Comparação de diferentes funções de ativação no reconhecimento ótico de caracteres alfabéticos

Gustavo da Silva Oliveira, Kenji Nose-Filho
Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas
Universidade Federal do ABC (UFABC)
Email: s.oliveira@aluno.ufabc.edu.br, kenji.nose@ufabc.edu.br

***Resumo*—Este artigo de iniciação científica compara o uso de diferentes funções de ativação (degrau, sigmoide e base radial) de uma rede neural artificial com apenas uma camada para o problema de reconhecimento ótico de caracteres do alfabeto latino. Além de diferentes funções de ativação foram utilizadas *features* com diferentes dimensões. Os resultados obtidos foram comparados com a ferramenta de OCR Tesseract. Apesar de preliminares, os resultados obtidos mostram que com este classificador bastante simples e utilizando uma função de ativação adequada é possível obter resultados bastante satisfatórios.**

***Palavras-chave*—funções de ativação, processamento digital de imagens, reconhecimento ótico de caracteres, redes neurais artificiais**

I. Introdução

Ao longo dos últimos anos a Visão Computacional tem se tornado de grande interesse não apenas para a comunidade científica mas também para a Indústria em geral. A visão computacional tem como objetivo principal a extração de informações presentes em imagens e/ou dados multimídia e, para isso, técnicas e ferramentas baseadas na Inteligência Artificial e no Aprendizado de Máquina podem ser amplamente utilizadas [1], [2].

Uma das aplicações é a possibilidade de localizar placas de sinalização em imagens de vídeo e/ou foto e realizar a identificação do que está sendo sinalizado ou escrito [3]. Tal ferramenta seria de grande importância para pessoas com algum tipo de deficiência visual, seja ela parcial ou completa.

Existem diversas ferramentas na literatura capazes de realizar o reconhecimento ótico de caracteres (OCR, do inglês *Optical Character Recognition*) [2]–[7]. Incluindo algumas de código aberto como o Tesseract[1], o Latin OCR[2], o GNU Ocrad [3] e algumas ferramentas comerciais como, por exemplo, a API Rekognition da Amazon[4].

Em geral, o reconhecimento/classificação é feito com base nos caracteres previamente extraídos da imagem, sendo necessário a identificação das caixas de texto, a separação dos caracteres e o cálculo das chamadas *features* [1], [7]. Recentemente, as ferramentas baseadas nas chamadas rede neurais convolucionais tem recebido um grande destaque na área. Devido a sua grande capacidade de, a partir de uma imagem, realizar a detecção automática de caixas de texto e fazer o reconhecimento das palavras [3], [8], [9].

Neste artigo de iniciação científica são comparadas o uso de diferentes funções de ativação de uma rede neural artificial com apenas uma camada para o problema de reconhecimento ótico de caracteres do alfabeto latino. Os caracteres são reconhecidos a partir de imagens de placas de sinalização com as caixas de texto previamente identificadas. O classificador foi treinado para conseguir identificar as 52 letras do alfabeto latino, sendo 26 letras maiúsculas e 26 letras minúsculas sem levar em consideração os acentos.

A partir da imagem com o texto previamente extraído é realizada a sua conversão para escala de cinza e posteriormente o Método de Otsu é aplicado para a sua binarização. Na imagem binarizada os caracteres são extraídos e, para cada carácter, são calculadas as *features*. As *features* servem de entrada para o classificador que irá realizar a classificação das 52 letras do alfabeto latino. Para a tarefa de classificação foi utilizado uma rede neural artificial com apenas uma camada, onde foram testadas diferentes funções de ativação, como a função degrau, a função sigmoide e a função de base radial e também *features* com diferentes dimensões. Os resultados obtidos foram comparados com a ferramenta Tesseract.

