

MÓDULO 3

# Redes Neurais Convolucionais

**Bruno Légora Souza da Silva**

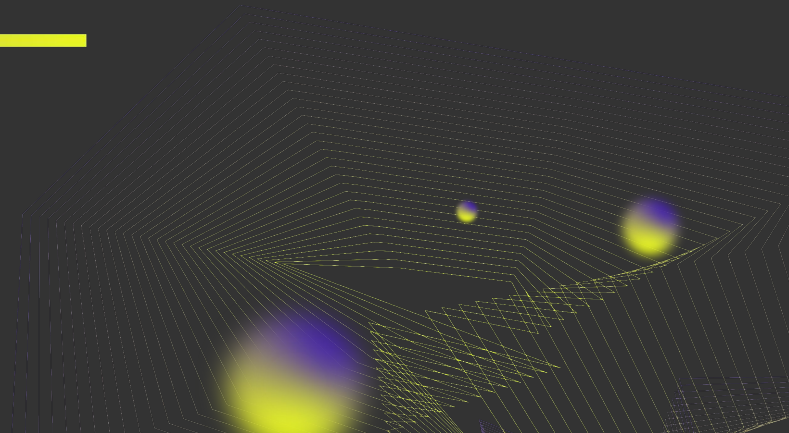
Professor do Departamento de Informática/UFES

ESPECIALIZAÇÃO

INTELIGÊNCIA ARTIFICIAL  
& CIÊNCIA DE DADOS

SEAD  
UFES

Superintendência de  
Educação a Distância



# ÍNDICE



1. Imagens e Redes Neurais
2. Redes Neurais Convolucionais
3. Parâmetros da CNN e Pooling
4. Pytorch e Redes Neurais  
Convolucionais
5. Laboratório 06

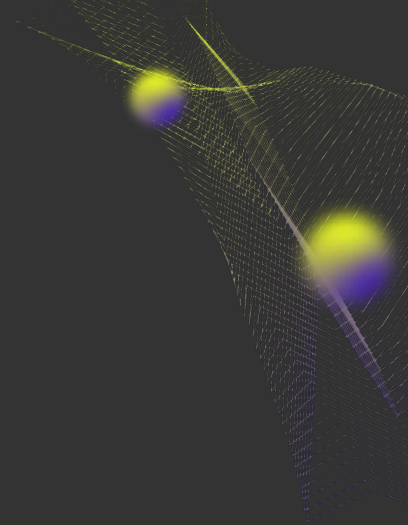


# 1. Imagens e Redes Neurais



# Imagens e Redes Neurais

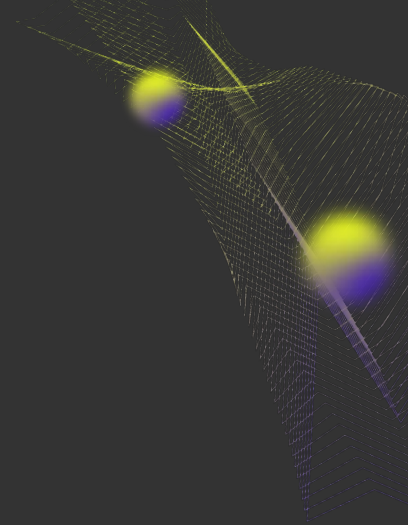
- No Laboratório 5, trabalhamos com imagens 64x64 em tons de cinza - que resultaram em uma entrada “achatada” com 1024 elementos
  - Em algum lugar da rede neural, o algoritmo lida com uma matriz 1024 linhas e/ou colunas;





# Imagens e Redes Neurais

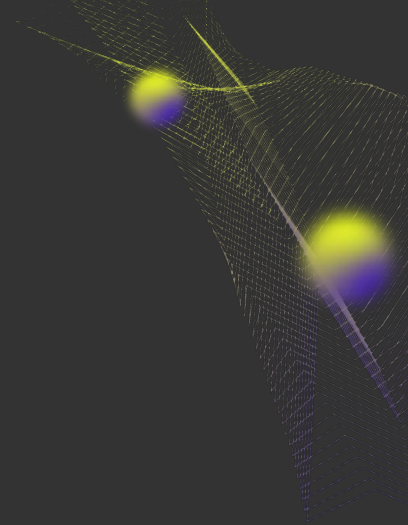
- É comum encontrarmos imagens com alta dimensão. Ex: uma imagem em tons de cinza com resolução 640x480 “achatada” gera um vetor de 307 mil elementos;
- Quase 1 milhão se for RGB;



