Processamento Digital de Sinais
Identificação De Sinais de Transito Utilizando Redes Neurais

Thiago Machado da Cunha
Centro de Energias Alternativas e Renováveis
Departamento de Engenharia Elétrica - UFPB
João Pessoa, Brasil
thiago.cunha@cear.ufpb.br

Resumo – Redes neurais são técnicas computacionais baseadas em estruturas neurais de organismos inteligentes. O aprendizado de máquina é vastamente utilizado em diversas aplicações. Neste projeto, o objetivo principal é a utilização da técnica de redes neurais para identificação de placas de trânsito. Uma grande tendência que aos poucos torna-se realidade é o surgimento de carros autônomos, que pode fazer uso das técnicas desenvolvidas neste projeto como solução para implementação progressiva de carros autônomos, que a princípio compartilhará as estradas com carros guiados por humanos. Uma outra possível aplicação seria no auxílio na locomoção de deficientes visuais.

I. INTRODUÇÃO

O aprendizado de máquina consiste na utilização de algoritmos para processar dados de tal forma que a máquina tome ações baseadas em dados pré-processados. Isso se assemelha ao comportamento humano que adquire conhecimento conforme vive novas experiências.

O presente relatório refere-se ao projeto final da disciplina Processamento Digital de Sinais sobre Redes Neurais, mais especificamente o emprego desta técnica para identificação de placas de sinalização de trânsito, visando a aplicação em carros autônomos, ou para auxilio de deficientes visuais. A princípio, a rede neural é capaz de identificar placas de "PARE", "SIGA" e placas de informação de direção. Para atingir tais objetivos foi utilizado o MATLAB. O trabalho apresentará uma breve revisão teórica acerca de processamento digital de imagens e redes neurais, seguido do procedimento para desenvolvimento do projeto e por fim, uma breve conclusão.

II. REVISÃO TEÓRICA

A. Processamento Digital de Imagens

A imagem digitalmente processada pode é usada em remoções de ruído, como também são utilizadas no reconhecimento de através de características obtidas no processamento digital.

A imagem digital consiste é um grupo de pixels que assumem três possíveis valores, em diferentes níveis de intensidade, dependendo do nível de bits do conversor ADC. Para este experimento, a intensidade dos pixels assume valores máximos de 255, sendo assim 8 bits o nível de codificação do ADC.

B. Componentes de cor RGB

O método mais utilizado para codificar cores é o RGB (*Red Green Blue – Vermelho Verde Azul)* em computadores, monitores LCD. A partir de suas três cores base pode-se obter qualquer cor misturando-as em diferentes níveis de intensidade. Sendo valor equivalente a cor preta, em qualquer das componentes, e seu valor máximo, para este experimente equivale a 255, representa a cor branca.

A Figura 1 representa o espaço de cores RGB.

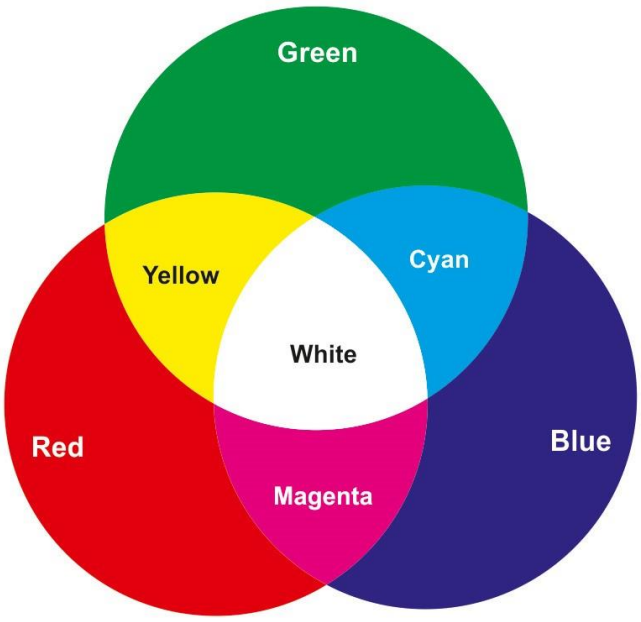

Figura 1: Componentes RGB

C. Histograma

O histograma, no processamento digital de imagem, é uma forma de representação das componentes de frequência

das cores das imagens, sendo assim uma ferramenta ideal para processamento de imagens facilitando a eliminação de ruídos, ou na filtragem de componentes desejadas.