O cálculo das *features* foi feito com base no histograma da imagem de forma a buscar, da forma mais intuitiva possível, uma representação simples para cada carácter. Já a rede neural artificial com apenas uma camada foi escolhida devido a sua simplicidade de programação e treinamento. As diferentes funções de ativação foram utilizadas com o objetivo de verificar e comparar a capacidade de generalização de cada uma delas no problema de OCR. Outro ponto que vale a pena destacar neste trabalho foi a utilização/construção de um banco de dados próprio ao invés de utilizar um banco de dados público. O objetivo de construir o seu próprio banco de dados foi feito com o intuito para que o aluno pudesse compreender melhor as dificuldades e limitações que a sua ferramenta poderia ter com diferentes imagens, contribuindo ainda mais para o seu aprendizado. Por fim, vale ressaltar que todas as funções utilizadas foram programadas pelo próprio aluno, em Octave, sem o auxílio de *toolboxes* específicas.

Este artigo está organizado em 4 seções, sendo esta primeira uma breve introdução ao assunto. Nas seções posteriores serão

Os autores gostariam de agradecer ao PIBIC/CNPq pelo apoio financeiro.

[1]Disponível em:https://github.com/UB-Mannheim/tesseract/wiki

[2]Disponível em: https://latinocr.org/

[3]Disponível em:https://www.gnu.org/software/ocrad/

[4]Disponível em: https://aws.amazon.com/pt/rekognition/?ft=n

ISSN 2447-8326. V.1

tratados a Metodologia, os Resultados e algumas Conclusões. A seção Metodologia discute como o trabalho foi realizado e está dividida em 5 subseções: Limiarização, Segmentação, Cálculo das *features*, Classificação e Treinamento. A seção Resultados trata dos resultados obtidos com as diferentes *features* e funções de ativação. Por fim a seção Conclusões finaliza o trabalho e traz algumas possíveis perspectivas de trabalhos futuros.

II. Metodologia

Nesta seção são explicados os processos de limiarização, segmentação, cálculo das *features*, classificação e treinamento que foram utilizados. Na etapa de limiarização a imagem colorida é transformada em uma imagem binária onde o valor "1" deverá corresponder às letras e o valor "0" ao fundo da imagem. Após realizar este processo a segmentação é aplicada, separando cada uma das letras da imagem, para que possam ser calculadas as *features*, que serão utilizadas como entradas para o classificador. Um treinamento com letras de diferentes fontes e fotografias reais de placas de sinalização foi realizado para que o perceptron pudesse realizar a classificação. Vale ressaltar que todos os códigos foram programados na linguagem de programação Octave, sem a utilização de *toolboxes* específicas de Inteligência Artificial e Aprendizado de Máquina.

A. Limiarização

As imagens utilizadas na fase de treinamento e teste foram imagens de placas cujo texto está disposto em apenas uma linha na horizontal e contêm apenas caracteres maiúsculos e minúsculos do alfabeto latino sem acentos, como podem ser vistas na Figura 1.

Além disso estas imagens são originalmente coloridas e precisam ser transformadas em imagens binárias (preto e branco), com o objetivo de separar as letras do fundo da imagem. Primeiramente, as imagens coloridas são transformadas em escala de cinza, como ilustrado pela Figura 2 e a limizarização é feita com base no método de Otsu, conforme [10].

No método de Otsu a ideia principal é a de encontrar um limiar ótimo que minimize a variância denominada intra-classe [11], dada por:

$$\sigma_w^2(t) = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t), \tag{1}$$

onde w_0 e w_1 são as probabilidades das classes separadas pelo limiar t, σ_0^2 e σ_1^2 são as variâncias dessas duas classes.

Esse procedimento é equivalente a maximizarmos a variância extra-classe, que pode ser calculada através da expressão abaixo:

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) \\ &= w_0(\mu_0 - \mu_T)^2 + w_1(\mu_1 - \mu_T)^2 \\ &= w_0(t)w_1(t)[\mu_0(t) - \mu_1(t)]^2.\end{aligned} \tag{2}$$

Este limiar é obtido através de uma busca exaustiva, calculando, para cada possível valor de limiar, a variância extra-classe. Por fim é escolhido o limiar que maximiza este valor.

Após a aplicação desse método tem-se como produto uma imagem binária, mas não há garantias de que o fundo da imagem assumirá o valor 0 (preto), e as letras assumirão o valor 1 (branco). Portanto é executada uma função que soma os pixels brancos e os pixels pretos da imagem e, caso o número de pixels brancos seja maior que o de pixels pretos, os pixels que tinham valor 0 (preto), assumem o valor 1 (branco) e vice-versa. O funcionamento da função é baseado no pressuposto de que o número de pixels que correspondem ao fundo da imagem é maior do que o número de pixels correspondentes às letras. A figura 3 é um exemplo de imagem após o processo de limiarização e definição do fundo.

