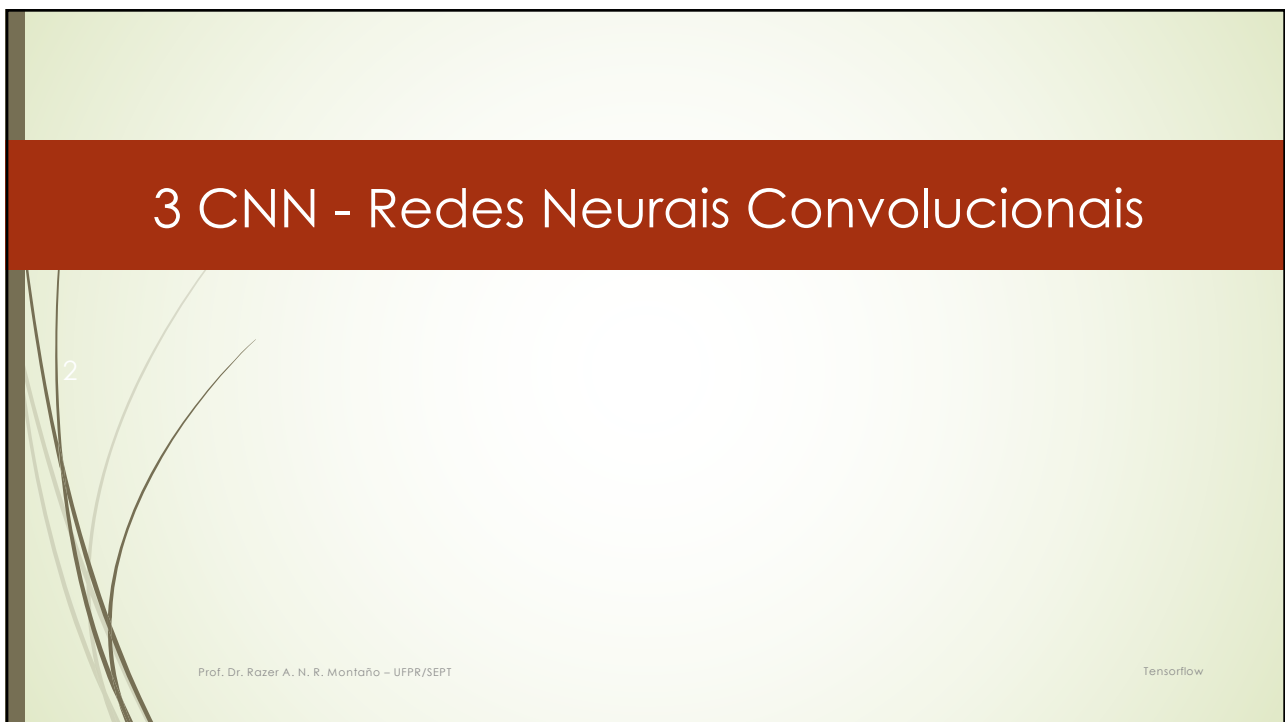




1

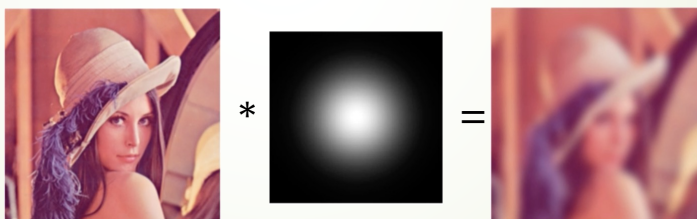


2

3

CNN: Convolução

- O que é **CNN**?
 - CNN é uma rede neural com a operação de Convolução
- O que é **Convolução**?
 - Aplicação de um Filtro/Kernel na imagem, gerando uma próxima



Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

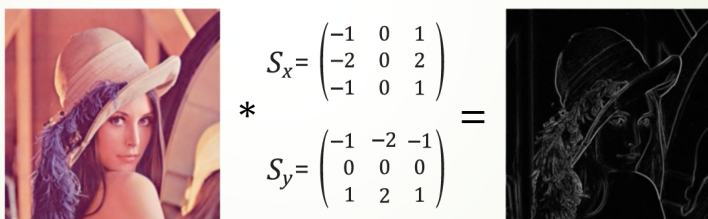
Tensorflow

3

4

CNN: Convolução

- Detecção de Bordas (Filtro Sobel)



Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

4

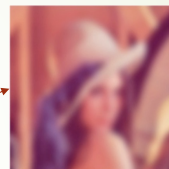
5

CNN: Convolução

- Convolução = Modificador da Imagem
 - O que muda é o Filtro/Kernel



Blurr



Sobel



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

5

6

CNN: Convolução

- Aplicação do Filtro
 - Varre-se o filtro na entrada: Multiplica-se os elementos entre a entrada e o filtro e soma-se o resultado final

