Relatório 16 - Redes Neurais Convolucionais 2 (Deep Learning)

Guilherme Loan Schneider

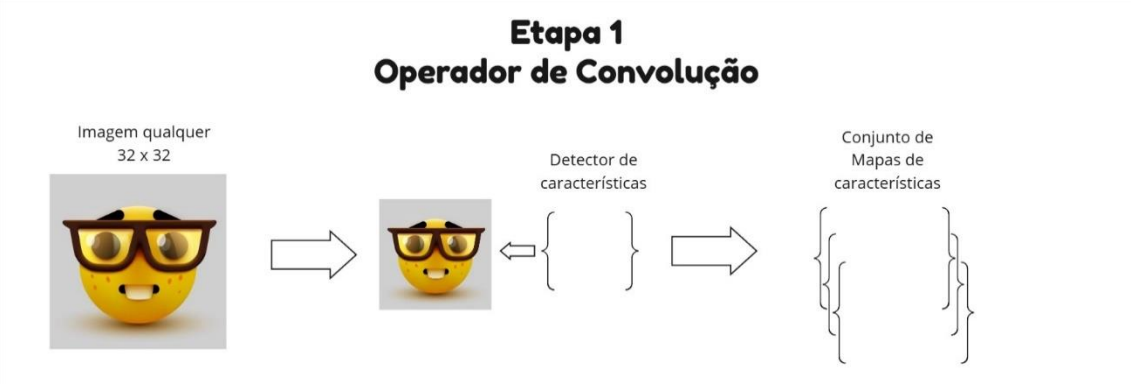
Esse documento contém material do relatório anterior (Card 15) visto que o conteúdo abordado é o mesmo, mas de uma forma mais aprofundada e com alguns conteúdos novos.

Descrição da atividade

As redes neurais convolucionais são, de maneira simplificada, uma rede neural com algumas etapas antes de chegar no processamento tradicional de rede (neurônios de entrada, ocultos e de saída). Esse tipo é principalmente utilizado para reduzir a complexidade de imagens, tendo aplicações no DLSS (Deep Learning Super Sampling) de placas de vídeo da NVIDIA.

Parte Convolucional da rede neural

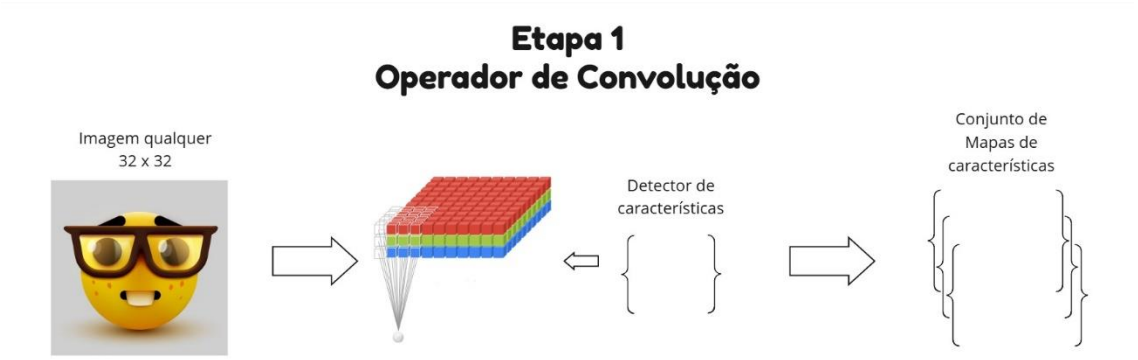
Operador de Convolução



A primeira etapa da rede convolucional é reduzir a complexidade de uma imagem. É possível visualizar que a imagem acima está no espectro RGB, normalmente é aplicada uma redução para a escala de cinza, que consegue preservar as características da imagem e reduzir a complexidade da rede como um todo.

Em seguida o algoritmo define os melhores valores para o detector de características, que é uma matriz, de acordo com a imagem passada. O tamanho da matriz também e definido pelo algoritmo, variando muito de tamanho conforme o comprimento e largura da imagem passada.