Fig. 1. Imagem de uma placa de sinalização com a caixa de texto previamente identificada, cujo texto está disposto em apenas uma linha na horizontal.

Fig. 2. Imagem após utilizar a função *rgb2gray* do Octave.

Fig. 3. Imagem binária obtida após a aplicação do método Otsu e da identificação do fundo da imagem.

B. Segmentação

Após a limiarização da imagem é necessário separar cada uma das letras para a sua posterior classificação. Para isso foi utilizado um algoritmo que soma todos os elementos de cada coluna da imagem e guarda essa soma em um vetor. Após ter feito esse primeiro passo o vetor é varrido por um laço de repetição. Quando se encontra um elemento neste vetor com o valor diferente de 0, a posição desse elemento é guardada, o laço continua até que um elemento com o valor igual a 0 seja achado, essa posição também é salva. Em um glifo é colocada a parte da imagem que corresponde a posição inicial (primeiro elemento de valor diferente de 0) e final (um elemento antes de

ser achado um elemento com valor igual a 0). Isso corresponde a um corte vertical da imagem, em que as letras são separadas. Esse processo continua até que todo vetor seja varrido.

Após ser finalizado esse laço de repetição, um processo análogo é feito para recortar a borda superior e inferior da imagem. Este processo se baseia em dois laços de repetição, um varre o vetor começando pelos elementos de cima (referente a primeira linha da imagem) e outro pelos elementos de baixo. Um exemplo de separação das letras em glifos é ilustrado pela Figura 4.

Após a separação das letras em glifos distintos, todos esses glifos são redimensionados de modo que tenham o mesmo número de linhas, N_{lin}, e sua largura N_{col} mantenha as proporções originais da letra. Desta forma, cada glifo redimensionado irá possuir um valor fixo para N_{lin}, mas com N_{col} de modo a preservar a taxa de proporção original de cada letra. Neste trabalho, são utilizados 3 tamanhos diferentes para N_{lin}: 26, 52 e 128. Cada tamanho diferente de N_{lin} é utilizado para uma *feature* com dimensão diferente.

Fig. 4. Exemplo de separação da letra "g" da placa referente a Figura 3.

C. Cálculo das features

Para que o algoritmo possa realizar um bom reconhecimento, é preciso escolher *features* que sejam capazes de representar cada uma dos 52 letras que desejamos classificar [12]. Essas *features* são as entradas do classificador (perceptron) [2]. Na literatura, são descritas uma série de *features* que poderiam ser utilizadas para este problema [12] e a escolhida neste trabalho se baseia no histograma de projeção vertical de cada caractere.

O primeiro elemento da *feature* é dado por:

$$\hat{x}_1 = \frac{N_{col}}{N_{lin}}, \tag{3}$$

e as demais *features* são dadas por:

$$\hat{x}_{i+1} = \sum_{j=1}^{N_{col}} \frac{I(i,j)}{N_{col}}, \tag{4}$$

onde $I(i,j)$ representa cada glifo redimensionado e i varia de 1 até N_{lin}.

Após o cálculo das *features* foi necessário subtrair a média, de modo que o algoritmo fosse capaz de identificar tanto letras mais finas como letras mais grossas como, por exemplo, em Negrito e sem Negrito. Vale ressaltar que um processo de esqueletização da imagem poderia ter sido realizado, no entanto, optou-se por esta solução. Desta forma as *features* são recalculadas como sendo:

$$x_i = \hat{x}_i - \sum_{i=1}^{N_{lin}} \frac{\hat{x}_i}{N_{lin+1}}, \tag{5}$$

A Figura 5 ilustra as *Features* de dois caracteres referentes a letra "Q" maiúscula antes da subtração pela média. Já a Figura 6 ilustra as *Features* de dois caracteres referentes a letra "Q" maiúscula subtraídas pela média.

Neste projeto, foram considerados três dimensões ($N = N_{lin} + 1$) distintas para as *features*, com N igual a 27, 53 e 129.

