Classificação de imagens utilizando o dataset *MNIST*

Alexandre H. Borba
Departamento de Computação
Universidade Estadual de Londrina
Londrina, Brasil
alexandrehborba@gmail.com

***Abstract*—Python MNIST recognition using image classification pipeline.**

***Index Terms*—Python, mnist,neural network, classification pipeline.**

I. Introdução

Seres humanos possuem são capazes de distinguir e classificar objetos de forma simples, como por exemplo reconhecer a figura de um pássaro e classificar de acordo com sua especie. Porém quando o objeto a ser classificado se torna mais complexo seres humanos se demonstram limitados a realizar esta tarefa, por exemplo, somos incapazes de determinar com precisão o grau de contaminação de uma planta por uma determinada bactéria, bem como seu grau de amadurecimento e etc ...

Para isto é necessário um processamento computacional desta imagem, geralmente este processamento é feito a partir de um pipeline de classificação de imagens. Este pipeline pode ser subdivido nas seguintes etapas:

- Aquisição de imagens digitais.
- Pré-processamento.
- Extração de características.
- Criação de um modelo.
- Avaliação do modelo.

Nesta atividade foi proposto o reconhecimento de dígitos feitos a mão utilizando o banco de dados *MNIST*. Para isto, serão feitas duas abordagens, a primeira irá utilizar o pipeline de classificação de imagens, já a segunda irá utilizar apenas os dados da imagem como entrada. Ambas as abordagens irão utilizar uma rede neural artificial para realizar o reconhecimento de dígitos. Por fim será comparado o resultado obtido em cada abordagem.

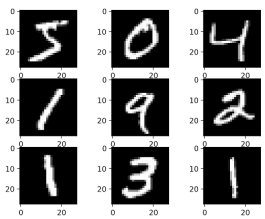

Fig. 1. Exemplo de dígitos do dataset *MNIST*.

II. Rede Neural.

Primeiramente foi criado um modelo de rede neural que pudesse ser utilizado em ambas abordagens, de forma de que facilitasse a comparação entre elas. Para isto foi utilizada a biblioteca *Keras* na linguagem *Python*. Como modelo foi utilizada a classe *Sequential()*, que é o modelo mais simples da biblioteca. Após ser definido o modelo, foram definidas as camadas, sendo adicionadas 3 camadas, duas camadas ocultas e uma ultima de saída. As camadas ocultas possuem uma composição de 50 neurônios cada, utilizando *relu* como função de ativação. A camada de saída por sua vez utiliza 10 neurônios, e sua função de ativação é a *softmax*, o uso desta função tem como finalidade gerar uma distribuição da probabilidade entre as 10 classes existentes. Durante o treinamento da rede neural o calculo do custo é feito através da função *sparse_categorical_crossentropy* e como métrica é utilizada a acurácia.

III. Aquisição de imagens digitais

O processo de aquisição de imagens foi realizado utilizando o banco de dados *MNIST*.
O *MNIST* é um subconjunto do *NIST Special Database 19*, que foi criado a partir da coleta de dados do Departamento do Censo dos Estados Unidos, contendo um conjunto de dígitos e letras. O *MNIST* contém apenas as imagens referentes a dígitos, sendo tratadas e padronizadas no tamanho de 28x28 pixels.

IV. Pré processamento.

O pré processamento é a etapa do pipeline onde a imagem é preparada para os processos seguintes. Na implementação realizada foram destacados os pixels da imagem que compõe o desenho do dígito através da seguinte função *cv2.threshold()* combinada com a flag *THRESH_BINARY_INV*.
Após o formato do numero ser destacado, a imagem também foi recortada, pois alguns dos números presentes no *dataset* estavam deslocados na imagem, podendo interferir no resultado obtido. Desta forma o recorte obtido foi uma imagem 23x23 pixels, pois os valores de saída devem ser padronizados e existiam imagens que ocupavam ao menos 23 pixels na vertical e horizontal.

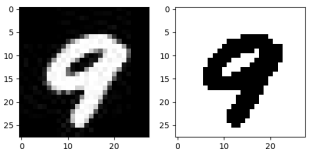

Fig. 2. Resultado obtido ao transformar a imagem original em preto e branco.

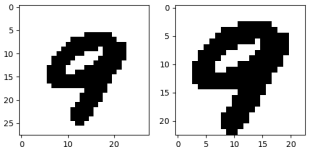

Fig. 3. Imagem recortada

V. Extração de características.

A etapa de extração de características tem como objetivo extrair dados que tenham a capacidade de representar o objeto de interesse, no nosso caso o digito. Inicialmente foi desenvolvido um pré processamento aonde após o digito ser recortado também era subdividido em 4. Após essa subdivisão eram retirados do histograma de cada quadrante que compunha a imagem, os valores estatísticos:

- Média.
- Mediana.
- Moda.
- Assimetria.
- Curtose.

Os valores de media, moda e assimetria foram descartados pois não apresentavam distinção entre as diferentes classes (dígitos). A partir do valores de mediana e curtose foi gerado um modelo, porém não houve convergência. Tendo em vista o fracasso de expressar através de estatísticas as características de uma imagem, foi proposto um novo modelo que aproveitasse o resultado obtido no pré processamento. Desta forma foi aplicada sobre a imagem a transformada de *Fourier* afim de obter o domínio de frequência dos respectivos dígitos.