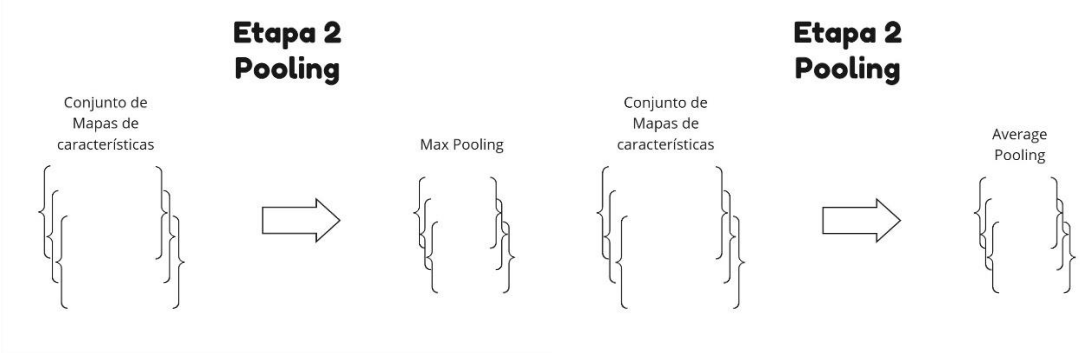
Como a imagem passada é do tipo RGB, o detector de características deverá atuar nos 3 espectros de cor da imagem. Além disso, a ideia desse detector é criar mapas de características que irão dizer se determinada característica foi encontrada na imagem, como um óculos. O foco não é dizer onde ocorreu, mas sim que ocorreu.



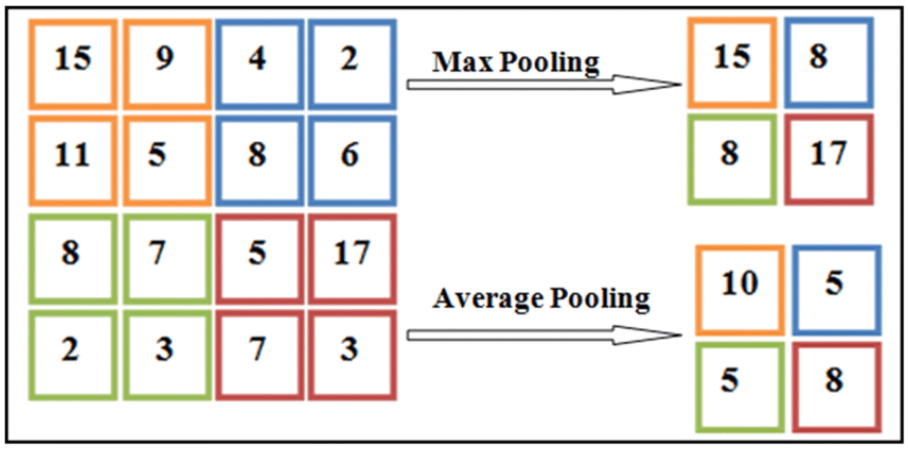
É interessante destacar também que existem vários tipos de matrizes para o detector de características, como uma matriz para deixar a imagem mais nítida, para adicionar Blur, remover o Blur, identificar bordas, dentre outras (https://en.wikipedia.org/wiki/Kernel_(image_processing)).

Por fim, após aplicar o detector de características na imagem, tem-se como resultado um conjunto de mapas de características, que possuem tamanho menor que a imagem original, e preservam as principais diferenças de uma imagem para outra.

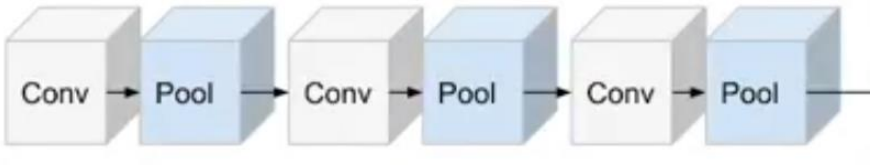
Pooling



Finalizada a etapa 1, os valores obtidos nos mapas de características são refinados mais uma vez, passando pela técnica Max Pooling, Average Pooling ou ambas, onde no Max pooling, dado uma seleção de valores μ x μ em um mapa de características, acessa o maior valor e armazena-o em uma matriz de Pooling. Já o Average Pooling faz a mesma seleção de valores, no entanto, soma os valores e os divide pelo número de valores.