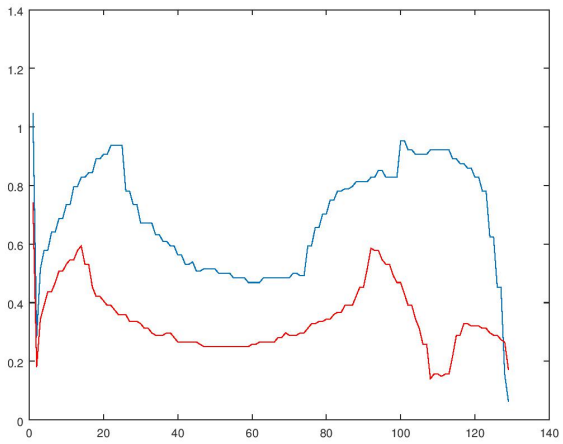


Fig. 5. *Features* de dois caracteres referentes a letra "Q" maiúscula, antes da subtração pela média.

D. Classificação

Para a tarefa de classificação foi utilizado uma rede neural composta de apenas uma camada com 52 perceptrons, com $N = N_{lin} + 1$ entradas [13]. O objetivo deste classificador é reconhecer 26 letras maiúsculas e 26 letras minúsculas do alfabeto latino, sem considerar acentos.

A saída de cada perceptron está relacionada com as letras em ordem alfabética. Por exemplo, o primeiro perceptron está relacionado a letra "A", já o 26 está relacionado com a letra "Z", o 27 está relacionado com a letra "a" e o 52 está relacionado com a letra "z".

A Figura 7 ilustra um percepetron com N sinais de entrada, dados por x_1, x_2,...,x_N, e apenas um sinal de saída y. O sinais de entrada são multiplicados pelos pesos w_1,w_2,...,w_N

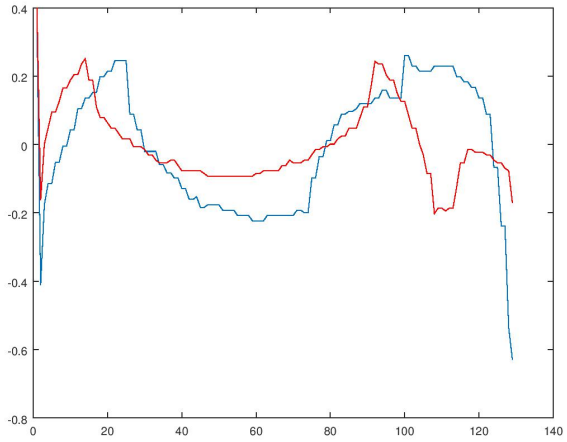


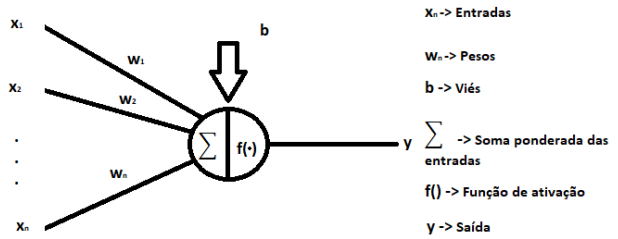
Fig. 6. *Features* de dois caracteres referentes a letra "Q" maiúscula, após a subtração pela média.

e este sinal resultante é então somado a um termo w_0 também conhecido como *bias*. A saída y é dada por:

$$y = f\left(\sum_{i=1}^{N} w_i x_i + w_0\right), \tag{6}$$

onde $f()$ é a chamada função de ativação.

Fig. 7. Modelo matemático de um perceptron.



Neste projeto, foram utilizadas três funções de ativação diferentes, a função degrau, a função sigmoide e a função de base radial. Para a função degrau temos saídas de zeros e uns, onde "0" indica a não correspondência do glifo de entrada com aquela letra e "1" indica a sua correspondência. Nas funções sigmoide e de base radial os valores de saída possuem valores reais entre "0 e "1", onde o perceptron com maior valor de saída é a correspondência mais provável para aquela *feature*.

