Projeto 2 - CNN para treinamento e classificação do dataset CIFAR-10

Thauan Leandro Gonçalves

27 de outubro de 2018

1 Introdução

O *Dataset* utilizado neste trabalho foi o CIFAR-10 [2] cujo contém um conjunto de 60000 imagens RGB de 32x32 pixels. As imagens estão separadas em 6 lotes contendo cada uma 10000 imagens podem pertencer a umas das 10 classes: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship* ou *truck*. Além disso, foi utilizado 5 lotes para treinamento e 1 lote para teste.

Cada lote está representado no formato matricial sendo que há 10000 linhas e 3072 colunas (32 x 32 x 3 pixels). Os primeiros 1024 pixels representam os pixels referentes à cor vermelha, sendo que a disposição dos pixels segue a representação *row major order*, ou seja, os primeiro 32 pixels são referentes à primeira coluna, os próximos 32 são referentes à segunda linha, e assim por diante. Os próximos 1024 pixels referem-se ao verde e os últimos 1024 referem-se ao azul. A Figura 1 mostra a representação vetorial de uma imagem RGB.

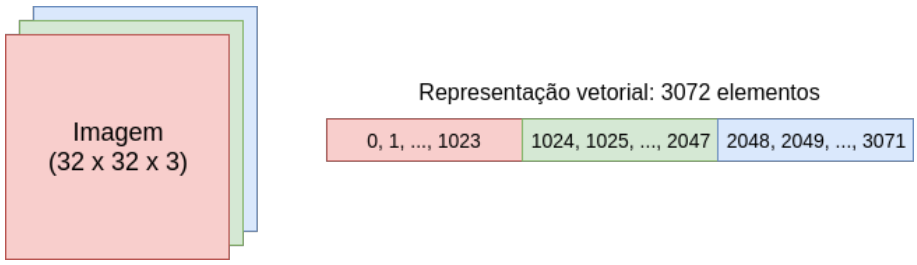

Figura 1: Representação vetorial de uma imagem RGB de 32 por 32 pixels

2 Arquitetura

A arquitetura é composta por duas sub arquiteturas, onde a primeira é responsável por identificar e obter as características das imagens através da aplicação de convoluções, e a outra é responsável por gerar o aprendizado com base nas características obtidas pela primeira arquitetura. As próximas duas seções irão explicar com mais detalhes como cada uma dessas sub arquiteturas foram implementadas.

2.1 Identificação de características

A arquitetura responsável por identificar e obter as características das imagens está mostrada na Figura 2. Como pode ser notado, a arquitetura é razoavelmente complexa e grande, sendo utilizado a plataforma *Google Colab* para realizar o treinamento da CNN. As escolhas do número de camadas e filtros foi baseado no tutorial sobre CNN usando *TensorFlow* [1], sendo realizadas algumas alteração para permitir utilizar imagens com 3 canais(RGB).