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

6

7

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

$$\begin{matrix} 1*1 + 1*0 + 1*1 + \\ 0*0 + 1*1 + 1*0 + \\ 0*1 + 0*0 + 1*1 \end{matrix}$$

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

| | | |
|---|--|--|
| 4 | | |
| | | |
| | | |

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

7

8

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

$$\begin{matrix} 1*1 + 1*0 + 0*1 + \\ 1*0 + 1*1 + 1*0 + \\ 0*1 + 1*0 + 1*1 \end{matrix}$$

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

| | | |
|---|---|--|
| 4 | 3 | |
| | | |
| | | |

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

8

Prof. Dr. Razer A N R Montano

4

9

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | | |
|---|---|---|
| 4 | 3 | 4 |
| | | |
| | | |

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

9

10

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | | |
| | | |

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

10

Prof. Dr. Razer A N R Montano

5

11

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | |
| | | |

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

11

12

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| | | |

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

12

13

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| 2 | | |

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

13

14

CNN: Convolução

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| 2 | 3 | |

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

14

15

CNN: Convolução.

Aplicação do Filtro

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

 \ast

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

 $=$

| | | |
|---|---|---|
| 4 | 3 | 4 |
| 2 | 4 | 3 |
| 2 | 3 | 4 |

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

15

16

CNN: Padding

- Pode-se perceber que a dimensão da imagem resultante diminui
- Há o recurso de *padding*
 - É a adição de pixels para manter a dimensionalidade
- Três possibilidades
 - **Valid:** Não há *padding*. O movimento do filtro é limitado às bordas reais da imagem. O resultado é uma imagem menor. (exemplo apresentado anteriormente)
 - **Same:** Adiciona-se uma coluna e linha de pixels para aplicação do filtro. O resultado é uma imagem de tamanho igual à entrada.
 - **Full:** Adiciona-se tantas colunas/linhas com zeros quantas sejam necessárias para que cada pixel seja visitado a mesma quantidade de vezes pelo filtro. Aumenta o tamanho do resultado

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

16

17

CNN: Padding

Resultado

Filtro 3x3

Original

Valid

Sem padding

Diminui a saída

Same

Mantém o tamanho da saída

Full

Cada pixel é visitado o mesmo número de vezes

Aumenta a saída

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

17

18

CNN: Padding.

Dadas as seguintes dimensões:

Tamanho da Imagem (assumindo imagens quadradas): $n \times n$

Tamanho do Kernel (assumindo quadrados): $k \times k$

| Modo | Efeito na Saída | Tamanho da Saída | Uso |
|-------|-----------------|------------------|---------|
| Valid | Diminui | $n - k + 1$ | Típico |
| Same | Mantém | n | Típico |
| Full | Aumenta | $n + k + 1$ | Atípico |

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

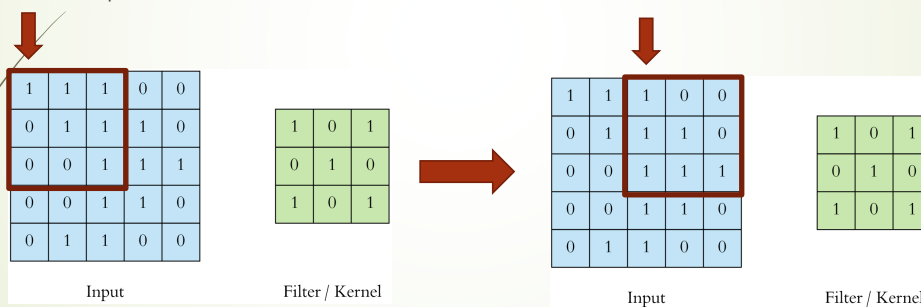
Tensorflow

18

19

CNN: Stride

- **Stride** é o passo dado na aplicação do filtro
 - Se for 1, funciona como esperado (no exemplo de padding)
 - Se for 2, por exemplo, o tamanho da imagem resultante diminui
- Exemplo, Stride = 2



Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

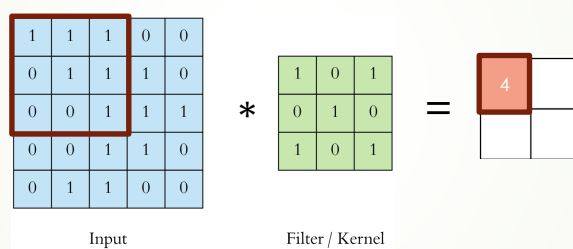
Tensorflow

19

20

CNN: Stride

- Stride = 2



Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

20

21

CNN: Stride

Stride = 2

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | |
|---|---|
| 4 | 4 |
| | |

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

21

22

CNN: Stride

Stride = 2

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | |
|---|---|
| 4 | 4 |
| 2 | |

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

22

23

CNN: Stride.