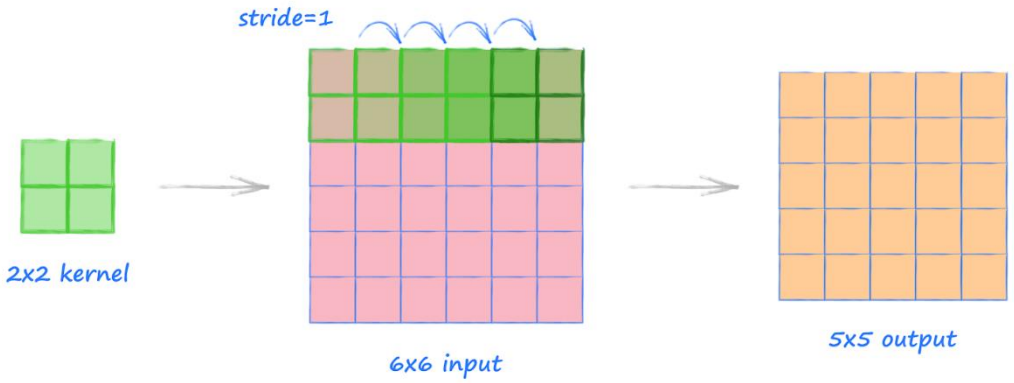
O tamanho do filtro de pooling definirá quanto a imagem reduzirá, por exemplo, se uma imagem 100x100 utilizar um filtro 2x2, a imagem será reduzida pela metade, resultando em 50x50.

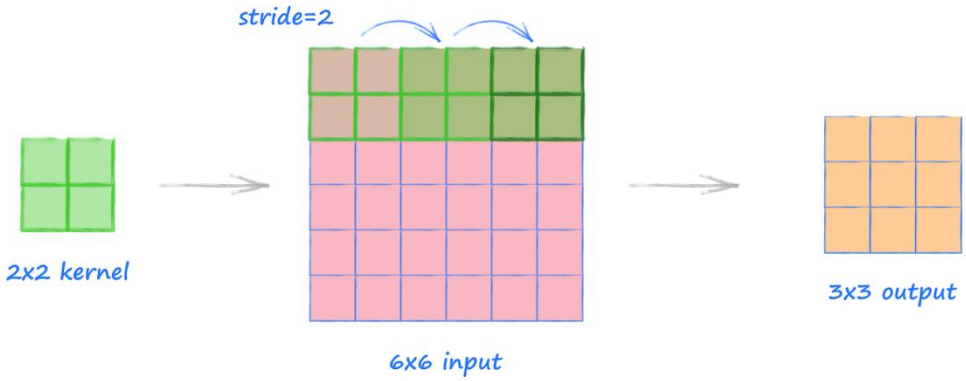


Uma sequência de uma camada de convolução, seguida de uma de pooling fará com que a imagem reduza significativamente de tamanho. A ideia é que ela chegue ao final com tamanho muito pequeno.

Alternativa ao Pooling - Stride

Ele funciona similarmente ao pooling que acabamos de discorrer, mas adiciona um parâmetro que indicará a movimentação do kernel sobre os dados de um mapa de características. Esse parâmetro permite que o kernel se sobreponha na multiplicação do mapa.