1) Função degrau: Primeiramente, foi utilizada a função degrau como função de ativação, que, para valores de $z < 0$ é igual a $f(z) = 0$, e para valores de $z \geq 0$ é igual a $f(z) = 1$, onde $z = \sum_{i=1}^{N} w_i x_i + w_0$. A aprendizagem do perceptron é feita pela adaptação de seus pesos w_i e *bias* b. Esta adaptação é feita baseada na minimização do erro, de acordo com a seguinte regra de adaptação [14]:

$$w_i^{k+1} = w_i^k + \mu \sum_{i=1}^{L} e(l) x_i(l), \tag{7}$$

$$b^{k+1} = b^k + \mu \sum_{i=1}^{L} r(l), \tag{8}$$

onde

$$r(l) = d(l) - y(l), \tag{9}$$

é o sinal de erro dado pela diferença entre o sinal na saída da rede e o sinal desejado, k corresponde à k-ésima iteração, μ é o passo de adaptação, responsável pela convergência da rede e L é o número de amostras utilizadas no treinamento. O objetivo é obter, ao final de K iterações, valores de w_i e b que minimizem o sinal de erro.

2) Função sigmoide: Após o teste com a função degrau, a função sigmoide foi escolhida como função de ativação. Esta se difere da função degrau por ser uma função cuja saída é contínua. Esta função é dada pela seguinte equação:

$$f(z) = \frac{1}{1 + e^{-z}}, \tag{10}$$

Assim como na função degrau, a aprendizagem do perceptron é feita pela adaptação de seus pesos w_i e *bias* b. Uma das formas de se realizar esta adaptação é através do método dos mínimos quadrados (MMQ) [13]. Este método visa minimizar o erro quadrático dado pela saída $y(l)$ e o sinal desejada $d(l)$:

$$\sum_{l=1}^{L} r^2(l), \tag{11}$$

onde $r(l) = d(l) - y(l)$. Derivando a função custo com relação aos pesos e bias é possível então obter a seguinte regra de adaptação:

$$w_i^{k+1} = w_i^k + \mu \sum_{l=1}^{L} r(l)(1 - y(l))y(l)x_i(l), \tag{12}$$

$$b^{k+1} = b^k + \mu \sum_{l=1}^{L} r(l)(1 - y(l))y(l). \tag{13}$$

3) Função de base radial: A última função de ativação que foi testada foi a função de base radial. Assim como na função sigmoide os valores da saída são contínuos. Esta função é dada pela seguinte equação:

$$f(z) = e^{-z^2}, \tag{14}$$

A aprendizagem do perceptron para esse função também é baseada no método dos mínimos quadrados e é feita pela adaptação dos pesos e *bias* através da seguinte regra[5]:

$$w_i^{k+1} = w_i^k + \mu \sum_{l=2}^{L} (-2r(l)z(l)y(l))x_i(l), \tag{15}$$

[5]Vale ressaltar que a regra de adaptação utilizada para o treinamento do perceptron com a função sigmoide e a função de base radial foi baseada no método do gradiente descendente e que o gradiente foi calculado pelo próprio aluno.

$$b^{k+1} = b^k + \mu \sum_{l=2}^{L} (-2r(l)z(l)y(l)). \quad (16)$$

E. Treinamento

Neste trabalho, tanto para o treinamento como para os testes foi construído um banco de dados próprio. Apesar de existirem uma série de bancos de dados disponíveis de forma aberta e gratuita como os apresentados em [3], a construção de um banco de dados próprio é bastante interessante do ponto de vista didático/pedagógico para o aluno, sendo possível perceber melhor as características do problema a ser resolvido.

Para o treinamento foram utilizados os caracteres pertencentes às seguintes fontes: Verdana, Verdana em negrito, Arial e Arial Black (retirando a letra "i" maiúscula e a letra "l" minúscula das fontes Arial e Arial Black), Helvetica (retirando o "i" maiúsculo), Helvetica em negrito (retirando o "i" maiúsculo e o "l" minúsculo), Interstate e os caracteres de 20 placas de sinalização (com o total de 203 letras), ilustrados pela Figura 8. Ao todo foram utilizadas 561 letras no treinamento. Em geral, buscou-se construir um banco de dados com letras sem serifa, visto que este tipo de letra é comumente encontrada em placas de sinalização.