Stride = 2

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | |
|---|---|
| 4 | 4 |
| 2 | 4 |

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT


Tensorflow

23

24

CNN: Descoberta de Padrões

Entrada



Filtro



Saída



*

=

↑

COMO SABER QUAL FILTRO APLICAR????

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

24

25

CNN: Descoberta de Padrões.

- Qual filtro aplicar para se obter o resultado?
- O filtro fará parte da Rede:

$$Conv = \sigma(W * x + b)$$

- Onde
 - W é o filtro
 - x é a entrada
 - b é o bias (presente nas redes)
- DEIXAR A CNN DESCOBRIR ☺**
- Usar o mesmo mecanismo das Redes Neurais: **Descida de Gradiente**
 - Assim, para todas as entradas, a rede “aprende” o que é W (o(s) filtro(s)) que devem ser aplicados para se descobrir o padrão

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

25

26

CNN: Descoberta de Padrões

- Exemplo, a rede recebe várias imagens de pessoas e não pessoas

| Entrada | Filtro | |
|---|--------|--------------|
|  | * | ??? |
| | | = PESSOA |
|  | * | ??? |
| | | = NÃO PESSOA |

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

26

27

CNN: Descoberta de Padrões.


- Então ela descobre o filtro que diferencia PESSOA e NÃO PESSOA



Filtro

*  = DETECTADO: PESSOA



*  = NÃO DETECTADO:
NÃO PESSOA

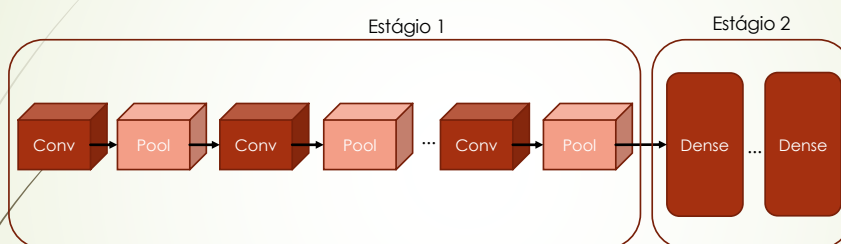
Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

27

28

Arquitetura das CNNs



Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

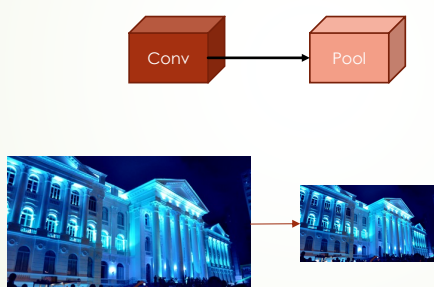
Tensorflow

28

29

Camada de Pooling

- Diminuir o tamanho da imagem
- Ex: 100x100 -> 50x50
- Geralmente aplicada após uma camada de Convolução



Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

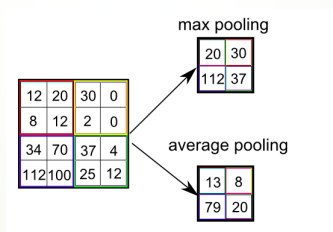
Tensorflow

29

30

Camada de Pooling

- Diminuição:
 - Como um filtro, p.ex, 2 x 2 com salto (*stride*) de 2
 - Diminui o tamanho pela metade
- Tipos de Pooling
 - **Max Pooling:** Obtém o maior valor
 - **Average Pooling:** Obtém o valor médio



Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

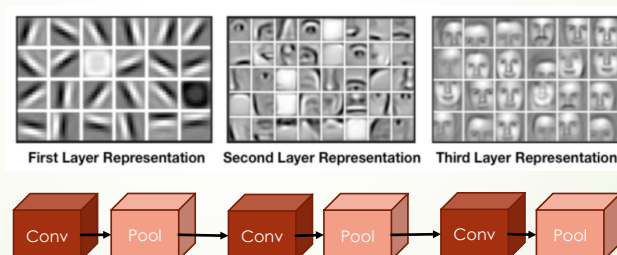
Tensorflow

30

31

Camadas Conv-Pooling

- Cada Camada acaba sendo responsável por identificar características mais ou menos refinadas
 - 1ª camada : reconhece traços e bordas
 - 2ª camada : reconhece características isoladas
 - 3ª camada : reconhece características compostas, como faces



Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

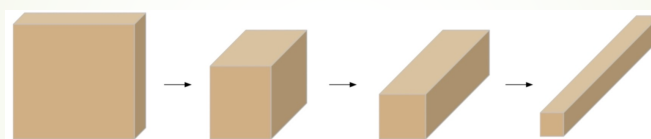
Tensorflow

31

32

Camadas Conv-Pooling

- Acaba perdendo informação?
 - $32 \times 32 \rightarrow 16 \times 16 \rightarrow 8 \times 8 \rightarrow 4 \times 4$