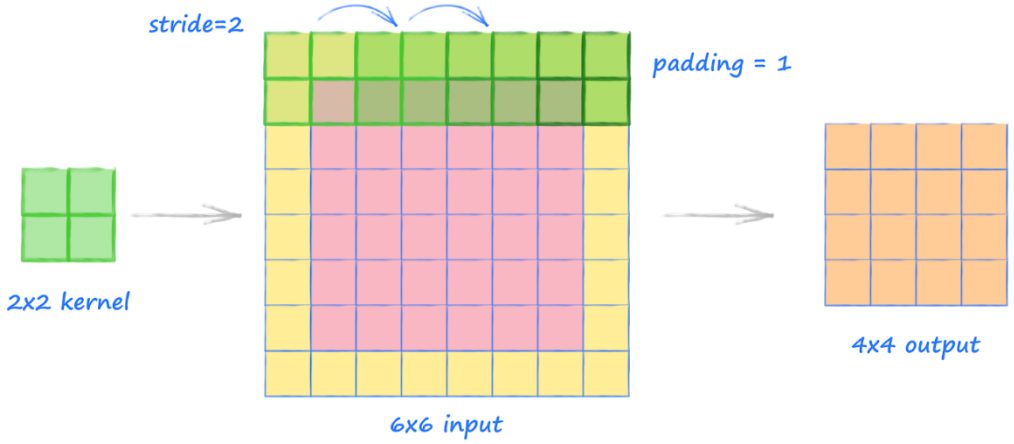




A ideia é tentar simplificar essa multiplicação de matrizes, por conta de que, em uma imagem, espera-se que se exista um pixel azul, é muito provável que os pixels anexos a esse também sejam da mesma cor. Por conta disso, o stride faz esses grandes saltos para fazer com que as características sejam extraídas de forma mais rápida e eficiente.

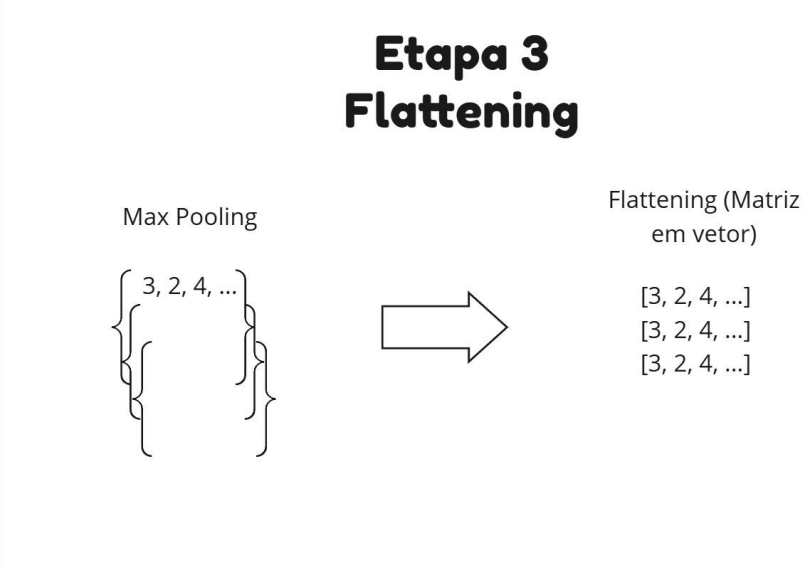
Pode-se também utilizar em conjunto com o stride, o padding, que fará com que a imagem seja estendida, adicionando zeros “ao redor” do mapa de características. É utilizado para extrair informações com maior precisão, reduzindo a perda de informações durante as etapas de convolução.

Existem tipos diferentes de padding, onde o tipo “full” fará com que sejam adicionados zeros suficientes para que todos os pixels sejam visitados a mesma quantia de vezes que o filtro (aumenta o output). Já o tipo “same” adiciona zeros suficientes para que o output tenha o mesmo tamanho que a imagem original passada. Por fim, o tipo “valid” não adiciona nenhum padding, reduzindo o tamanho da imagem (exemplos acima).



Essa alternativa pode se tornar até mais eficiente em alguns casos, mas claro, depende do que está sendo passado para a rede neural.

