Convolutional Neural Network - LeNet-5

Gustavo Aparecido de Souza Viana

***Abstract*— *Convolutional Neural Network* ou mais conhecida como *CNN* é um modelo classificador *Deep Learning* que vem crescendo exponencialmente, com intuíto de desenvolver algoritmos ou aplicações para analisar grande bases de dados e assim tirar conclusões. Esse modelo basedado no conceito matemático de convolução, que está relacionado a uma tipo de operação linear, geralemnte é utilizado para tratamento de imagens. O objetivo deste trabalho é implementar a LeNet-5 utilizando a biblioteca keras, sendo um modelo de *CNN*, para realizar a classificação na base de dados *MNIST*. Os resultados mostraram que o é possível aplicar a *LeNet-5* como classificador com uma boa acurácia mesmo utilizando poucas épocas.**

I. INTRODUÇÃO

A necessidade de classificações está sendo bem comum na atualidade, podemos dar o exemplo de uma empresa que necessita predizer qual estado da máquina (bom, normal, ou ruim) e assim com base no histórico dela e com base nos dados atuais, conseguimos classificar a mesma.

Conseguimos predizer algum informação baseada em uma base conhecida contendo resultados utilizando machine learning ou com métodos estatísticos. É de extrema importância conhecer o ramo de atividade da aplicação pois métodos de machine learning geralmente estão relacionados a rede neural, consquentemente necessitam de mais dados para que as predições tenham uma maior acurácia. De contra partida, métodos estatísticos não necessitam.

Neste trabalho foi abordado a implementação do *LeNet-5*, iniciado por *Le Cun* em [2] baseado em neurônios inspirado pela organização visual do Córtex animal, com objetivo de classificar a base de dados *MINIST*. Os testes foram feitos em 10 épocas com a atualização dos pesos a partir de 128 execuções.

II. CONCEITOS FUNDAMENTAIS

Nesta seção, será apresentado os conceitos básicos utilizados para a *LeNet*.

A. Convolutional Neural Network

Como citado na introdução o *Convolutional Neural Network* iniciado por [1] que segue o modelo de aplicação matemática convolucional, que comumente aplicado em aprendizado de máquina e reconhecimento de imagens. A ideia geral desse modelo é uma simples rede neural que utiliza a combinação de camadas chamadas de *Convolutional Layer*, de *Pooling Layer*.

Nas camadas de *Convolutional* há uma multiplicação de matrix $X * Matrix_{convolution}$, sendo X a entrada da imagem ou dado, como podemos ver na Figura 1 que exemplica o processo de convolução.

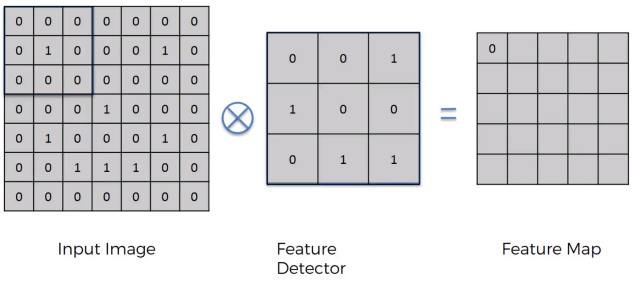

Fig. 1: Exemplo de uma convolução matricial - *Convolutional Layer*

Nas camadas de *Pooling* não seguem a ideia de convolução, onde necessita de uma multiplicação da entrada e máscara. O processo consiste em obter o maior valor na região da máscara como podemos ver na Figura 2.

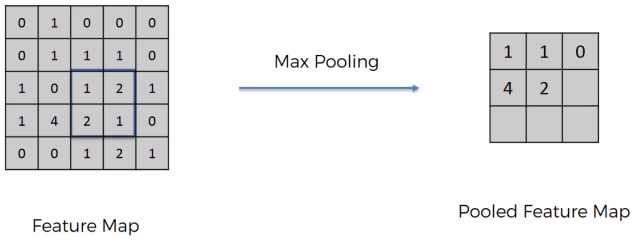

Fig. 2: Exemplo de um ”filtro” - *Pooling Layer*

E por fim a última camada chamada de *Fully Connected* segue o modelo de *Multilayer Perceptron*. A Figura 3 exemplifica um modelo completo de *CNN*.

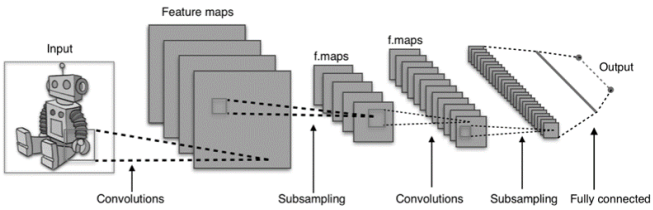

Fig. 3: Exemplo de um *CNN*

B. LeNet

Existem algumas arquiteturas de *CNN* pré-definidas pela literatura, uma delas é a LeNet criado por [2], que passou por transformações desde a *LeNet-1* e atualmente se encontra na *LeNet-5*. Ela é composta por camadas esparsas, convolucionais e de *max-pooling*, as camadas mais acimas são totalmente conectadas e seguem o modelo tradicional de

MLP (camada escondida + regressão logística). A Figura 4 mostra a arquitetura utilizada pela *LeNet-5*.