- **Sim**, perde a informação de **ONDE** está a característica
- Mas ganha a informação se a característica **ESTÁ PRESENTE**
 - Geralmente não importa onde a característica está, e sim se está presente

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

32

33

Alternativa ao Pooling

- Para não precisar de uma camada de Pooling, pode-se aplicar a Convolução com um passo (*stride*) maior que 1

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

$$\begin{matrix}
 1 & 0 & 1 \\
 0 & 1 & 0 \\
 1 & 0 & 1
 \end{matrix}
 \begin{matrix}
 * \\
 \\
 =
 \end{matrix}
 \begin{matrix}
 4 & \\
 & \\
 &
 \end{matrix}$$

Filter / Kernel

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

33

34

Alternativa ao Pooling

- Stride = 2

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

$$\begin{matrix}
 1 & 0 & 1 \\
 0 & 1 & 0 \\
 1 & 0 & 1
 \end{matrix}
 \begin{matrix}
 * \\
 \\
 =
 \end{matrix}
 \begin{matrix}
 4 & 4 \\
 &
 \end{matrix}$$

Filter / Kernel

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

34

35

Alternativa ao Pooling

Stride = 2

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | |
|---|---|
| 4 | 4 |
| 2 | |

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

35

36

Alternativa ao Pooling

Stride = 2

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Input

*

| | | |
|---|---|---|
| 1 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |

Filter / Kernel

=

| | |
|---|---|
| 4 | 4 |
| 2 | 4 |

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

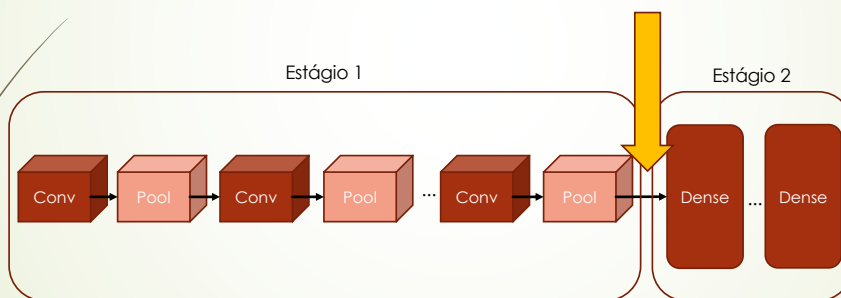
Tensorflow

36

37

Camada Densa

- Uma imagem $H \times W \times C$ é inadequada para entrar em uma RNA
 - Deve-se passar um vetor de uma dimensão (1-D)
- Pode-se "achatar" (*flatten*) a saída do Estágio 1



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

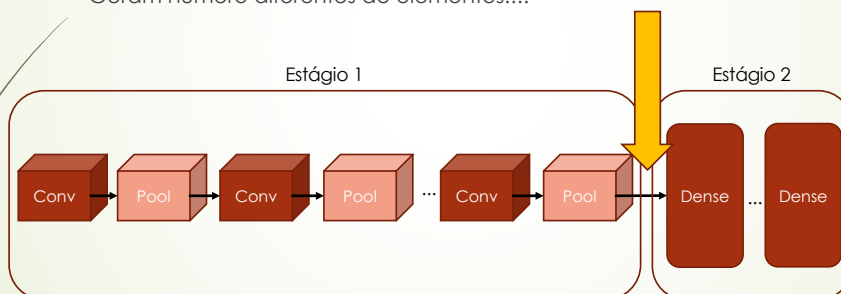
Tensorflow

37

38

Camada Densa

- O que acontece quando as imagens têm tamanhos diferentes ???
 - 64 x 64 ou 32 x 32
- 4 camadas de CONV-POOL, com mapa de características em 100:
 - Geram número diferentes de elementos!!!!