Flattening

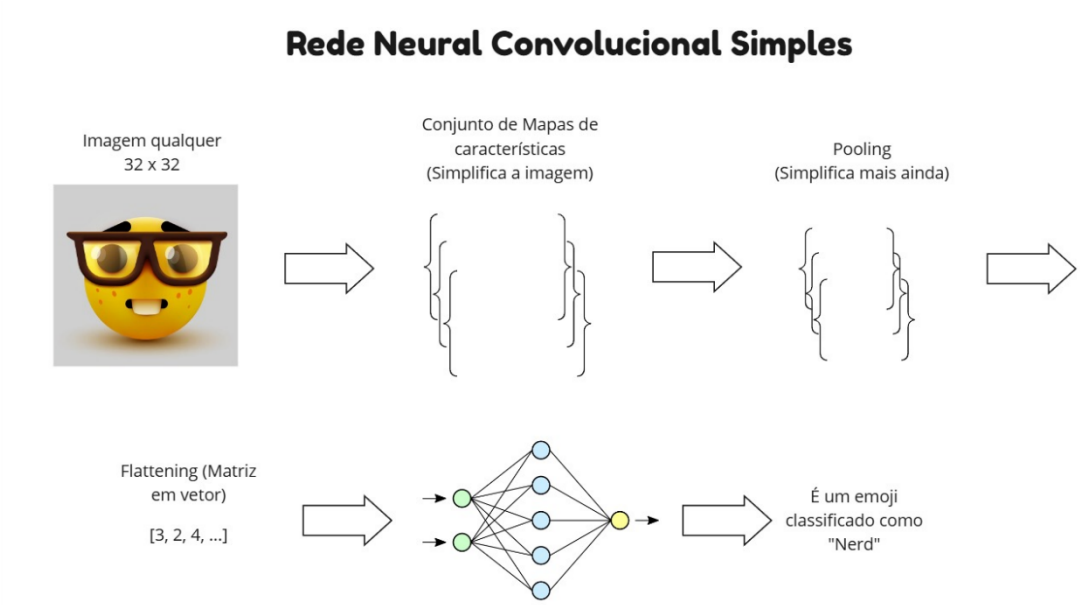


A última etapa da rede convolucional transforma as matrizes obtidas no processo anterior em um único vetor. Os valores desses vetores serão utilizados na camada de entrada da rede neural.

É importante salientar também que caso sejam inseridas imagens de tamanhos diferentes, a rede neural não irá funcionar, como por exemplo, criar uma rede neural

para imagens 480x480 e inserir imagens 1024x1024. Ao final das etapas de convolução, o vetor gerado será maior que o vetor esperado em uma imagem 480x480.