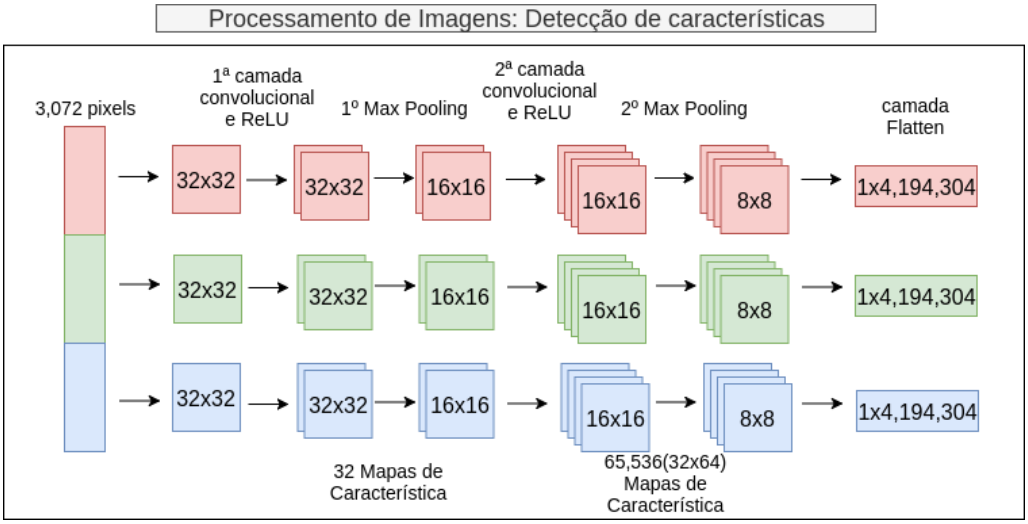

Figura 2: A camada irá receber um array com tamanho de 3072 elementos, sendo que eles representam os valores dos pixels das imagens em rgb. A primeira etapa será separar os pixels em sub matrizes de 32x32 onde cada uma contem os valores referentes a cada uma das cores. Todos os processos seguintes ocorrerão em paralelo e de forma independente para as 3 matrizes. Inicialmente será realizado convoluções com 32 filtros de tamanho 3x3 em cada matriz, seguido pela aplicação da função de ativação *ReLU*, resultando em 32 mapas de características(MC) referentes a cada uma das cores. Logo após é aplicado uma função de *MaxPooling* com stride de 2x2 sobre os MC, resultando em 32 MC com tamanho de 16x16 para cada cor. Será aplicado em seguida mais uma convolução, utilizando 64 filtros de tamanho 3x3 em todos os 32 MC anteriores, seguidos pela aplicação da função *ReLU* novamente, resultando em 65536 (64x32) MC de tamanho 16x16 para cada cor. Logo depois é aplicado novamente a função *MaxPooling*, resultando em matrizes de 8x8. A última etapa desta arquitetura será realizar a linearização das matrizes, criando-se um array com tamanho 4194304 (8x8x65536) para cada cor. Esses arrays serão utilizados pela próxima sub arquitetura.

2.2 Aprendizado

A arquitetura responsável pelo aprendizado esta representada na Figura 3. A determinação do número de camadas e de neurônios também segue como base o tutorial sobre o MNIST, sendo realizadas algumas modificações por causa da estrutura da imagem.

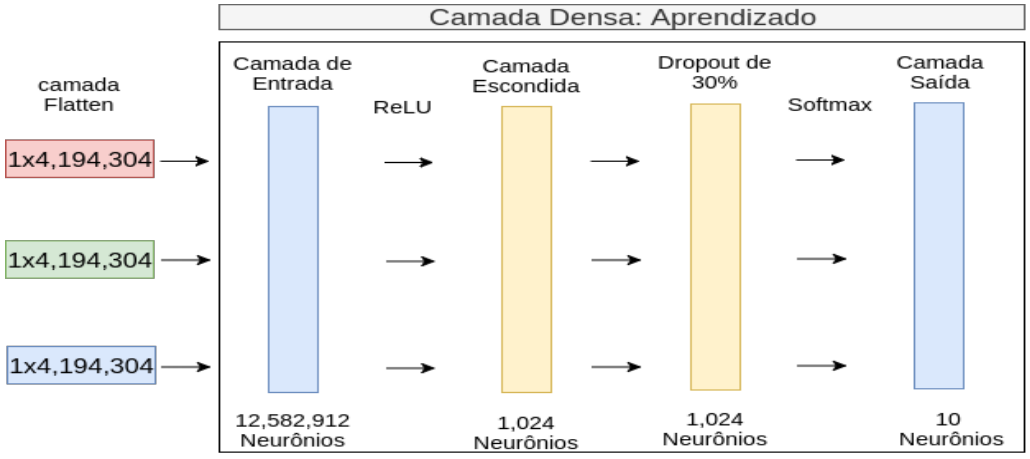

Figura 3: A entrada para esta arquitetura serão os arrays referentes às cores vermelho, verde e azul fornecidos pela arquitetura de identificação de características. Na arquitetura de aprendizado, as características obtidas em relação às cores serão juntadas para tentar se definir uma estrutura que realize boas previsões, sendo definido portanto 12582912(3*4194304) neurônios para a camada de entrada desta arquitetura. Logo depois será aplicado a função de ativação *ReLU* sobre a camada de entrada, chegando à camada escondida, cuja possui 1024 neurônios. Logo após é inserida uma camada de *Dropout* que é responsável por desativar algumas conexões entre os neurônios da camada escondida e a camada de saída, evitando principalemente o *overfitng* do modelo. Por fim, é utiliza a função de ativação *Softmax* que irá determinar a probabilidade da imagem pertencer as umas das 10 classes de saída.

3 Resultados

O modelo treinado neste trabalho foi resultado do treinamento desta arquitetura com 500000 ciclos no *TensorFlow* e no *Google Colab*. Assim, a acurácia sobre o lote de teste foi de 70% e sobre as imagens obtidas na internet também possui uma acurácia de 70%.

Referências

[1] Mark Daoust. Build a convolutional neural network using estimators, 2018.

[2] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009.