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

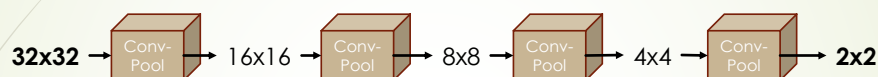
Tensorflow

38

39

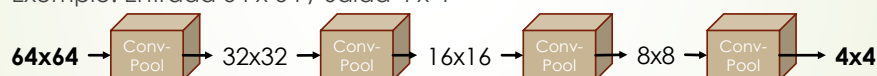
Camada Densa

- Exemplo: Entrada 32 x 32 / Saída 2 x 2



- Se aplicar **Flatten()**: 4 (2x2) elementos * 100 = 400 elementos

- Exemplo: Entrada 64 x 64 / Saída 4 x 4



- Se aplicar **Flatten()**: 16 (4x4) elementos * 100 = 1600 elementos

- A camada Densa deve receber o mesmo número de elementos, não pode ser variável

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

39

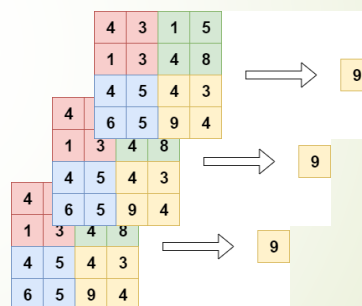
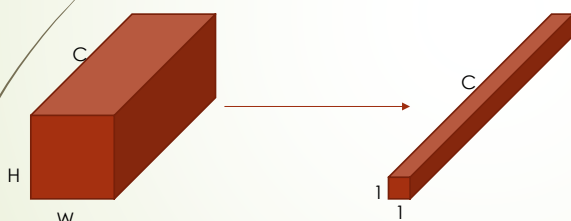
40

Global Max Pooling

- Global Max Pooling** sempre transforma a entrada em um vetor com a quantidade do mapa de características

- Aplica o pooling com o tamanho correto (que pode variar) para transformar

- H x W x C em 1 x 1 x C



Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

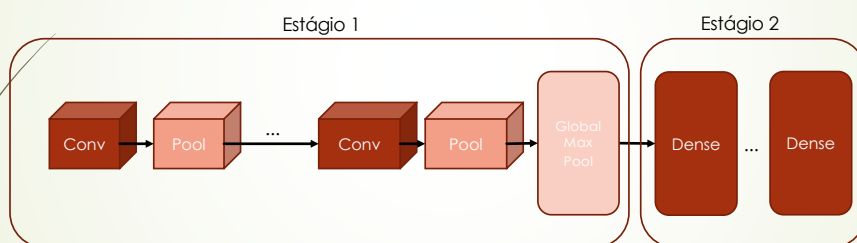
Tensorflow

40

41

Arquitetura Final.

- Camadas de **Convolução** → **Pooling** ou **Convolução com Stride maior**
- Camada de **Global Max Pooling** ou **Flatten**
- Camadas **Densas**



Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

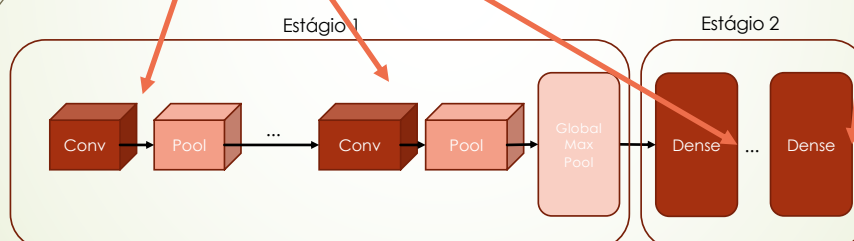
Tensorflow

41

42

Funções de Ativação

- Outras camadas
 - Em geral ReLU
- Na Camada de Saída:
 - **Regressão**: Nenhuma ou Linear
 - **Classificação – 2 classes**: Sigmoid
 - **Classificação – n classes**: Softmax



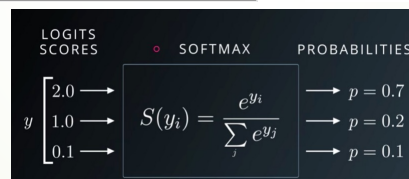
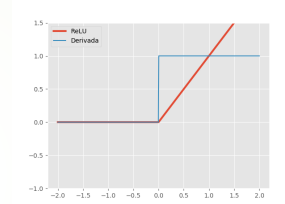
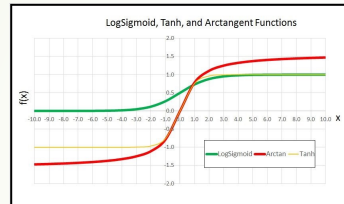
Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

42

43

Lembrete Sobre Funções de Ativação..



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

43

44

Exemplo de Rede: fashion_mnist

```
# número de classes
K = len(set(y_train))

# Camadas
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
x = Flatten()(x)
x = Dropout(0.2)(x)
x = Dense(512, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)

# Model ( lista entrada, lista saida)
model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

44

45

Exemplo de Rede: fashion_mnist

model.summary()

Model: "model_11"

| Layer (type) | Output Shape | Param # |
|-----------------------|---------------------|---------|
| input_12 (InputLayer) | [(None, 28, 28, 1)] | 0 |
| conv2d_33 (Conv2D) | (None, 13, 13, 32) | 320 |
| conv2d_34 (Conv2D) | (None, 6, 6, 64) | 18496 |
| conv2d_35 (Conv2D) | (None, 2, 2, 128) | 73856 |
| flatten_11 (Flatten) | (None, 512) | 0 |
| dropout_22 (Dropout) | (None, 512) | 0 |
| dense_22 (Dense) | (None, 512) | 262656 |
| dropout_23 (Dropout) | (None, 512) | 0 |
| dense_23 (Dense) | (None, 10) | 5130 |

Total params: 360,458
Trainable params: 360,458
Non-trainable params: 0

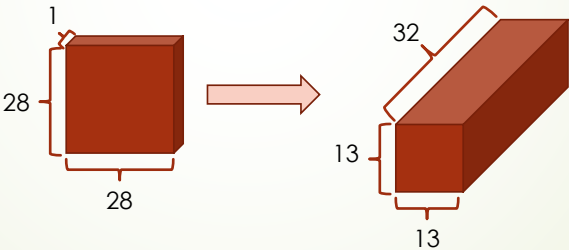
45

46

Exemplo de Rede: fashion_mnist

