Trabalho Final de Processamento Digital de Sinais
Classificador de Dígitos

Hugo Silveira Sousa, 378998. *hugosousa111@gmail.com.*

I. RESUMO

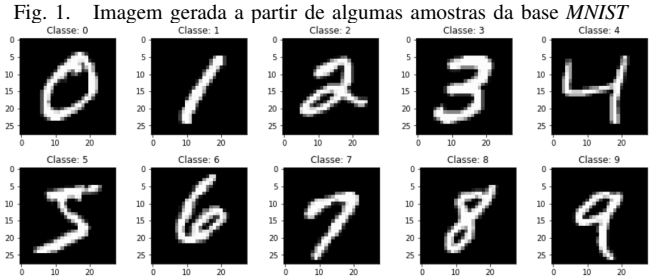
O trabalho trata do desenvolvimento de um classificador de dígitos, utilizando a base de imagens *MNIST*. Esse projeto contém dois algoritmos, no primeiro, durante o pré-processamento dos dados é realizado a convolução com um filtro, e no segundo algoritmo, a imagem passa pela transformada de fourier, e depois é realizado uma filtragem da imagem no espectro da frequência. O objetivo do trabalho é entender os efeitos dos filtros nas imagens, e fazer a análise dos resultados dos classificadores, comparando os melhores filtros, percebendo se esses processos melhoram os resultados dos classificadores.

II. INTRODUÇÃO

A área de visão computacional vem crescendo bastante nos últimos anos, tendo o potencial de resolver inúmeros problemas, de diversas áreas, como nas indústrias, com automatização de processos, na medicina, com a análise de imagens médicas, nos carros autônomos, como na identificação de pedestres, dentre outras áreas. O trabalho trata do desenvolvimento de um classificador de dígitos, utilizando a base de dados *MNIST*, com foco na filtragem da base, através de filtros de convolução e da transformada de fourier.

A base *MNIST* contém 70.000 amostras, que são imagens de números escritos a mão, com 10 classes, de 0 à 9, como mostrado na Fig. 1.

Fig. 1. Imagem gerada a partir de algumas amostras da base *MNIST*



Fonte: Autoria Própria

O algoritmo vai dividir a base em treinamento e teste, com a base de treinamento, o algoritmo vai extrair características, treinar um algoritmo de classificação, e por fim, na fase de testes, prever as classes das imagens da base de testes.

III. FUNDAMENTAÇÃO TEÓRICA

A. Convolução

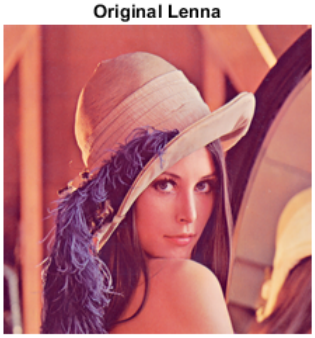
A convolução de imagens deve ser analisada para o caso de 2 dimensões, a equação a seguir representa a função de convolução:

$$C(j,k) = \sum_p \sum_q A(p,q)B(j-p+1, k-q+1) \quad (1)$$

Onde A é o filtro, B é a imagem, e p e q percorrem todas as posições das duas matrizes.

A seguir, é apresentada a aplicação dessa equação, em uma imagem. A imagem original usada para a demonstração é a seguinte:

Fig. 2. Fonte: *https://en.wikipedia.org/wiki/Lenna*



Para aplicar os filtros, foi analisado apenas a imagem em tons de cinza, ficando da seguinte forma:

Fig. 3.

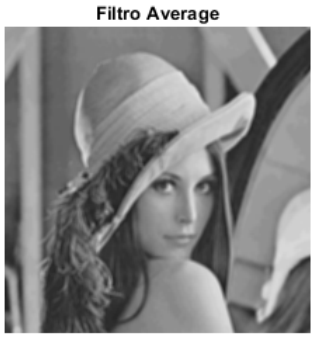
Os filtros a seguir foram encontrados através da função *fspecial* do *MATLAB*, as seguintes imagens demonstram o efeitos desses filtros, quando convoluidos com uma imagem.

Filtro *Average*:

0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

Imagem filtrada com o *average*:

Fig. 4.



Filtro *Prewitt*:

1	1	1
0	0	0
-1	-1	-1

Imagem filtrada com o *prewitt*:

Fig. 5.