Foram realizados treinamentos utilizando *features* com diferentes dimensões $N = 27, 53$ e 129 e diferentes funções de ativação (degrau, sigmoide e base radial), resultando em um total de 9 resultados. Nos treinamentos com a função degrau foram realizadas 3000 iterações enquanto que para a funções sigmoide e de base radial foram realizadas 10000 iterações. Estes números foram escolhidos ao analisar a curva de convergência para o treinamento em todos os casos.

Outro fator a ser considerado no treinamento foi o passo de adaptação. Nos treinamentos com a função degrau o valor utilizado foi igual a 1 (neste caso, não foi possível observar mudanças significativas ao se utilizar um valor diferente de 1 para o passo de adaptação). Já com a função sigmoide, o passo de adaptação utilizado foi igual a 3 e com a função de base radial, o valor foi de 0,1. A escolha destes valores foi feita ao realizar diversos treinamentos com diferentes passos de adaptação para cada função de ativação. Os valores escolhidos foram os que obtiveram um menor erro de treinamento após a convergência da rede.

III. Resultados

Para testar o funcionamento do programa foram utilizadas 7 placas de sinalização, ilustradas pela Figura 8, totalizando 61 letras. Os resultados obtidos são apresentados pela Tabela I, considerando a *feature* com $N = 27$ elementos. Pela Tabela II, considerando a *feature* com $N = 53$ elementos. E pela Tabela III, considerando a *feature* com $N = 129$ elementos.

Nestas tabelas, a linha "corretos" indica as letras que foram classificadas com apenas uma ativação para a função degrau e com a maior probabilidade de correspondência (maior valor de saída) para a função sigmoide e de base radial.

Para a função degrau, a linha "2" indica o número de casos em que houve a ativação de duas letras, sendo uma delas a correta. Para as funções de base radial e sigmoide, indica que a saída correspondente à letra correta apresentou o segundo maior valor de saída, e assim por diante.

Já a linha "incorretos" indica o número de letras que não foram identificadas ou que foram identificadas incorretamente. E as últimas colunas apresentam os resultados obtidos pela ferramenta de OCR Tesseract, sendo que as letras das placas são classificadas, por esta ferramenta, apenas como corretas ou incorretas.

TABLE I
TABELA REFENTE AOS RESULTADOS DA *feature* COM 27 ELEMENTOS.

	Degrau		Sigmoide		Base Radial		Tesseract	
Corretos	10	16,4%	28	45,9%	26	42,6%	48	78,7%
2	5	8,2%	7	11,5%	8	13,8%	0	0%
3	8	13,1%	3	4,9%	5	8,2%	0	0%
4	0	0%	1	1,6%	0	0%	0	0%
+4	7	11,5%	0	0%	2	3,3%	0	0%
Incorretos	31	50,8%	22	36,1%	20	32,8%	13	21,3%

TABLE II
TABELA REFENTE AOS RESULTADOS DA *feature* COM 53 ELEMENTOS.

	Degrau		Sigmoide		Base Radial		Tesseract	
Corretos	10	16,4%	38	62,3%	31	50,8%	48	78,7
2	11	18%	5	8,2%	8	13,1%	0	0%
3	4	6,6%	2	3,3%	2	3,3%	0	0%
4	3	4,9%	0	0%	2	3,3%	0	0%
+4	9	14,8%	0	0%	0	0%	0	0%
Incorretos	24	39,3%	16	26,2%	18	29,5%	13	21,3%

TABLE III
TABELA REFENTE AOS RESULTADOS DA *feature* COM 129 ELEMENTOS.

	Degrau		Sigmoide		Base Radial		Tesseract	
Corretos	15	24,6%	39	63,9%	38	63,3%	48	78,7
2	14	23%	10	16,4%	7	11,5%	0	0%
3	6	9,8%	2	3,3%	3	4,9%	0	0%
4	1	1,6%	0	0%	0	0%	0	0%
+4	6	9,8%	0	0%	0	0%	0	0%
Incorretos	19	31,1%	10	16,4%	13	21,3%	13	21,3%

Através dos resultados apresentados pelas Tabelas I, II e III é possível observar que, ao aumentar a dimensão da *feature*, os resultados foram melhorando, levando à um número maior de caracteres identificados corretamente e à um número menor de caracteres identificados incorretamente, para todas as funções de ativação.