```
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
```

| Layer (type) | Output Shape | Param # |
|-----------------------|---------------------|---------|
| input_12 (InputLayer) | [(None, 28, 28, 1)] | 0 |
| conv2d_33 (Conv2D) | (None, 13, 13, 32) | 320 |



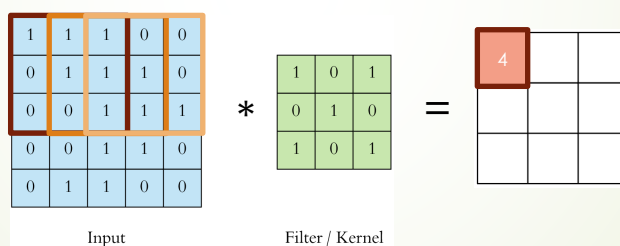
46

47

Exemplo de Rede: fashion_mnist

```
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
```

- Por que $28 \times 28 \times 1$ se torna $13 \times 13 \times 32$???
- Kernel 3×3 , com *stride* (passo) 1 diminui em 2



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

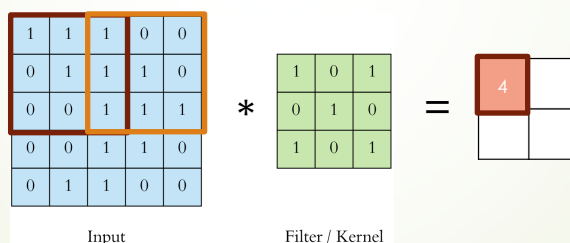
47

48

Exemplo de Rede: fashion_mnist

```
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
```

- Por que $28 \times 28 \times 1$ se torna $13 \times 13 \times 32$???
- $28 \times 28 \rightarrow$ [Filtro 3×3] $\rightarrow 26 \times 26$
- Stride* (passo) 2 dá a metade da dimensão
- $28 \times 28 \rightarrow$ [Filtro 3×3] $\rightarrow 26 \times 26 \rightarrow$ [Stride 2] $\rightarrow 13 \times 13$



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

48

49

Exemplo de Rede: fashion_mnist