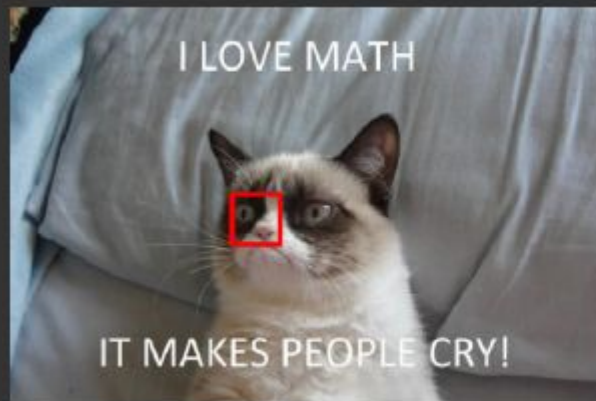


# Imagens e Redes Neurais

- Hoje em dia lidamos com imagens FullHD - 1920 x 1080 pixels, que são muito maiores que as citadas no slide anterior...
- Usá-las nas RNAs vistas é um problema grave...



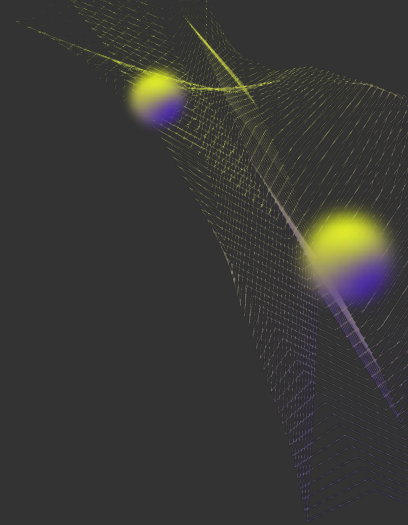
# Imagens e Redes Neurais





# Imagens e Redes Neurais

- Além disso, vimos que imagens possuem uma característica espacial que se perde ao “achatar” a imagem:
  - Pixels vizinhos geralmente possuem alguma similaridade





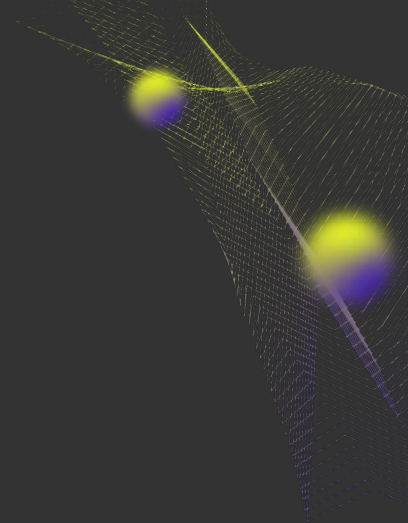
# Imagens e Redes Neurais





# Imagens e Redes Neurais

- Outra característica de imagens é que pequenos deslocamentos **não alteram** as características da imagem - enquanto vetores “achatados”  $[1, 0]$  e  $[0, 1]$  podem significar coisas completamente diferentes;



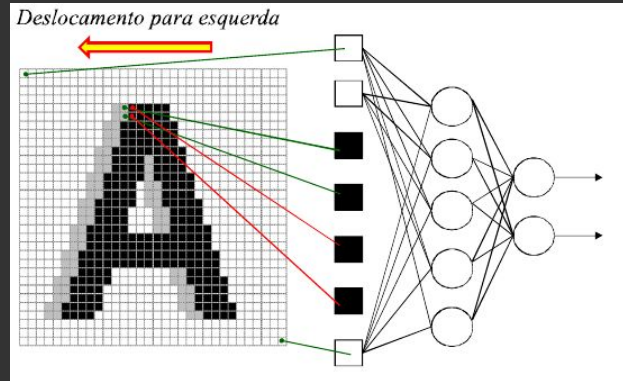
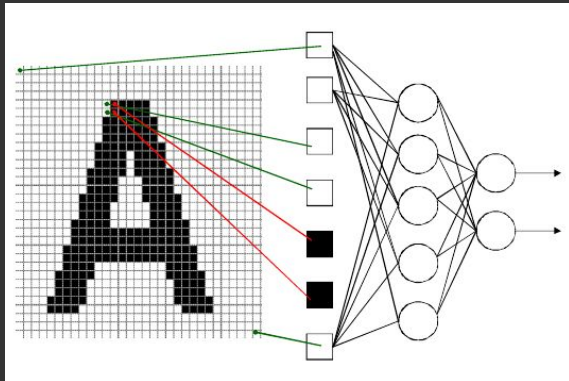
# Imagens e Redes Neurais

