métricas disponíveis, e posteriormente embarcada em C++ no *Raspberry Pi* em um protótipo montado sobre uma base móvel, permitindo a captura das amostras de terreno.

Em relação as técnicas para a obtenção das informações para a rede, foram utilizados de descritores estatísticos que apesar de apresentarem baixa complexidade em relação à muitos outros abordados em artigos, mostrou-se uma ferramenta bastante útil na obtenção de características particulares das imagens. A abordagem de se obter informações relativas a cor e textura é algo bastante comum, porém também se apresentam úteis já que é possível descrever ambientes simples através deles. A questão da iluminação não foi tão preocupante, pois o banco e testes foram realizados em ambientes controlados, porém com a inserção de um novo espaço de cores o impacto foi reduzido conforme constatado nos testes do protótipo.

A rede foi capaz de obter um índice bom de reconhecimento dos cinco diferentes tipos de terrenos, podendo quase ser comparado com outros trabalhos na área de reconhecimento. Em artigos utilizados como base para o desenvolvimento do projeto pode ser encontrados taxas de acerto entre 76% até mesmo 99%, sendo assim os dados do teste permaneceu neste intervalo de taxas. Os métodos mantiveram-se com um nível inferior de complexidade em relação à alguns trabalhos, principalmente pelo uso de uma rede neural PMC.

Como perspectivas futuras pode-se apresentar um banco de dados maior, que abranja mais tipos de terrenos e com uma maior variedade em relação à cada terreno. Isso aumentará bastante a capacidade de reconhecimento da rede. O outro aspecto importante seria aumentar a confiabilidade da rede, como aumentar os tipos de sensores para coletar mais informações sobre o terreno, conforme o trabalho de Kragh at al [10], que utiliza de um sensor LIDAR para aumentar a qualidade de seus dados, tornando a rede além de robusta também mais confiável.

O tratamento de interferência em relação a luz na imagem também poderia aumentar a robustez da rede, algo pouco abordado neste trabalho. Desenvolver técnicas para que os descritores não sejam alterados pela quantidade de luz tornariam o reconhecimento mais confiável.

Referências Bibliográficas

1 HALLOWAY, W. *TECHNOLOGY GUIDE: TERRAIN RESPONSE*. 2015. <https://www.landrover.co.uk/explore-land-rover/one-life/technology/technology-guide-terrain-response.html>. [Online; accessed 27-January-2019].

2 MA, X.; HAO, S.; CHENG, Y. Terrain classification of aerial image based on low-rank recovery and sparse representation. In: *2017 20th International Conference on Information Fusion (Fusion)*. [S.l.: s.n.], 2017. p. 1–6.

3 FONG, T.; NOURBAKHSH, I.; DAUTENHAHN, K. A survey of socially interactive robots. *Robotics and autonomous systems*, Elsevier, v. 42, n. 3-4, p. 143–166, 2003.

4 SILVA, I. N. d.; SPATTI, D. H.; FLAUZINO, R. A. *Redes neurais artificiais: para engenharia e ciências aplicadas*. [S.l.]: ARTLIBER, 2010. ISBN 9788588098534.

5 HAYKIN, S. *Neural Networks. A Comprehensive Foundation*. 2. ed. [S.l.]: Prentice Hall, 1998. ISBN 9780132733502,0132733501.

6 CORKE, P. *Robotics, Vision and Control*. [S.l.]: Springer International Publishing, 2017. ISBN 978-3-319-54413-7.

7 SZELISKI, R. *Computer Vision*. [S.l.]: Springer International Publishing, 2011. ISBN 978-1-84882-935-0.

8 HARALICK, R. M.; SHANMUGAM, K.; DINSTEIN, I. Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-3, n. 6, p. 610–621, Nov 1973. ISSN 0018-9472.

9 OJALA, T.; PIETIKAINEN, M.; MAENPAA, T. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 24, n. 7, p. 971–987, July 2002. ISSN 0162-8828.

10 KRAGH, M.; JøRGENSEN, R.; PEDERSEN, H. Object detection and terrain classification in agricultural fields using 3d lidar data. In: . [S.l.: s.n.], 2015.

11 RABELO, R. T. *Arquitetura de hardware dedicada de uma rede neural perceptron para reconhecimento de terreno aplicado a robótica móvel*. Brasília: [s.n.], 2014.

12 SHIH, J.; CHEN, L. Colour image retrieval based on primitives of colour moments. *IEEE Proceedings - Vision, Image and Signal Processing*, v. 149, n. 6, p. 370–376, Dec 2002. ISSN 1350-245X.

13 NOGUEIRA, D. *MLP*. 2016. <https://github.com/davidalbertonogueira/MLP>. [Online; accessed 20-October-2018].

14 HATA, A. Y. et al. Terrain mapping and classification using neural networks. In: *Proceedings of the 2009 International Conference on Hybrid Information Technology*. New York, NY, USA: ACM, 2009. (ICHIT '09), p. 438–442. ISBN 978-1-60558-662-5. Disponível em: <http://doi.acm.org/10.1145/1644993.1645074>.

15 CHANG, J.-D. et al. Hsv-based color texture image classification using wavelet transform and motif patterns. *Journal of Computers*, v. 20, 01 2010.

16 WONG, W. et al. Co-occurrence matrix with neural network classifier for weed species classification: A comparison between direct application of co-occurrence matrix (glcm) and haralick features as inputs. *International Journal of Enhanced Research in Science and Engineering*, v. 2, p. 1–6, 01 2013.