```
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
```

- Por que 28 x 28 x 1 se torna 13 x 13 x 32 ???
- 32 é a quantidade de dimensões de saída

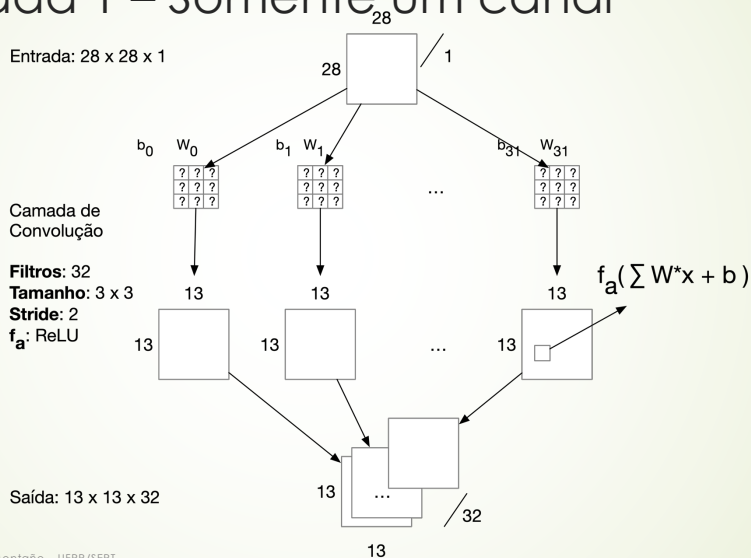
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

49

50

Camada 1 – Somente um canal



Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

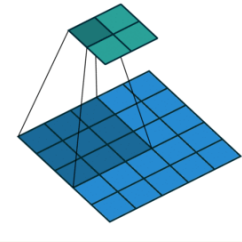
50

51

Camada 1 – Somente um canal

| | | | | |
|----------------|----------------|----------------|---|---|
| 3 ₀ | 3 ₁ | 2 ₂ | 1 | 0 |
| 0 ₂ | 0 ₂ | 1 ₀ | 3 | 1 |
| 3 ₀ | 1 ₁ | 2 ₂ | 2 | 3 |
| 2 | 0 | 0 | 2 | 2 |
| 2 | 0 | 0 | 0 | 1 |

| | | |
|------|------|------|
| 12.0 | 12.0 | 17.0 |
| 10.0 | 17.0 | 19.0 |
| 9.0 | 6.0 | 14.0 |



Fonte: <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f642faee1>

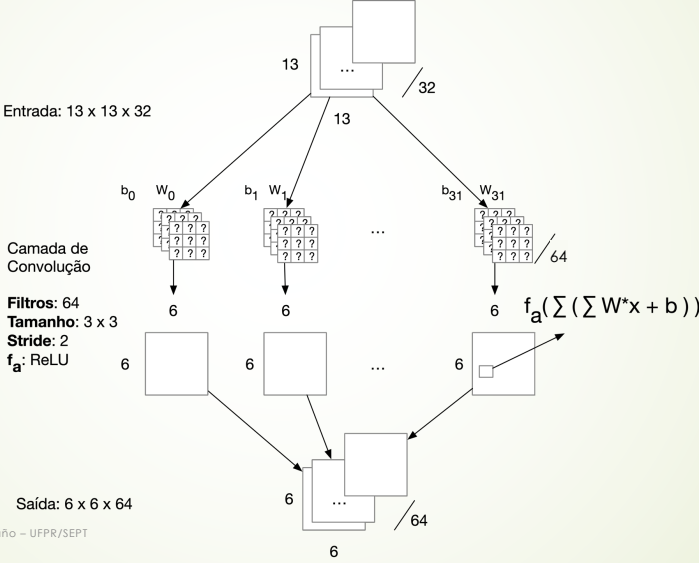
Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

51

52

Camada 2 – Vários canais



Entrada: 13 x 13 x 32

Camada de Convolução

Filtros: 64
Tamanho: 3 x 3
Stride: 2
 f_a : ReLU

Saída: 6 x 6 x 64

$f_a(\sum(\sum W \cdot x + b))$

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

52

53 Camada 2 – Vários canais

Convolução

Aggregação dos canais

Bias

Fonte: <https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42f9ee1>

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

53

54 Filtros x Kernel

- Um **Kernel**
 - Uma matriz aplicada na operação de convolução
 - Um kernel é único por canal
- Um **Filtro**
 - Um conjunto de kernels
 - Produce somente um canal

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

54

55

Exemplo de Rede: fashion_mnist

```
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
```

- Quantidade de parâmetros de uma camada:
 - Parâmetros treináveis
 - São os ajustes feitos na rede
- Fórmula para Camada de Convolução:
 - $\#param = \#canais\ saída * (\#canais\ entrada * altura\ do\ kernel * largura\ do\ kernel + 1)$
- Fórmula para Camada Densa:
 - $\#param = \#canais\ saída * (\#canais\ entrada + 1)$

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

55

56

Exemplo de Rede: fashion_mnist

```
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
```

- Na primeira camada:
 - $\#param = 32 * (1 * 3 * 3 + 1) = 320$

```
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
```

- Na segunda camada:
 - $\#param = 64 * (32 * 3 * 3 + 1) = 18.496$

```
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
```

- Na terceira camada:
 - $\#param = 128 * (64 * 3 * 3 + 1) = 73.856$

Prof. Dr. Razer A. N. R. Montano – UFPR/SEPT

Tensorflow

56

57

Exemplo de Rede: fashion_mnist

```
x = Flatten() (x)
x = Dropout(0.2) (x)
x = Dense(512, activation="relu") (x)
```

| | | |
|----------------------|-------------------|-------|
| conv2d_35 (Conv2D) | (None, 2, 2, 128) | 73856 |
| flatten_11 (Flatten) | (None, 512) | 0 |
| dropout_22 (Dropout) | (None, 512) | 0 |

Na camada densa:

Flatten gera 512 características

$\#param = 512 * (512 + 1) = 262.656$

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

57

58

Exemplo de Rede: fashion_mnist..

```
model.summary()
```

Model: "model_11"

| Layer (type) | Output Shape | Param # |
|-----------------------|---------------------|---------|
| input_12 (InputLayer) | [(None, 28, 28, 1)] | 0 |
| conv2d_33 (Conv2D) | (None, 13, 13, 32) | 320 |
| conv2d_34 (Conv2D) | (None, 6, 6, 64) | 18496 |
| conv2d_35 (Conv2D) | (None, 2, 2, 128) | 73856 |
| flatten_11 (Flatten) | (None, 512) | 0 |
| dropout_22 (Dropout) | (None, 512) | 0 |
| dense_22 (Dense) | (None, 512) | 262656 |
| dropout_23 (Dropout) | (None, 512) | 0 |
| dense_23 (Dense) | (None, 10) | 5130 |

Total params: 360,458
Trainable params: 360,458
Non-trainable params: 0

Total de Parâmetros Treináveis: 360.458

Prof. Dr. Razer A. N. R. Montañó – UFPR/SEPT

Tensorflow

58

59

LABORATÓRIO

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

59

60

Prática: Classificação de Imagens

■ Fashion MNIST

- Base de roupas
- Imagens em tons de cinza, 28 x 28
- 10 classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot
- Base está no formato: N x H x W
- Precisa transformar em: N x H x W x C