Filtro *Disk*:

0.02	0.14	0.02
0.14	0.31	0.14
0.02	0.14	0.02

Imagem filtrada com o *disk*:

Fig. 6.

Pode-se perceber que os filtros *average* e *disk* deixaram as imagens menos definida, já com a filtragem com o *prewitt*, houve bastante mudança na imagem, destacando as áreas com tons mais escuros.

B. Transformada de Fourier

A transformada de fourier de uma imagem também deve ser analisada para o caso de 2 dimensões, as equações a seguir representa a função da transformada:

$$Y_{p+1,q+1} = \sum_{j=0}^{m-1} \sum_{k=0}^{n-1} w_m^{jp} w_n^{kq} X_{j+1,k+1} \tag{2}$$

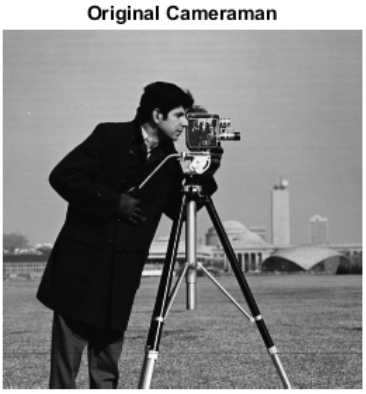
$$w_m = e^{-2\pi i/m} \tag{3}$$

$$w_n = e^{-2\pi i/n} \tag{4}$$

Onde *X* é a imagem que vai ser transformada, e *m* e *n* são as dimensões de *Y*.

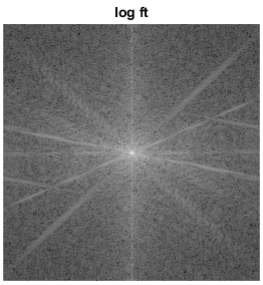
A seguir, é apresentado a aplicação dessa equação, em uma imagem. A imagem original usada para a demonstração é a seguinte:

Fig. 7. Fonte: *MATLAB*



Depois de transformada a imagem fica da seguinte maneira:

Fig. 8.



Na última imagem, os pontos mais próximos ao centro da imagem representam as frequências mais baixas, e os pontos mais afastados do centro, as frequências mais altas.

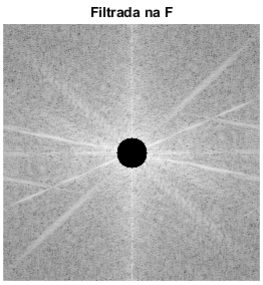
Agora com a imagem no domínio da frequência pode ser realizado filtros passa-baixa e passa-alta.

Filtro Passa-Alta:

Fig. 9.

Para filtrar a imagem, é efetuado uma multiplicação do filtro passa-alta, com a imagem transformada, obtendo o seguinte resultado:

Fig. 10.



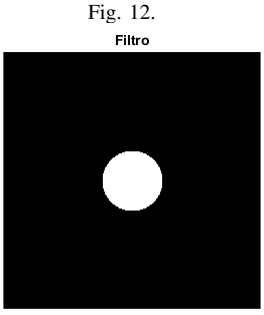
A ultima imagem mostra que as baixas frequências, foram ignoradas, ficando só as frequências mais altas que o corte do filtro. Depois pode ser feito a transformada inversa dessa imagem, tendo como resultado:

Fig. 11.

Nesse resultado nota-se um destaque nas bordas do objetos da imagem, pois o filtro passa-alta destaca as altas frequências da imagem, que representa no domínio da imagem, as áreas onde tem maior mudança nos tons, as bordas.

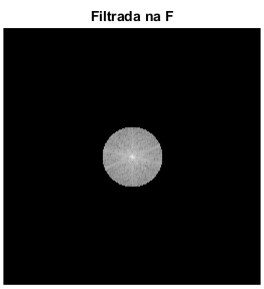
Filtro Passa-Baixa:

Fig. 12.



Depois da multiplicação com o filtro passa-baixa:

Fig. 13.



Esse é o contrário do processo anterior, a ultima imagem mostra que as altas frequências, foram ignoradas. Depois da transformada inversa, é obtido o seguinte resultado:

Fig. 14.

Nesse resultado, o processo inverso aconteceu, nota-se que as bordas estão menos destacadas do que o resto da imagem.