D. Redes Neurais

É uma das técnicas mais utilizadas no aprendizado de máquinas. Como o termo "Neurais" já sugere, é baseado no sistema neural humano que se baseia na forma como aprendemos para desenvolver uma rede.

Redes neurais consiste em camadas de entradas e saídas, com uma camada intermediaria que processa a entrada a se obter a saída desejada. A figura 2 consiste num desenho demonstrativo de uma rede neural simples.

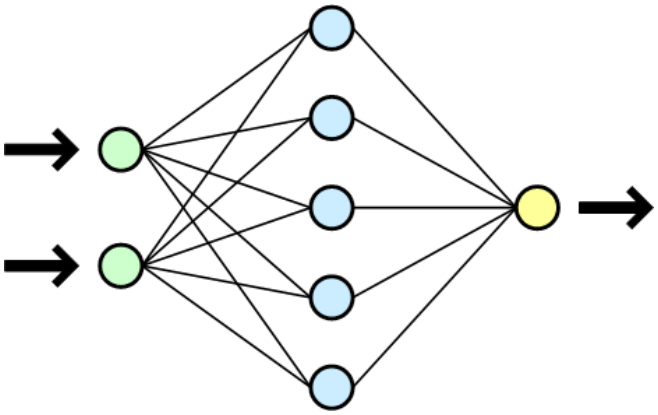

Figura 2: Rede neural

O processamento em cada neurônio é feito através de funções de ativação. A escolha da função ativação dirá o como deverão ser os dados de entrada.

Para realização deste projeto foi utilizada a função *sigmoide*. A equação e gráfico da equação sigmoide são apresentados abaixo.

$$\varphi(x)=1 / (1+e^{-x})$$

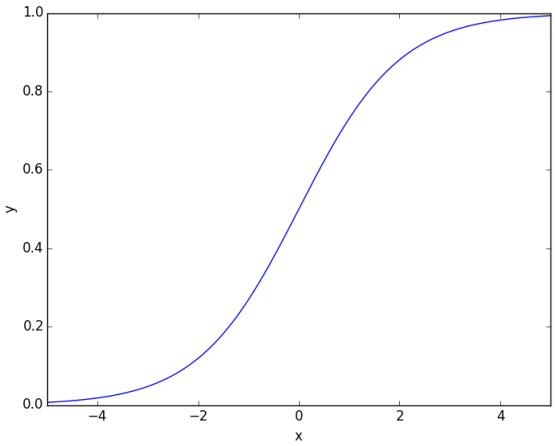

Figura 3: Função Sigmóide

III. Desenvolvimento

O desenvolvimento deste projeto é dividido em três estágio, onde o primeiro consiste em captar a imagem a ser processada, e extrair apenas a região de interesse. Ou seja, cortar a imagem para que haja apenas a informação da placa.

O segundo estágio consiste em extrair as informações dos componentes de cores da imagem cortada e exportar para o dado para o pré-processamento da rede neural.

E no terceiro estágio é montado o algoritmo para treinamento da rede.

A. Primeiro Estágio

Tendo em vista a utilização em veículos autônomos, observou-se que as placas de sinalização encontram-se na maior parte no canto superior direito da imagem, conforme apresentado na Figura 4.

Figura 4: Placa de sinalização.