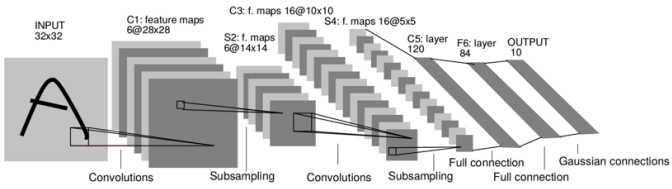

Fig. 4: Arquitetura da *LeNet-5*

A Figura 5 exemplifica minimamente o processo de utilizado pela *LeNet-5*.

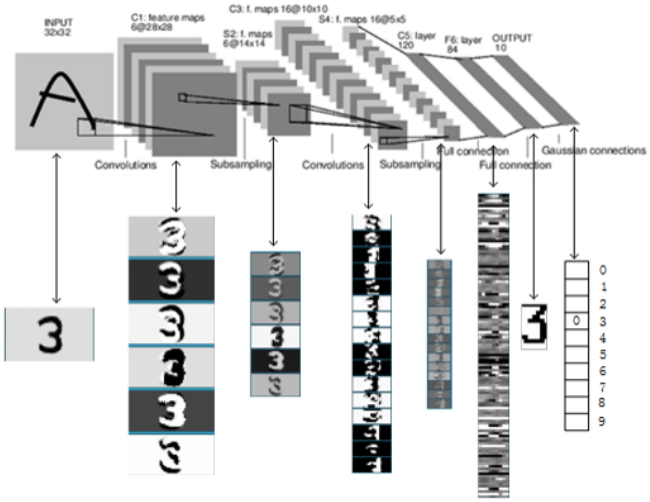

Fig. 5: Exemplo - *LeNet-5*

III. BASE DE DADOS

Nesta seção será apresentado as bases de dados utilizadas para a execução dos experimentos.

É uma grande base de dados do *Instituto Nacional de Padrões e Tecnologia Modificado* criada por [3] que contém digítos manuscritos que geralmente é utlizado para treinamento de vários sistemas de processamento de imagem e aprendizado de máquina.

A base de dados contém 60 mil imagens de treinamento e 10 mil de teste. Além disso, as imagens são em preto e branco com tamanho de 28x28 pixels e anti-alias, o que introduziu níveis de escala de cinza.

Fig. 6: Exemplo da base de dados *MNIST*

IV. METODOLOGIA

Nesta seção será apresentado a metodologia utilizada para implementar a *LeNet-5* juntamente, implementado em Python utilizando a biblioteca *Keras*. O código fonte pode ser encontrado em https://github.com/apparecidoo/master-special-learning-topic.

O pseudocódigo 1 utilizado para a criação da arquitetura utilizada pelo *LeNet-5*.

Algorithm 1: new Lenet()

```
lenet = keras.models.Sequential()
lenet.add(keras.layers.Conv1D( filters=6,
  kernel_size=5, activation='relu', use_bias=True,
  input_shape=(28, 28)))
lenet.add(keras.layers.MaxPooling1D(pool_size=2,
  strides=2))
lenet.add(keras.layers.Conv1D(filters=16,
  kernel_size=5, activation='relu', use_bias=True))
lenet.add(keras.layers.MaxPooling1D(pool_size=2,
  strides=2))
lenet.add(keras.layers.Flatten())
lenet.add(keras.layers.Dense(120, activation='relu',
  use_bias=True))
lenet.add(keras.layers.Dense(84, activation='relu',
  use_bias=True))
lenet.add(keras.layers.Dense(10, activation='softmax',
  use_bias=True))
lenet.compile(
  loss=keras.losses.categorical_crossentropy,
  optimizer=keras.optimizers.Adadelta(),
  metrics=['accuracy'] )
return lenet
```

O pseudocódigo 2 e 3 utilizado para treinamento e predição respectivamente pelo *LeNet-5*.

Algorithm 2: Fit(x_train, y_train, epochs, batch_size)

```
return lenet.fit(x_train, y_train, epochs=epochs,
  batch_size=batch_size)
```

Algorithm 3: Predict(x_test)

```
return lenet.predict(x_test)
```

V. EXPERIMENTOS E RESULTADOS

Nesta seção será apresentado os experimentos e seus respectivos resultados.

O experimento consiste na aplicação do *LeNet-5* para a base de dados *MNIST*, com 60 mil imagens de treinamento e 10 mil de teste. Foram realizado o treinamento utilizando a *batch size* $= 128$, e 10 épocas ou iterações.

A Figura 7 representa os resultados graficamente obtidos do experimento. Sendo que o eixo X representa as épocas e o eixo Y a porcentagem, a linha em verde representa a acurácia da *LeNet-5* e a linha em laranja a taxa de perca por cada época.

Fig. 7: Resultado de perda e acurácia - *LeNet-5*

VI. CONCLUSÃO

Neste trabalho foi implementado o modelo de *LeNet-5*, que segue o modelo de uma *CNN* baseada em neurônios inspirado pela organização visual do Córtex animal, para assim executar tarefas computacionais. O *LeNet-5* é utilizado como classificador que segue a arquitetura citada na sessão Conceitos Fundamentais.

Após a análise dos conceitos e resultados obtidos após a aplicação do método *LeNet-5*, podemos concluir que tem capacidade de classificar dados com uma alta acurácia com poucas épocas.

Além disso, analisando o gráfico 7 podemos ver que há uma curva de aprendizado e que em certo momento ela se estabiliza, ou seja, que a acurácia não está relacionada totalmente e proporcionalmente a quantidade de épocas.

REFERENCES

[1] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. 1998.

[3] Y. LeCun and C. Cortes. The mnist database of handwritten digits. 2005.