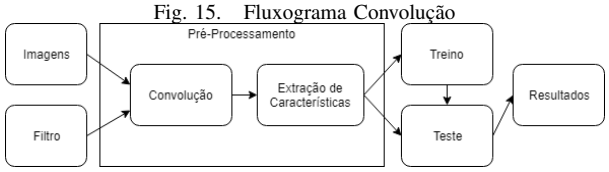
IV. OBJETIVOS

- Utilizar de forma prática os conceitos aprendidos na disciplina de Processamento Digital de Sinais;
- Compreender o significado dos conceitos teóricos quando aplicados em um sistema real;
- Construir um classificador de dígitos;
- Entender os efeitos da filtragem de imagens;
- Analisar os resultados dos classificadores.

V. METODOLOGIA

Os códigos foram implementados utilizando a ferramenta *MATLAB*. Primeiro vai ser analisado o código que utiliza convolução, o fluxograma da Fig. 2, mostra de forma gráfica os passos do sistema.

Fig. 15. Fluxograma Convolução

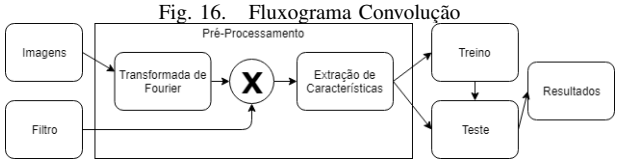


Fonte: Autoria Própria

Inicialmente, cada imagem é convoluida com um filtro, depois essas imagens filtradas são extraídas suas características, treinando um algoritmo com esses atributos, e por fim testando o classificador e obtendo os resultados, no caso o principal resultado são acurácias obtidos nas previsões.

Agora será analisado o código que utiliza transformada de fourier, o fluxograma da Fig. 3 apresenta o sistema.

Fig. 16. Fluxograma Convolução



Fonte: Autoria Própria

Cada imagem é transformada para o domínio da frequência, depois são multiplicadas com um filtro, e apartir desse ponto, se comporta como o algoritmo da convolução, com extração de suas características, finalizando com o treinamento e o teste.

Os algoritmos tem as seguintes estruturas:

A. Caso com Convolução

Passo 1: Carrega a base *MNIST* e permuto as imagens.

Passo 2: Separa a base com 80% para treino e 20% para teste.

Passo 3: Faz a convolução de todas as imagens com um filtro, no caso, foram escolhidos 3 filtros para serem analisados, o filtro *Average*, *Prewitt* e *Disk*. Esses filtros foram obtidos através da função *fspecial* do *MATLAB*.

Passo 4: É realizado a extração de atributos da base de imagens filtradas, são extraídos 40 atributos, utilizando a função *sparsefilt* do *MATLAB*, essa função retorna um modelo de atributos da base.

Passo 5: É feito o treino com um algoritmo de classificação, no caso, foram escolhidos os classificadores *KNN*, *Naive Bayes* e *Decision Tree*, utilizando as funções *fitcKNN*, *fitcnb*, *fitctree*, respectivamente.

Passo 6: Testa o classificador com a base de atributos de teste, e calcula os valores de acurácia.

B. Caso com Transformada de Fourier

Passo 1: repete-se o Passo 1 do caso anterior.

Passo 2: repete-se o Passo 2 do caso anterior.

Passo 3: Realiza a transformada de fourier de todas as imagens da base.

Passo 4: Escolhe um filtro passa-baixa ou passa-alta, e multiplica pela base que está no domínio da frequência.

Passo 5: repete-se o Passo 4 do caso anterior.

Passo 6: repete-se o Passo 5 do caso anterior.

Passo 7: repete-se o Passo 6 do caso anterior.

VI. RESULTADOS

Os resultados que serão apresentados estão divididos entre os algoritmos dos casos apresentados anteriormente.

Legenda dos gráficos:

- Filtro 1 = *Average*;
- Filtro 2 = *Prewitt*;
- Filtro 3 = *Disk*;
- Filtro *Low* = Passa-baixa;
- Filtro *High* = Passa-alta.

A. Resultado 1: Convolução, KNN.

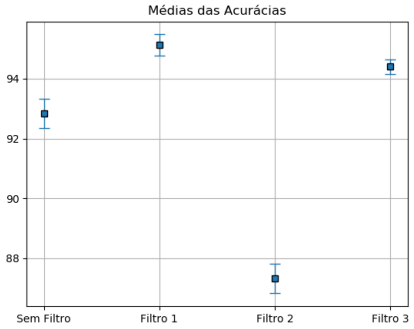


Fig. 17. Grafico 1

Média das Acurácias: Sem Filtro = 92.8414%; Filtro 1: 95.1329%; Filtro 2: 87.3271%; Filtro 3: 94.4071%.