A princípio, após ser carregada, a imagem é cortada em 4 partes, e aproveitada ao equivalente ao primeiro quadrante, fazendo uma analogia com o sistema cartesiano ortogal.

Figura 5: Imagem cortada

```
Im = imread(I);

Im = imcrop(Im,[size(Im(:,:,1),2)/2 0
size(Im(:,:,1),2)/2 size(Im(:,:,1),1)/2]);
```

Após cortada, foi extraído as componentes de cores vermelho da imagem cortada, e em seguida binarizada. É interessante ressaltar que esta parte do código é destinada a identificar a localização do objeto, portanto é estabelecido um limiar, ou seja, um nível para que acima deste nível, a imagem seja branca e abaixo, preta. O resultado desta etapa do código é apresentado na Figura 7, após o trecho do código referente a operação.

Figura 6: Componente vermelho da imagem.

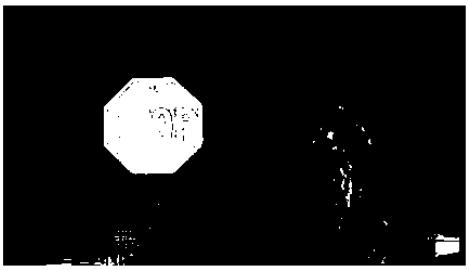

Figura 7: Imagem após binarização

```
levelr = 0.45;

i1=im2bw(rmat,levelr);

I_stop = imfill(i1,'holes'); %to stop signs

Icomp = imcomplement(i1); %to green signs
I_green = imfill(Icomp,'holes');
```

Nota-se que após binarizada, a região equivalente a placa apresenta muitos pontos pretos. Para reduzir estes ruídos, a região é preenchida. Mas o resultado ainda não é satisfatório conforma apresentado na Figura 8.

Como solução foi utilizado uma função *buit-in* do MATLAB que preenche áreas com círculos de raio determinado na própria função. O resultado é apresentado na Figura 9. Bem satisfatório.

Figura 8: Preenchimento parcial dos buracos indesejados

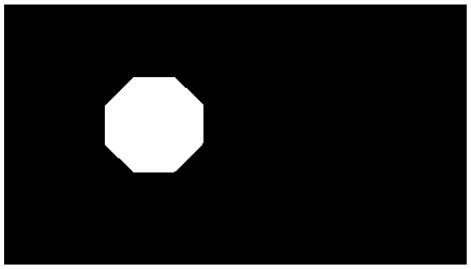

Figura 9: Preenchimento definitivo

```
se_stop = strel('disk', 30);
se_green = strel('disk', 25);

Io_stop = imopen(I_stop,se_stop);
Io_green = imopen(I_green,se_green);
```

Após o objeto ser destacado, é mapeado uma região quadrada em torno do objeto para ser cortada. O resultado e o respectivo trecho do código estão a seguir.

Figura 10: Resultado final após corte

Para consolidar o teste do código foi feito um teste com uma placa de cor diferente.

Figura 11: Segunda imagem após corte do primeiro quadrante

Após extração de cores, diferentemente da placa de PARE, em nenhum componente de cor a imagem destacou-se.

Figura 12: Componente vermelho da segunda imagem

Para solucionar este problema, foi utilizado uma função *built-in* que inverte a imagem binarizada. O resultado é apresentado na Figura 13.

Figura 13: Inversão dos valores binarizados.

O resto processo é similar a placa PARE obtendo o resultado final da Figura 14.

Figura 14: Resultado final

B. Segundo Estágio

O segundo estágio consiste simplesmente em exportar os dados de cores das imagens cortadas. A informação exportada que será utilizada na rede neural será o valor máximo de pixels que contém o máximo valor de intensidade nos componente vermelho e verde.