```
x_train.shape: (60000, 28, 28)
y_train.shape: (60000,)
x_test.shape: (10000, 28, 28)
y_test.shape: (10000,)
```

■ CIFAR10

- Base de imagens variadas
- Imagens coloridas, 32 x 32
- 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck
- Já está no formato correto: N x H x W x C

```
x_train.shape: (50000, 32, 32, 3)
y_train.shape: (50000,)
x_test.shape: (10000, 32, 32, 3)
y_test.shape: (10000,)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

60

61

Fashion MNIST

■ Importações

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.models import Model
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

61

62

Fashion MNIST

■ Carga da Base e Pré-processamento

```
# Carga dos dados
fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

# normalizar os dados
x_train, x_test = x_train / 255.0, x_test / 255.0
print("x_train.shape: ", x_train.shape)
print("x_test.shape: ", x_test.shape)

# Adicionar uma dimensão:
# N x H x W -> N x H x W x C
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train.shape: ", x_train.shape)
print("x_test.shape: ", x_test.shape)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

62

63

Fashion MNIST

Construir o Modelo

```
# número de classes
K = len(set(y_train))

# Camadas
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
x = Flatten()(x)
x = Dropout(0.2)(x)
x = Dense(512, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)

# Model ( lista entrada, lista saída)
model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

63

64

Fashion MNIST

Compilar e Treinar o Modelo

```
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])

r = model.fit(x_train, y_train, validation_data=(x_test, y_test),
             epochs=15)
```

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

64

65

Fashion MNIST

■ Avaliar o Modelo

```
# Plotar a função de perda
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()

# Plotar acurácia
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

65

66

Fashion MNIST

■ Efetuar Predições e Matriz de Confusão

```
# Efetuar predições
y_pred = model.predict(x_test).argmax(axis=1)

# Mostrar a matriz de confusão
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7), show_normed=True)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

66

67

Fashion MNIST.

- Analisar Algumas Predições Incorretas

```
labels = ["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal",  
"Shirt", "Sneaker", "Bag", "Ankle boot"]  
  
misclassified = np.where(y_pred != y_test)[0]  
  
i = np.random.choice(misclassified)  
  
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")  
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]],  
labels[y_pred[i]]))
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

67

68

EXERCÍCIO.

- Executar o exercício de reconhecimento de imagens do MNIST.

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

68

69

CIFAR10

■ Importações

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense,
Flatten, Dropout
from tensorflow.keras.models import Model
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

69

70

CIFAR10

■ Carga da Base e Pré-processamento

```
cifar10 = tf.keras.datasets.cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# normalizar os dados
x_train, x_test = x_train / 255.0, x_test / 255.0
y_train, y_test = y_train.flatten(), y_test.flatten()
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

70

71

CIFAR10

Construir o Modelo

```
# número de classes
K = len(set(y_train))

# Camadas
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
x = Flatten()(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)

# Model ( lista entrada, lista saída)
model = Model(i, x)
```

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

71

72

CIFAR10

Compilar e Treinar o Modelo

```
model.compile(optimizer="adam",
              loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])

r = model.fit(x_train, y_train, validation_data=(x_test, y_test),
             epochs=15)
```

Prof. Dr. Razer A. N. R. Montaña - UFPR/SEPT

Tensorflow

72

73

CIFAR10

■ Avaliar o Modelo

```
# Plotar a função de perda
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()

# Plotar acurácia
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
```

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

73

74

CIFAR10

■ Efetuar Predições e Matriz de Confusão

```
# Efetuar predições
y_pred = model.predict(x_test).argmax(axis=1)

# Mostrar a matriz de confusão
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7), show_normed=True)
```

Prof. Dr. Razer A. N. R. Montão – UFPR/SEPT

Tensorflow

74

75

CIFAR10.

- Analisar Algumas Predições Incorretas

```
# Mostrar algumas classificações erradas
labels = ["airplane", "automobile", "bird", "cat", "deer", "dog",
         "frog", "horse", "ship", "truck"]

misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)

plt.imshow(x_test[i], cmap="gray")
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]], labels[y_
pred[i]]))
```

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

75

76

EXERCÍCIO..

- Executar o exercício de reconhecimento de imagens do CIFAR10.

Prof. Dr. Razer A. N. R. Montaña – UFPR/SEPT

Tensorflow

76