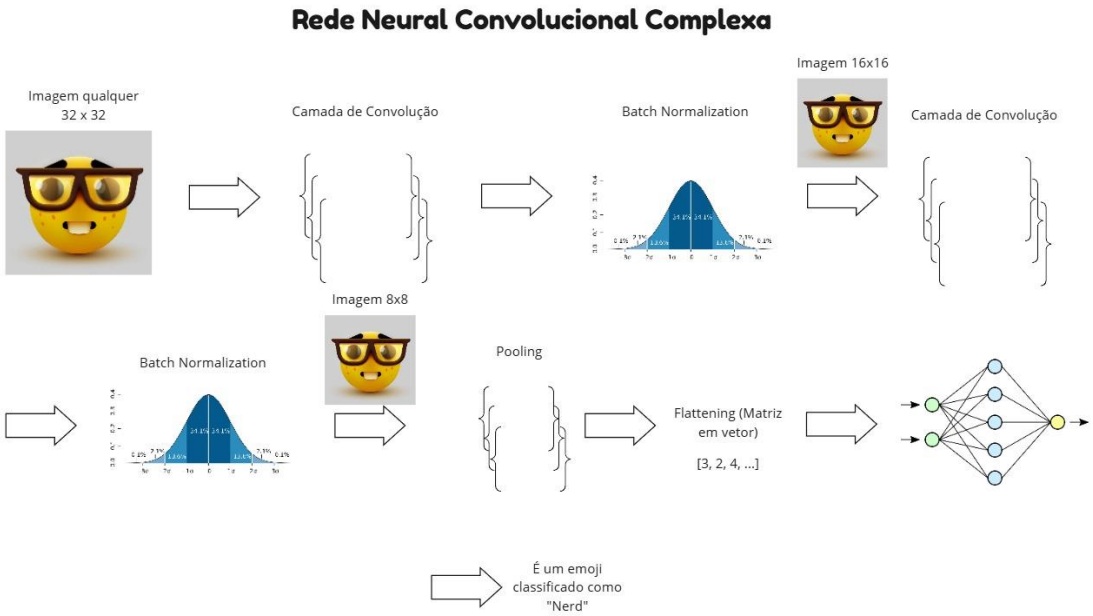
Rede neural simples em sua totalidade



Por fim, na imagem acima é mostrado todo o processo de uma rede neural convolucional simples.

1. Inserção de um conjunto de imagens;
2. Extração das características dessa imagem a partir do detector de características;
3. Aplicação do Max Pooling, a fim de manter as especificidades da imagem e torna-la menor ainda.
4. Planarização das matrizes;
5. Utilização dos vetores na camada de entrada da rede neural tradicional;
6. Obtenção do resultado final da rede.

Adicionando camadas na CNN simples



É possível verificar nessa CNN, que se adiciona camadas de normalização, chamadas de “Batch Normalization”. Essas camadas serão responsáveis por padronizar os dados, de acordo com uma distribuição normal. Ele faz isso acessando cada batch gerada, calculando a média e o desvio padrão e então fazendo a normalização baseada nesses dados.

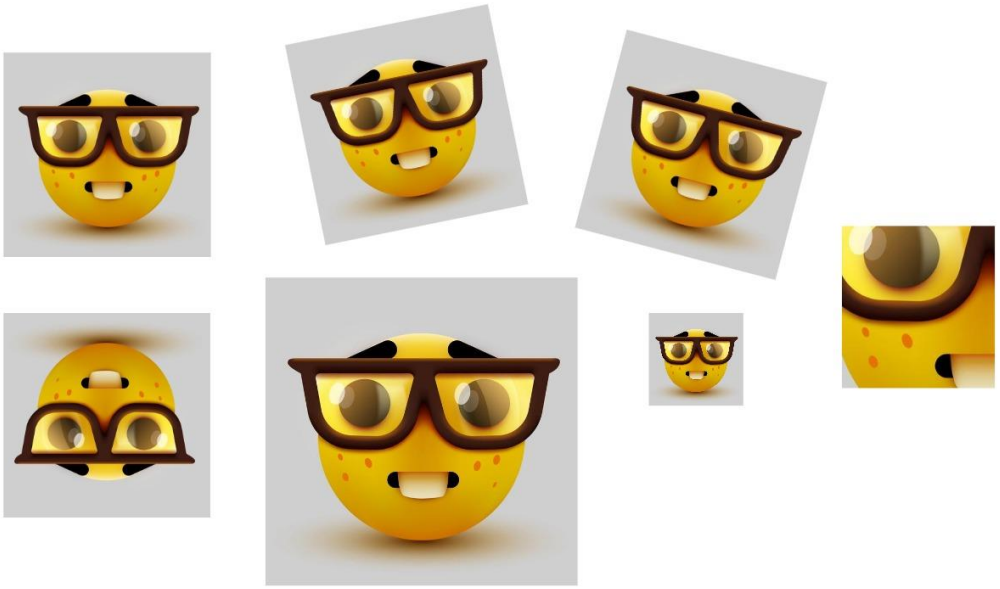
Devemos fazer esse tipo de padronização por conta de que, em alguns casos, adicionar esse tipo de camada pode evitar o acontecimento do overfitting.

Data Augmentation

O que poderíamos fazer em um caso em que não temos dados suficientes para gerar um modelo com boa acurácia?

A resposta disso é utilizar o data augmentation, que basicamente irá gerar, em tempo real de execução do código, imagens com pequenas mudanças definidas pelo usuário (essas imagens a serem geradas são baseadas no conjunto de imagens utilizada pelo usuário).