Analisando as funções de ativação, os melhores resultados foram obtidos com a função sigmoide, seguido pela função de base radial. Já a função degrau foi a que apresentou os piores resultados, com uma diferença bastante considerável, principalmente com relação ao número de caracteres identificados corretamente. Desta forma é possível observar que, para o problema proposto, as funções sigmoide e de base radial foram as que tiveram uma maior capacidade de generalização.

Quando comparado com o Tesseract, o número de caracteres identificados corretamente foi bem menor, sendo de $63,9\%$ para o classificador utilizando a função sigmoide e uma *feature* com 129 elementos e $78,7\%$ com o Tesseract. No entanto, se

Fig. 8. Placas utilizadas na etapa de treinamento e placas utilizadas na fase de teste, separadas pela linha tracejada.

considerarmos as vezes em que a saída correspondente à letra correta apresentou o segundo ou o terceiro maior valor de saída, temos que o classificador utilizando a função sigmoide e uma *feature* com 129 elementos se saiu melhor do que a ferramenta Tesseract, tendo uma taxa de incorretos inferior.

IV. Conclusões

Este trabalho de iniciação científica comparou o uso de diferentes funções de ativação em uma rede neural com apenas uma camada para o problema de OCR. Apesar dos resultados serem preliminares e terem considerado um número reduzido de caracteres utilizados na etapa de testes, foi possível perceber que as funções de ativação sigmoide e de base radial apresentaram uma capacidade de generalização bem maior quando comparada com a função degrau. Além disso observou-se que ao aumentar a dimensionalidade das *features* os resultados apresentaram melhoras significativas. Como perspectivas para trabalhos futuros está a utilização de um banco de dados público e também na utilização de outros classificadores como, por exemplo, redes neurais multicamadas.

References

[1] R. E. W. C. R. Gonzalez, *Processamento Digital de Imagens*, 3rd ed. Pearson, 2010.

[2] J. R. Parker, *Algorithms for Image Processing and Computer Vision*, 2nd ed. Willey, 2010.

[3] H. Lin, P. Yang, and F. Zhang, "Review of scene text detection and recognition," *Archives of Computational Methods in Engineering*, Jan 2019. [Online]. Available: https://doi.org/10.1007/s11831-019-09315-1

[4] S. Singh, "Optical character recognition techniques: A survey," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 2, no. 6, pp. 2009 – 2015, 2013.

[5] ——, "A brief review of classifiers used in ocr applications," *International Journal of Computer Trends and Technology*, vol. 34, no. 2, pp. 80 – 88, 2016.

[6] B. V. Kakani, D. Gandhi, and S. Jani, "Improved ocr based automatic vehicle number plate recognition using features trained neural network," in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, July 2017, pp. 1–6.

[7] R. Anugrah and K. Bintoro, "Latin letters recognition using optical character recognition to convert printed media into digital format," *Jurnal Elektronika dan Telekomunikasi*, vol. 17, no. 2, pp. 56–62, 2017. [Online]. Available: https://www.jurnalet.com/jet/article/view/163

[8] M. Liao, B. Shi, X. Bai, X. Wang, and W. Liu, "Textboxes: A fast text detector with a single deep neural network," *CoRR*, vol. abs/1611.06779, 2016. [Online]. Available: http://arxiv.org/abs/1611.06779

[9] M. Liao, B. Shi, and X. Bai, "Textboxes++: A single-shot oriented scene text detector," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3676–3690, Aug 2018.

[10] "Otsu's method," Wikipedia, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Otsu%27s_method

[11] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan 1979.

[12] Øivind Due Trier, A. K. Jain, and T. Taxt, "Feature extraction methods for character recognition-a survey," *Pattern Recognition*, vol. 29, no. 4, pp. 641 – 662, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0031320395001182

[13] T. B. L. A. P. Braga, A. P. L. F. De Carvalho, *Redes Neurais Artificiais Teoria e Aplicações*, 2nd ed. LTC, 2007.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.