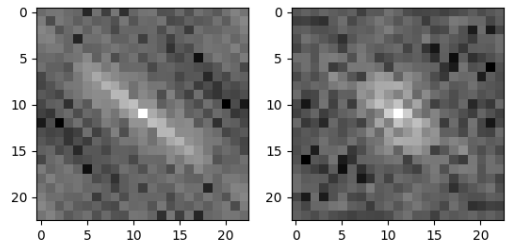

Fig. 4. Comparação entre o fourier do numero 1 e 9

VI. Criação de um modelo

Para a entrada desta rede foi utilizado um dataset contendo 1000 imagens, das quais foi utilizado o dominio de frequencia a partir da transformada de fourier, os pixels resultantes foram representados como um vetor de tamanho 529. Para a saída da rede foi utilizado *hot encoding* no numero da classe (0-9). A fig. 6 apresenta o treinamento da rede com 50 épocas.

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	0	0	0	0

Fig. 5. Exemplo de hot encoding para a classe 5.

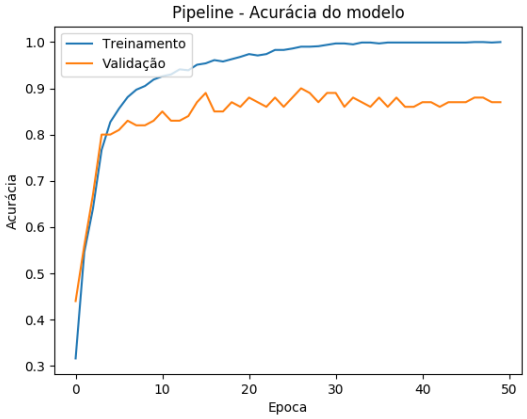

Fig. 6. Treinamento para 50 épocas.

VII. Avaliação do modelo.

Após ser realizado o treinamento do modelo, foi realiza uma avaliação. Esta avaliação utiliza testes com dados aos quais o modelo não foi treinado, sendo utilizado um segundo dataset contendo 100 imagens. O modelo obteve uma Acurácia = 87.99% e a matriz de confusão a seguir:

classes	0	1	2	3	4	5	6	7	8	9
0	100	0	0	0	0	0	0	0	0	0
1	0	100	0	0	0	0	0	0	0	0
2	0	0	90.91	0	0	0	0	0	9.09	0
3	0	0	0	100	0	0	0	0	0	0
4	0	0	9.09	0	90.91	0	0	0	0	0
5	0	0	0	0	0	87.5	0	0	9.09	0
6	0	0	9.09	0	0	0	91.67	0	0	0
7	0	0	0	0	0	0	0	100	0	0
8	0	10	0	0	0	37.5	8.33	0	54.55	0
9	0	0	0	0	18.18	0	0	20	9.09	69.23

Fig. 7. Matriz de confusão obtida.

VIII. Implementação com Rede Neural

Após serem realizados os testes da abordagem com pipeline foi realizada uma outra implementação, todos os detalhes da rede neural, camadas, neurônios e função de ativação foram mantidos. A diferença entre a utilização da rede neural está na entrada da mesma, enquanto a rede neural anterior processava as *features* extraídas da imagem, esta tem como entrada todos pixels da imagem, sem nenhum tipo de tratamento. O resultado obtido foi uma acurácia de 83.99% , a fig. 7 e 8 apresentam o treinamento para 50 épocas.

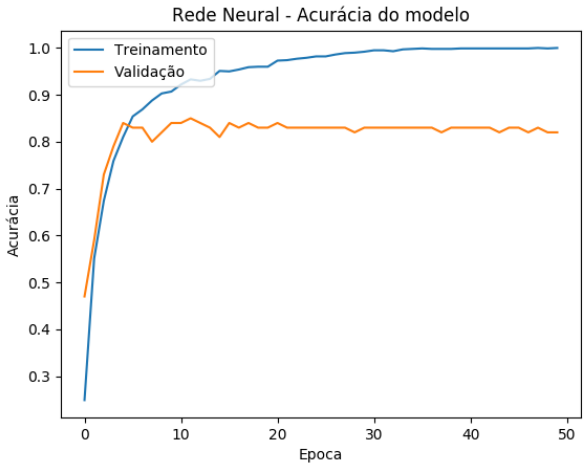

Fig. 8. Treinamento para 50 épocas.

classes	0	1	2	3	4	5	6	7	8	9
0	90.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	100	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	85.0	40.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	80.0	0.0	0.0	7.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	75.0	0.0	0.0	0.0	0.0	20.0
5	0.0	0.0	0.0	20.0	0.0	71.0	13.0	0.0	20.0	0.0
6	0.0	0.0	8.0	0.0	0.0	0.0	93.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	80.0	0.0	20.0
8	0.0	0.0	0.0	20.0	0.0	0.0	0.0	0.0	60.0	10.0
9	0.0	0.0	0.0	0.0	25.0	0.0	0.0	0.0	20.0	70.0

Fig. 9. Treinamento para 50 épocas.

IX. Conclusão

Ao utilizar o pipeline de classificação de imagens foi possível diminuir consideravelmente a quantidade de dados de entrada para a rede neural (529 contra 784) e mesmo com essa diminuição da quantidade de dados não houve perda de representatividade, pois o resultado obtido foi (87.99% contra 83.99%). Desta forma, concluí se que com a utilização do pipeline de classificação de imagens é possível obter resultados similares, ou até mais efetivos se comparada as outras abordagens, também foi constatada a utilização de menos recursos pois as características extraídas tem capacidade de representação de suas classes.

References

[1] PYTHON3.7. Fredericksburg, Virginia: Python Software Foundation, 2019. Retrieved from https://www.python.org/downloads/

[2] Cohen, G., Afshar, S., Tapson, J., van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. Retrieved from http://arxiv.org/abs/1702.05373

[3] Keras. Chollet,François and others 2015. Retrieved from https://keras.io/