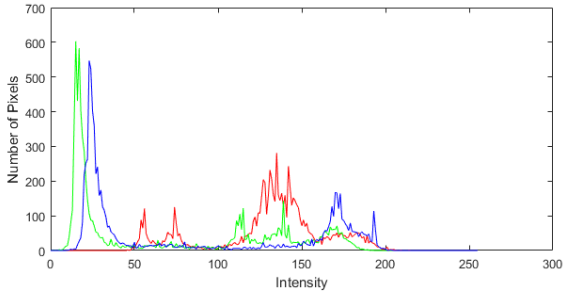

Figura 15: Histograma do sinal PARE

Na Figura 15, nota-se que na região de intensidade entre 100 e 200, a componente vermelha é que tem maior valor de pico, caracterizando uma imagem de predominância da cor vermelho.

Foi mapeado a partir do décimo valor de intensidade para evitar valores de pico iniciais devido a ruídos apresentado em algumas imagens.

O seguinte diagrama foi levantado com base nos valores levantados.

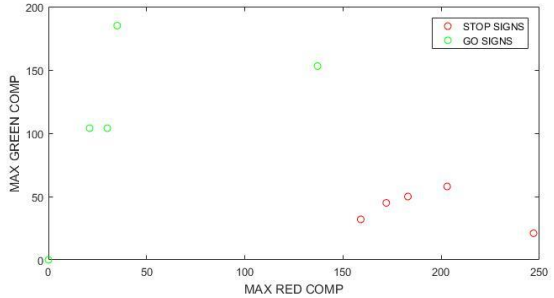

Figura 16: Diagrama a partir dos dados retornados da função

```
function [int_r,int_g] =
alert_sgn_comp(image)

sgn = crop_object(image);
%sgn = imread(image);

sgn_red = sgn(:,:,1);
sgn_green = sgn(:,:,2);
sgn_blue = sgn(:,:,3);

[h_sgn_red , x] = imhist(sgn_red);
h_sgn_green = imhist(sgn_green);
h_sgn_blue = imhist(sgn_blue);

[num_p_r,int_r]=max(h_sgn_red(10:end-20));
[num_p_g,int_g]=max(h_sgn_green(10:end-20));
[num_p__b,int_b]=max(h_sgn_blue(10:end-20));

int_r = int_r+10;
int_g = int_g+10;
int_b = int_b+10;
figure(1)
plot(x,h_sgn_red,'red',x,h_sgn_green,'green',
x,h_sgn_blue,'blue')

end
```

C. Terceiro Estágio

Neste estágio será realizado o treinamento da rede neural. Como dado, foi extraído de cada imagem o maior valor de intensidade da componente vermelha e da componente verde.

Primeiramente é atribuído pesos aleatórios, que conforme a rede é treinada, serão atualizados.

A rede funcionará da seguinte forma. Ao entrar com os dados da intensidade verde e vermelha, a rede retornará 0 ou 1, sendo 0 correspondente a placa “SIGA” e 1 correspondente a placa “PARE”.

Figura 17: Placa SIGA utilizada no projeto

A função responsável por retornar zero ou um é a função sigmoide, que quando tende a -∞ é zero e quando tende ∞ é 1.

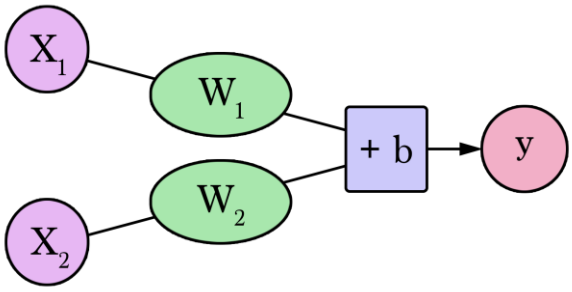

Figura 18: Rede utilizado no projeto

Após obter o primeiro dado de saída, ele é compara com o valor esperado, realizado a diferença entre eles, e elevando ao quadrado. Esse valor é chamado de cost (custo em português). É uma medida de erro, do valor obtido para o valor que deveríamos obter.