- Deslocamento de poucos pixels a esquerda:



# Imagens e Redes Neurais

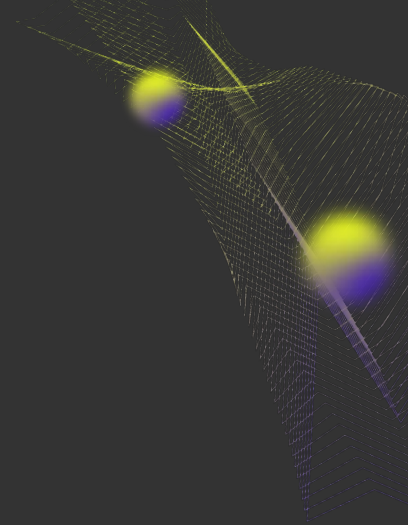
- Deslocamento de poucos pixels à esquerda:





# Imagens e Redes Neurais

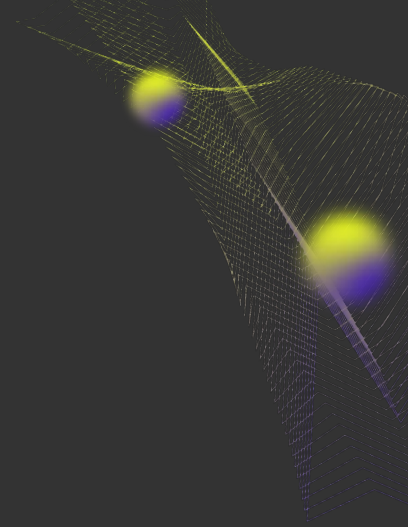
- As características apresentadas mostram que RNAs não são muito boas para serem usadas em imagens “cruas” – apenas com a utilização de extratores de características (HOG, LBP, etc)
- Mas como poderíamos usar RNAs com imagens?





# Imagens e Redes Neurais

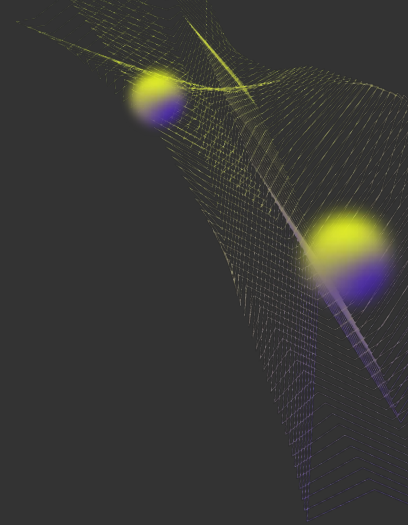
- Aprendemos uma operação básica na “Aula 3 - Processamento Básico de Imagens” que era capaz de detectar informações importantes de uma imagem (como as bordas, por exemplo)





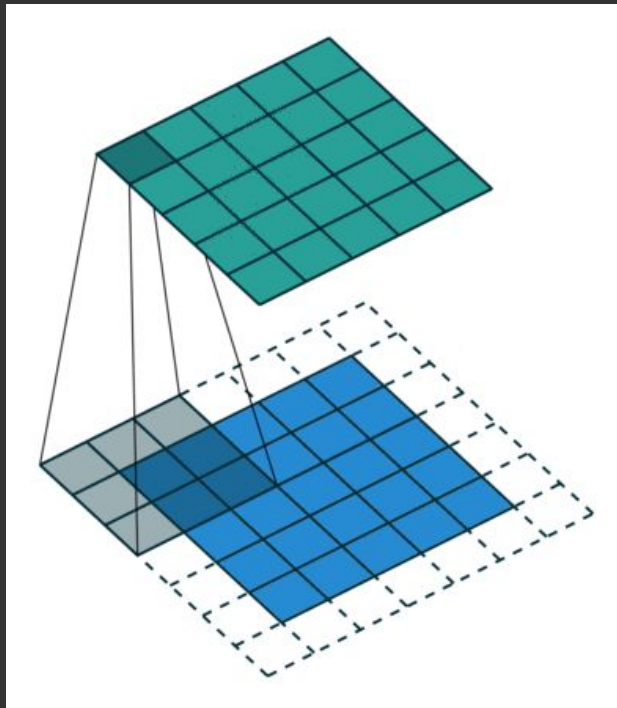
# Imagens e Redes Neurais

- Esses procedimentos são conhecidos como filtros lineares e a operação matemática principal a ser usada é a de convolução
  - Opera diretamente sobre os pixels