O usuário pode definir vários argumentos, como zoom, rotação, brilho, inverter a imagem tanto horizontalmente quanto verticalmente, dentre muitos outros. É importante ressaltar que quanto mais possibilidades de alterações são passadas para a função geradora, mais espaço e processamento é utilizado.



Utilizando CNN em processamento de linguagem natural

Nas aulas anteriores, foi explicado a utilização do One-Hot Encoding, que basicamente cria uma matriz de 0's e 1's, onde cada palavra existente na base de dados possuirá um id único (binário).

Color
Red
Red
Yellow
Green
Yellow

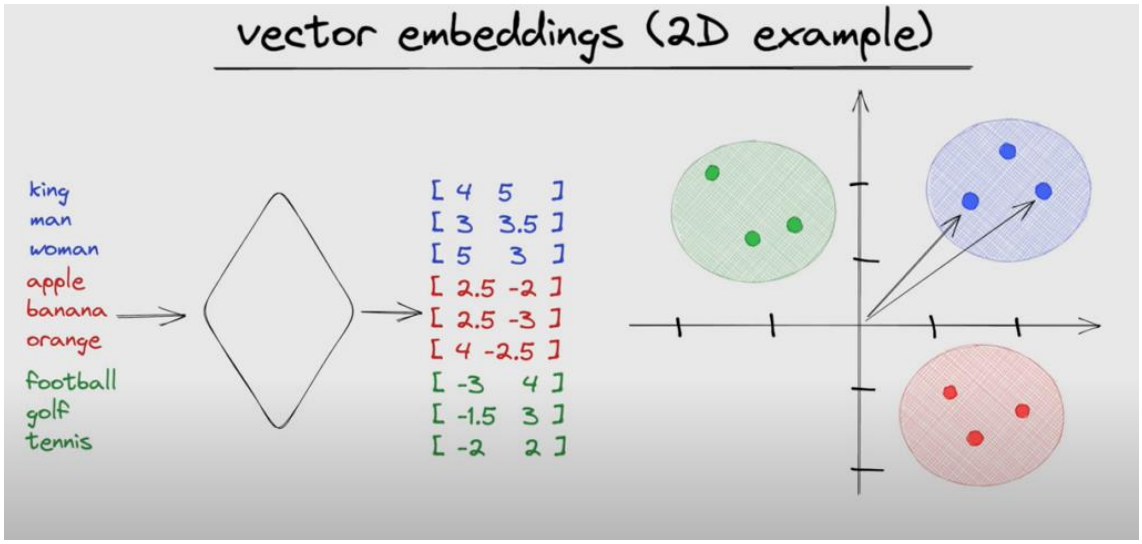
Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Essa abordagem funciona para dados não muito grandes. No entanto, à medida que utilizamos grandes conjuntos de dados (como o vocabulário inglês, que possui aproximadamente 200 mil palavras), acaba se tornando inviável utilizar esse método.

Além disso, o algoritmo utilizado nas CNNs não conseguirá encontrar uma relação geométrica entre os dados, visto que com essa abordagem, a distância entre quaisquer valores é a raiz quadrada de 2, por exemplo, [1,0,0] e [0,1,0].

Embedding

Esse método de transformação fará com que as sentenças sejam convertidas em sequências de mapeamento de palavras.



A primeira etapa é converter as palavras em valores inteiros. Por exemplo, “King”, “Man” e “Woman” serão convertidos em 25,11,8 respectivamente. Em seguida esses valores serão convertidos em uma sequência de vetores de inteiros. [25,11,8] → [[4,5], [3,3.5], [5,3]]. Os pesos são definidos automaticamente a partir da descida do gradiente enquanto o modelo é executado.

A proximidade desses valores indicará se eles são semelhantes uns aos outros ou não, permitindo uma relação geométrica.

Referencias

Deep Learning: Convolutional Neural Networks in Python (Seção 5 à 11)