

Redes de “Deep Learning” Exercícios

1. Indique se as seguintes afirmações são verdadeiras ou falsas e **justifique, indicando um exemplo concreto**:
- a) O número de parâmetros de uma rede CNN é bastante superior a uma rede MLP (densa) com o mesmo número de camadas e para classificação de imagens com a mesma dimensão.
 - b) Sem a aplicação de um operador de “padding” a dimensão de um “feature map” iguala sempre a dimensão da imagem de entrada.
 - c) A aplicação do operador “Full padding” gera um “feature map” de menor dimensão que a imagem de entrada.
 - d) Um valor de “stride” de 2 gera “features maps” com o dobro dos parâmetros.
 - e) Uma rede CNN não é treinada por um algoritmo do tipo gradiente, pois os filtros e seus parâmetros são determinados apenas com base em conhecimento pericial.

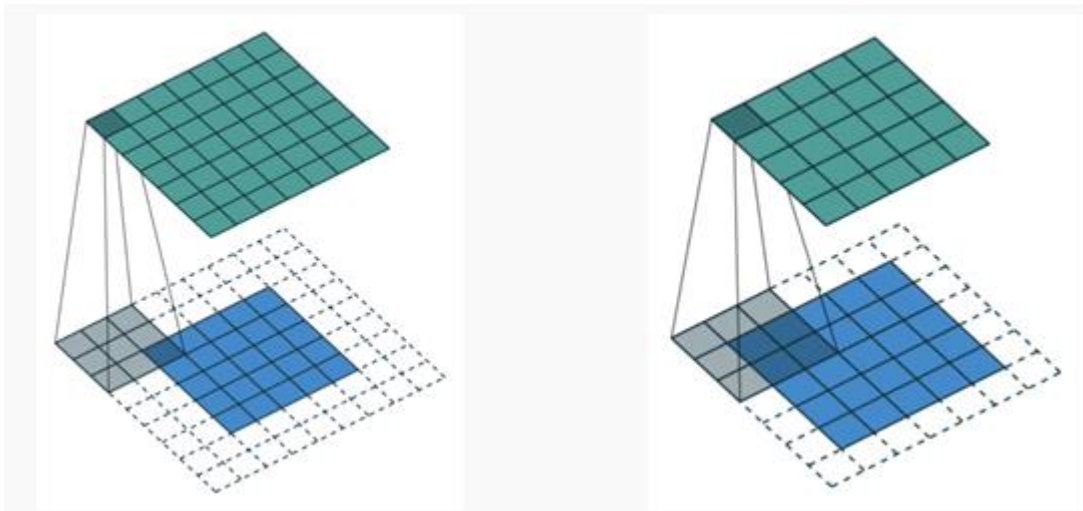


Figura 1. Full padding (esquerda) vs Same padding (direita) [1] : “Full padding” – Aumenta a dimensão da saída (feature map) relativamente à entrada (imagem), para garantir que todos os pixels sejam “visitados” o mesmo número de vezes ; “Same padding” – garante que o feature map resultante tenha a mesma dimensão do mapa de entrada;

2. Existem normalmente quatro operações numa rede CNN: operação de convolução, aplicação de uma função de ativação não linear (usualmente a função “Relu”), operação de “pooling” e finalmente o processo de classificação com uma rede densa “full connected”.

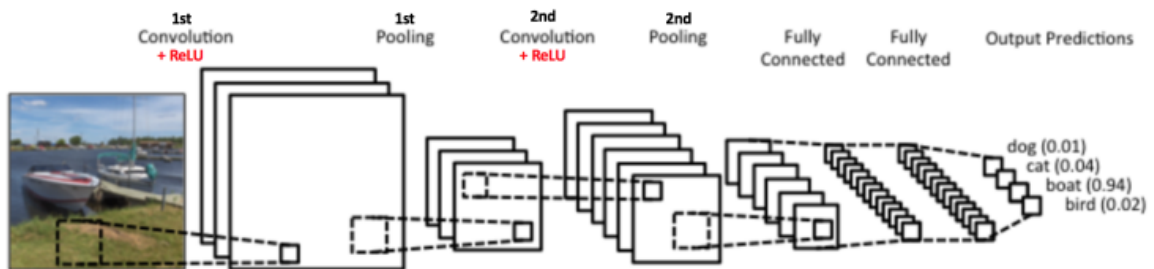


Figura 2. Exemplo de uma rede CNN [1]

Relativamente ao processo de convolução, considere uma imagem binária de 5x5:

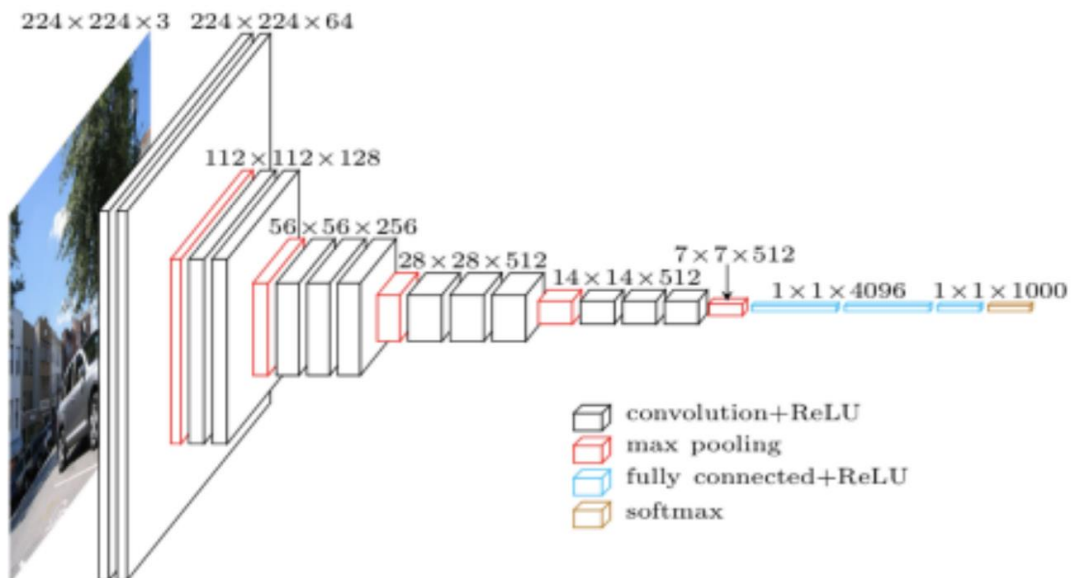
| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |

Considere a aplicação do seguinte filtro 3x3:

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

- a) Determine o “feature map” resultante do processo de convolução considerando:
- Stride = 1 e sem “padding”
 - Stride = 1 e “Same padding”.
 - Stride = 1 e “Full padding”.

- b) Aplique a função “Relu” ao feature map criado anteriormente
- com “bias”=0.
 - Com “bias”=2.
- c) Aplique o operador “Max Pooling” usando uma janela de 2x2 ao feature map resultante da alínea a).
- d) Aplique o operador “Average Pooling” usando uma janela de 2x2 ao feature map resultante da alínea a).
3. Relativamente à arquitetura da rede da Figura 4, responda às seguintes questões:
- Qual a dimensão da imagem e número de canais?
 - Quantas camadas de convolução existem?
 - Quantos filtros existem por camada? Comente o critério para definição do número de filtros nas camadas sucessivas.
 - Qual o operador de pooling e dimensão da janela?
 - Qual a arquitetura e funções de ativação da rede de classificação?



(a)

A rede tem a seguinte estrutura:

```
model = Sequential()  
model.add(Conv2D(input_shape=(224,224,3),filters=64,kernel_size=(3,3),padding="same",  
activation="relu"))
```

```
model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))

model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())
model.add(Dense(units=4096, activation="relu"))
model.add(Dense(units=4096, activation="relu"))
model.add(Dense(units=1000, activation="softmax"))
```

Figura 3. Rede VGG 16 para ImageNet 1000 classes [2]

4. Pretende-se a classificação de imagens de dimensão 150×100 RGB (3 canais). Considere uma rede CNN com 200 filtros 5×5 , com stride de 1 e ‘same padding’ (cria feature maps de 150×100).
- Calcule o número de parâmetros para a primeira camada de convolução.
 - Compare com o número de parâmetros de uma camada de uma rede densa “full connected” com o mesmo número de neurónios (15000) na primeira camada.

Soluções:

P2

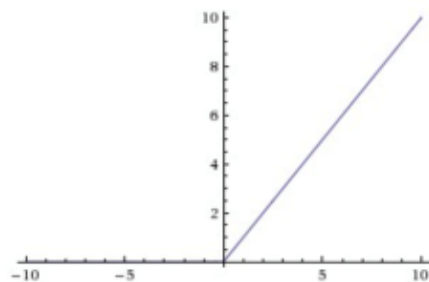
- a) A convolução de uma uma imagem 5x5 com um filtro 3x3, sem padding e com stride=1, gera uma mapa de 3x3. O primeiro elemento é calculado por:

$$1*1+1*0+1*1 + 0*0+1*1+1*0 + 0*1+0*0+1*1 = 4, \dots$$

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

- b) Relu com bias=0

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



- c) MaxPooling 2x2 (slide=1):

O primeiro elemento corresponde $\max(4,3,2,4), \dots$ e assim sucessivamente, obtemos:

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

P3

- a) Qual a dimensão da imagem e número de canais?
224x224 e 3 canais
- b) Quantas camadas de convolução existem?
 $2+2+3+3+3=13$
- c) Quantos filtros existem por camada? Comente o critério para definição do número de filtros nas camadas sucessivas.
64,64,128,128,256,256,512,512,512, 512,512,512
3 blocos de convolução. O número de filtros aumenta de forma a representar padrões mais complexos.
- d) Qual o operador de pooling e dimensão da janela?
Maxpooling, com janela de 2x2 e stride=2
- e) Qual a arquitetura e funções de ativação da rede de classificação?
Número de entradas = 25088; Duas camadas densas de 4096 neurónios e função "relu";
1000 saídas e função de ativação softmax.

P4

- a) Com filtros 5x5 produzindo 200 features maps, o número de parâmetros = $(5 \times 5 \times 3 + 1) \times 200$
(com bias) = **15 200**
- b) Para uma rede densa:
Número de entradas = $150 \times 100 \times 3$
Número de neurónios = 150×100 .
Para uma "fully connected layer", o número total de parâmetros sem bias = $(150 \times 100 \times 3) \times (150 \times 100) = \mathbf{675\ 000\ 000!!}$

Referências:

- [1] <https://www.mathworks.com/help/deeplearning/ref/vgg16.html>
- [2] <https://keras.io/api/applications/vgg/>