B. Resultado 2: Convolução, Naive Bayes.

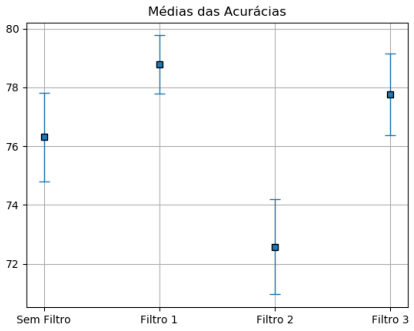


Fig. 18. Grafico 2

Média das Acurácias: Sem Filtro = 76.3129%; Filtro 1: 78.7800%; Filtro 2: 72.5786%; Filtro 3: 77.7700%.

C. Resultado 3: Convolução, Decision Tree.

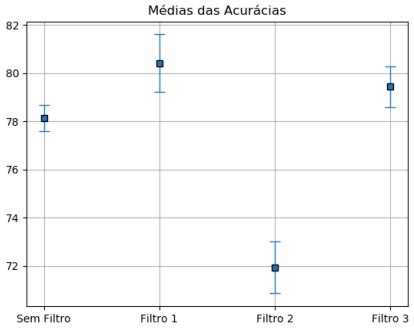


Fig. 19. Grafico 3

Média das Acurácias: Sem Filtro = 78.1286%; Filtro 1: 80.4086%; Filtro 2: 71.9357%; Filtro 3: 79.4343%.

D. Análise dos resultados 1, 2 e 3.

Pode-se perceber uma melhora nos resultados das acurácias quando se usa os filtros *Average* e *Disk*, sendo interessante usá-los para classificação dessa base. Também é visível uma diminuição do valor com o filtro *Prewitt*, isso se deve ao fato de quando é realizado a convolução das imagens com o *Prewitt*, há bastante perda de informações da imagem.

E. Resultado 4: Fourier, KNN.

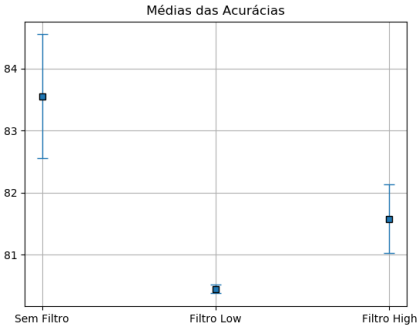


Fig. 20. Grafico 4

Média das Acurácias: Sem Filtro = 83.5543%; Filtro *Low*: 80.4443%; Filtro *High*: 81.5800%.

F. Resultado 5: Fourier, Naive Bayes.

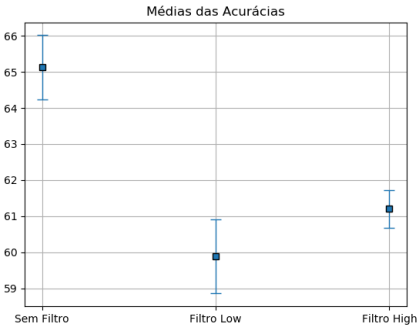


Fig. 21. Grafico 5

Média das Acurácias: Sem Filtro = 65.1271%; Filtro *Low*: 59.8786%; Filtro *High*: 61.1986%.

G. Resultado 6: Fourier, Decision Tree.

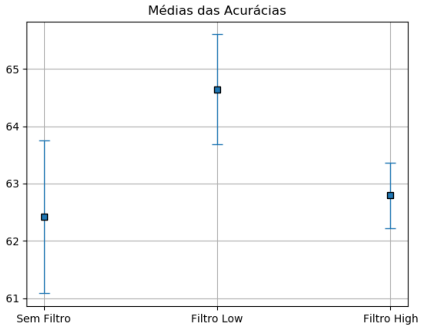


Fig. 22. Grafico 6

Média das Acurácias: Sem Filtro = 62.4157%; Filtro *Low*: 64.6457%; Filtro *High*: 62.7943%.

H. Análise dos resultados 4, 5 e 6.

Nos dois filtros os resultados das acurácias são menores do que o caso sem filtro, mostrando que não é interessante usar esse tipo de filtro para classificação dessa base. Isso acontece pela mesma razão do caso do *Prewitt*, há bastante perda de informações da imagem, o que não favorece para o treinamento.