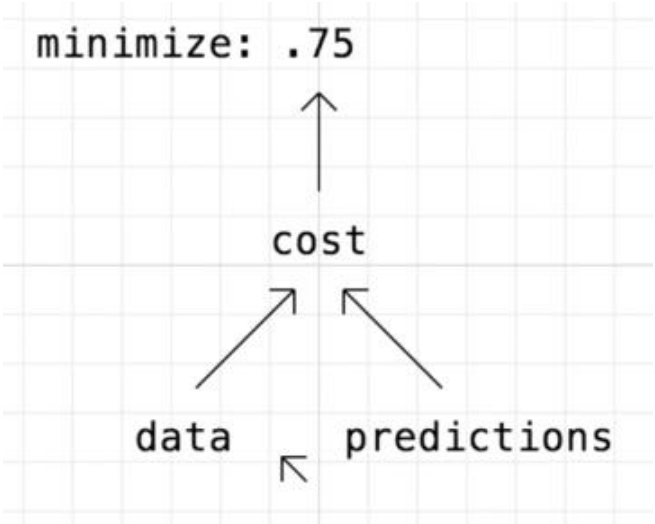

Figura 19: Neurônio responsável pelo cálculo do erro

Após obtido o cost, é preciso saber se devemos incrementar ou reduzir nossos parâmetros para que seja atingido o valor correto. Para obter tal informação é utilizada a derivada da função cost. Se obtiver um valor negativo, incrementamos, do contrário, é redudizo os parâmetros. Para que essa mudança não seja brusca, é multiplicado um *learning_rate*, este falor multiplica o valor calculado, da derivada do função *cost* suavizando as iterações.

Os três parametros são atualizados conforme suas derivadas variam, até que atinja o valor esperado.

A técnica utilizada neste projeto foi regressão linear, conforme apresentado no Código abaixo.

```
%CADA PONTO CORRESPONDE A [TIPO MAX_RED
MAX_GREEN]

data = [0 30 104;0 21 104 ;0 137 153 ;1
         247      21;...
    1 183 50 ;1 159 32; 1 172 245; 1
         203      58 ;0    35       185];
%data = data./255;

%   O   TIPO DA PLACA
%  / \  w1, w2, b
% O   O MAX_R, MAX_G

w1 = rand;
w2 = rand;
b = rand;

% treinando

learning_rate = 0.3;

for i = 1:10000
    rnd_idx = round(randi([1
length(data)],1));
    test_p = data(rnd_idx,:)/10;
    x = w1*test_p(2) + w2*test_p(3) + b;
    predic = sigmf(x,[1,0])
    data(rnd_idx,:)
    target = test_p(1)*10;
    cost = (predic - target)^2;

    dcost_dpredic = 2*(predic - target);
    dpredic_dx = sigmf(x,[1,0])*(1-
sigmf(x,[1,0]));
    dx_dw1 = test_p(2);
    dx_dw2 = test_p(3);
    dx_db = 1;

    dcost_dw1 = dcost_dpredic * dpredic_dx *
dx_dw1;
    dcost_dw2 = dcost_dpredic * dpredic_dx *
dx_dw2;
    dcost_db = dcost_dpredic * dpredic_dx *
dx_db;

    w1 = w1 - learning_rate*dcost_dw1;
    w2 = w2 - learning_rate*dcost_dw2;
    b = b - learning_rate*dcost_db;

end
```

IV. CONCLUSÃO

O estágio primário do circuito, desempenhou o papel proposta na eliminação de componentes desnecessários para analise proposta pelo projeto.

O segundo estágio também teve seu funcionamento comprovado levantando os dados necessários para a rede neural identificar a placa lida.

Bem como no terceiro estágio, após treinada, a rede neural identificou sem falhas o tipo de placa apresentada.

Esse conjunto de acertos nos permitiu concluir que o projeto foi corretamente executado e o valores obtidos referentes ao comportamento da rede neural satisfazem as condições esperadas.