Jupyter Notebook Execution Report

Name: Your Name

Date: December 12, 2025

Cell 1: ■ Markdown

Introdução

Tensorflow é uma biblioteca de código aberto do Google para machine learning e Deep Learning, permite criar e treinar modelos de IA, manipulando dados em tensores (array multidimensionais) e oferecendo ferramentas flexíveis para pesquisadores e desenvolvedores, sendo usada em serviços como Busca e Tradutor.

Comandos

Input

Representa a entrada da rede neural.

Serve para dizer qual é o formato dos dados que vão entrar no modelo.

- Define o tamanho da imagem

- É sempre o primeiro bloco da rede

- Não tem pesos só descreve o formato

Flatten

Transforma uma matriz (imagem) em um vetor único. Imagens têm formato (altura, largura, canais)

É necessário quando você vai conectar a imagem em camadas densas

Camada Dense

A camada **Dense** (totalmente conectada) conecta **todos os neurônios da camada atual a todos os neurônios da camada anterior**.

É uma camada fundamental em redes neurais, usada para aprender padrões complexos nos dados.

Fórmula

`saida = activation((W * x) + b)`

Onde:

- **activation** → função de ativação aplicada elemento a elemento

- **W * x** → multiplicação matricial entre entrada e kernel (pesos)

- **b** → vetor de bias

Exemplo

```python

densa = Dense(128, activation='relu')(x)

```

- Usada para classificação no final da rede

- Pode ter ReLU, Softmax, Sigmoid etc.

Características

- Usada normalmente **no final da rede** para classificação

- Aceita ativações como **ReLU**, **Softmax**, **Sigmoid**, etc.

--

Parâmetros

```python

dense = layers.Dense(

units = 128, # *obrigatório* Número de neurônios da camada

activation = 'relu' # *opcional* Função de ativação (`relu`, `sigmoid`, `softmax`, `tanh`, etc.).

use_bias=True, # *(default: True)* Indica se um vetor de bias será usado.

kernel_initializer="glorot_uniform", # Inicialização dos pesos (ex.: `glorot_uniform`, `he_normal`).

bias_initializer="zeros", # *(default: zeros)* Inicialização do bias.

kernel_regularizer=None, # Regularização aplicada aos pesos (`l1`, `l2`, etc.).

bias_regularizer=None, # Regularização aplicada ao bias.

activity_regularizer=None, # Regularização aplicada à saída da camada.

kernel_constraint=None, # Restrições aplicadas aos pesos (ex.: norma máxima).

bias_constraint=None # Restrições aplicadas ao bias (ex.: norma máxima).

)

```

Cell 2: ■ Markdown

Conv2D*

É a camada convolucional, usada para identificar padrões em imagens:

- Arestas

- Curvas

- Texturas

- Objetos

Ela aplica **filtros** (kernels) pela imagem

`conv = Conv2D(32, kernel_size=(3,3), activation='relu')(input)`

Onde,

- 32 filtros

- Kernels 3x3

- Cada filtro cria um mapa de características (feature map)

****Maxpooling****

Reduz o tamanho da imagem pegando apenas o valor máximo dentro de janelas

`pool = MaxPooling(pool_size=(2,2))(conv)`

Isso diminui a resolução, reduz parâmetros, retém as características mais importantes, ajuda a evitar overfitting

Referências para olhar depois

https://dingyan89.medium.com/calculating-parameters-of-convolutional-and-fully-connected-layers-with-keras-186590df36c6

Cell 3: ■ Markdown

Código

Cell 4: ■ Code

```
from tensorflow.keras import models, layers
```

```
model = models.Sequential([
```

```
])
```