VII. CONCLUSÃO

Com a obtenção dos resultados, foi identificado alguns casos onde os filtros melhoraram a classificação dos dígitos.

No caso do algoritmo com a convolução, os filtros que fazem uma espécie de média da imagem, obtém melhores resultados de acurácia, já o filtro prewitt, que destaca algumas regiões da imagem, não se mostrou interessante para fazer a classificação, esse tipo de filtro é mais aproveitado quando usado em imagens com maior número de detalhes e que seja necessário treinar com essas imagens. Para a base *MNIST*, que tem imagens simples, com poucos detalhes, as imagens filtradas perderam características essenciais da imagem.

O algoritmo que usa filtros na frequência não obteve bons resultados na fase teste, pois as imagens filtradas também perderam características relevantes para o treinamento, concluísse que esse tipo de filtro não é interessante para aplicações de classificação.

Os códigos desse trabalho e outras implementações sobre o assunto encontram-se no seguinte repositório: *https://github.com/hugosousa111/classificador_numeros*.

REFERENCES

[1] MATHWORKS. Documentation. Disponível em: https://www.mathworks.com/help/. Acesso em: 26 de Junho de 2019.

[2] MATHWORKS. conv2. Disponível em: https://www.mathworks.com/help/matlab/ref/conv2.html. Acesso em: 26 de Junho de 2019.

[3] MATHWORKS. fft2. Disponível em: https://www.mathworks.com/help/matlab/ref/fft2.html. Acesso em: 26 de Junho de 2019.

[4] MATHWORKS. fitcknn. Disponível em: https://www.mathworks.com/help/stats/fitcknn.html. Acesso em: 26 de Junho de 2019.

[5] MATHWORKS. fitcnb. Disponível em: https://www.mathworks.com/help/stats/fitcnb.html. Acesso em: 26 de Junho de 2019.

[6] MATHWORKS. fitctree. Disponível em: https://www.mathworks.com/help/stats/fitctree.html. Acesso em: 26 de Junho de 2019.

[7] MATHWORKS. fspecial. Disponível em: https://www.mathworks.com/help/images/ref/fspecial.html. Acesso em: 26 de Junho de 2019.

[8] MATHWORKS. sparsefilt. Disponível em: https://www.mathworks.com/help/stats/sparsefilt.html. Acesso em: 26 de Junho de 2019.

[9] YOUTUBE. Wiki YouTube. Fast Fourier Transform of an Image in Matlab (TUTORIAL). Disponível em: https://bit.ly/2JdwCmJ. Acesso em: 26 de Junho de 2019.

[10] YOUTUBE. Rashi Agrawal. Image processing using matlab. Disponível em: https://bit.ly/2FxivaQ. Acesso em: 26 de Junho de 2019.

[11] YOUTUBE. Robert Martin. Viewing mnist data using matlab. Disponível em: https://www.youtube.com/watch?v=9onT6_pz4ks. Acesso em: 26 de Junho de 2019.

[12] YOUTUBE. Nuruzzaman Faruqui. Convolutional Neural Network in Matlab. Disponível em: https://www.youtube.com/watch?v=ZOXOwYUVCqw. Acesso em: 26 de Junho de 2019.

[13] UTKARSH SINHA. Image convolution examples. Disponível em: http://aishack.in/tutorials/image-convolution-examples/. Acesso em: 26 de Junho de 2019.

[14] VICTOR POWELL. Image Kernels. Explained Visually. Disponível em: http://setosa.io/ev/image-kernels/. Acesso em: 26 de Junho de 2019.

[15] SUMIT SAHA. A Comprehensive Guide to Convolutional Neural Networks. Disponível em: http://twixar.me/XV4n. Acesso em: 26 de Junho de 2019.

[16] R. FISHER, S. PERKINS, A. WALKER AND E. WOLFART. Gaussian Smoothing. Disponível em: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm. Acesso em: 26 de Junho de 2019.

[17] R. FISHER, S. PERKINS, A. WALKER AND E. WOLFART. Fourier Transform. Disponível em: https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm. Acesso em: 26 de Junho de 2019.

[18] MNIST in CSV. Disponível em: https://pjreddie.com/projects/mnist-in-csv/. Acesso em: 26 de Junho de 2019.