# Imagens e Redes Neurais

- Convolução 2-D:

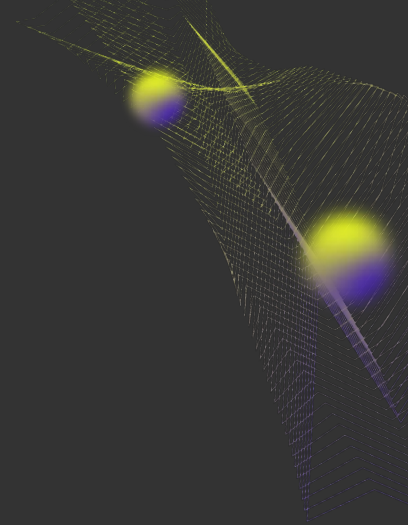






# Imagens e Redes Neurais

- Veremos, na próxima seção, as Redes Neurais Convolucionais
  - São redes neurais que usam a operação de convolução!



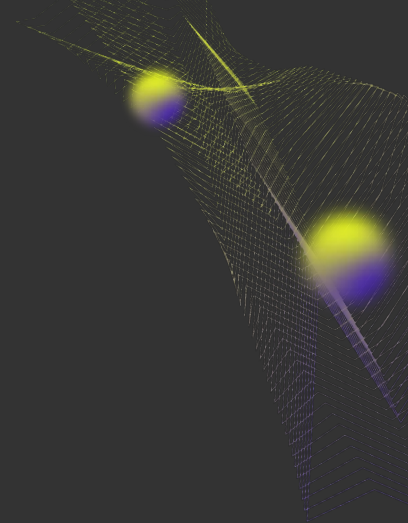


## 2. Redes Neurais Convolucionais

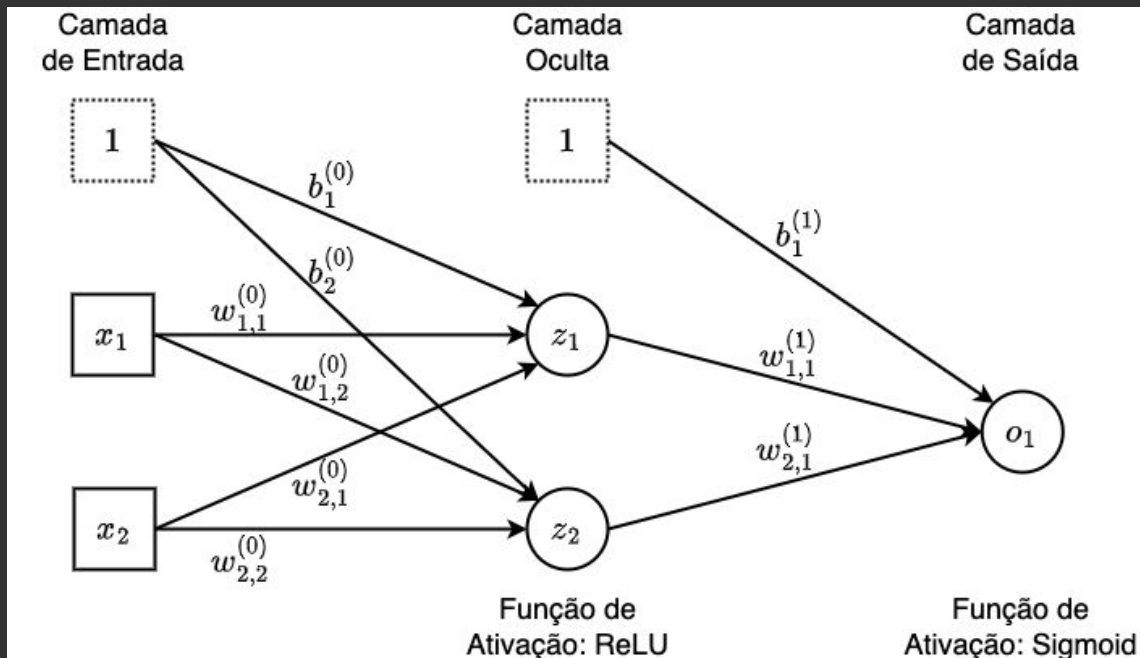


# Redes Neurais Convolucionais

- Na disciplina de Redes Neurais, vocês viram as MLPs:



# Redes Neurais Convolucionais



# Redes Neurais Convolucionais

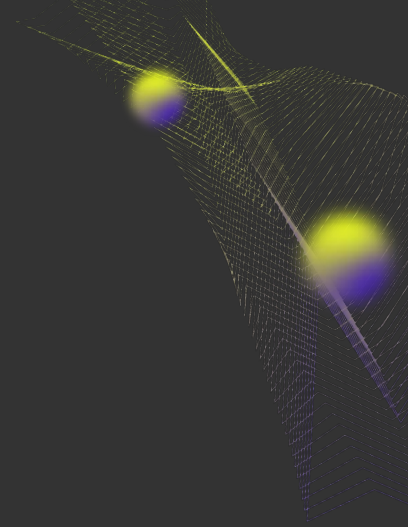
- A saída da camada de uma MLP é dada por:

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} f \left( w_{1,1}^{(0)} x_1 + w_{2,1}^{(0)} x_2 + b_1^{(0)} \right) \\ f \left( w_{1,2}^{(0)} x_1 + w_{2,2}^{(0)} x_2 + b_2^{(0)} \right) \end{bmatrix} = f(\overline{\mathbf{W}} \overline{x}) = \overline{z}$$



# Redes Neurais Convolucionais

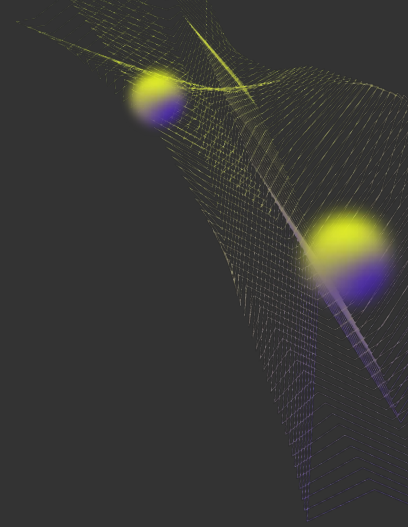
- Dizemos que esse tipo de camada de rede neural é “totalmente conectada”
  - a saída  $z_1$  depende de todas as saídas da camada anterior ( $x_1$  e  $x_2$ )
- Vetorialmente, dizemos que a saída  $z$  é dada pelo produto interno entre  $W$  e  $x$  (estendidos pelos *biases*);





# Redes Neurais Convolucionais

- Para cada neurônio em uma camada, deve existir um conjunto de parâmetros (um para cada neurônio da próxima camada) da rede neural naquela camada - voltamos ao problema das imagens grandes;

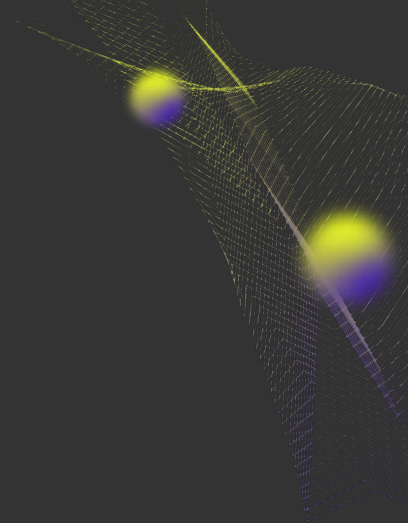




# Redes Neurais Convolucionais

- Já a camada convolucional substitui a operação por uma operação de convolução entre o filtro convolucional  $\mathbf{W}$  e a matriz  $\mathbf{X}$

$$\mathbf{z} = f(\overline{\mathbf{W}} * \overline{\mathbf{x}})$$

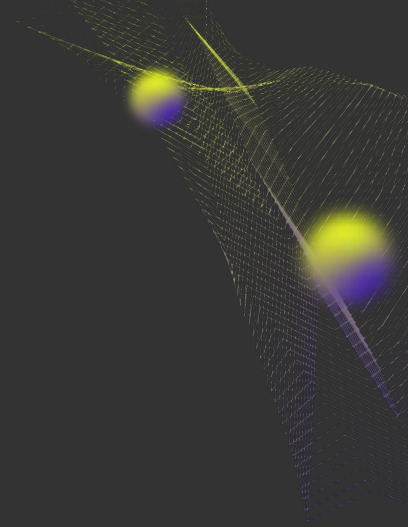




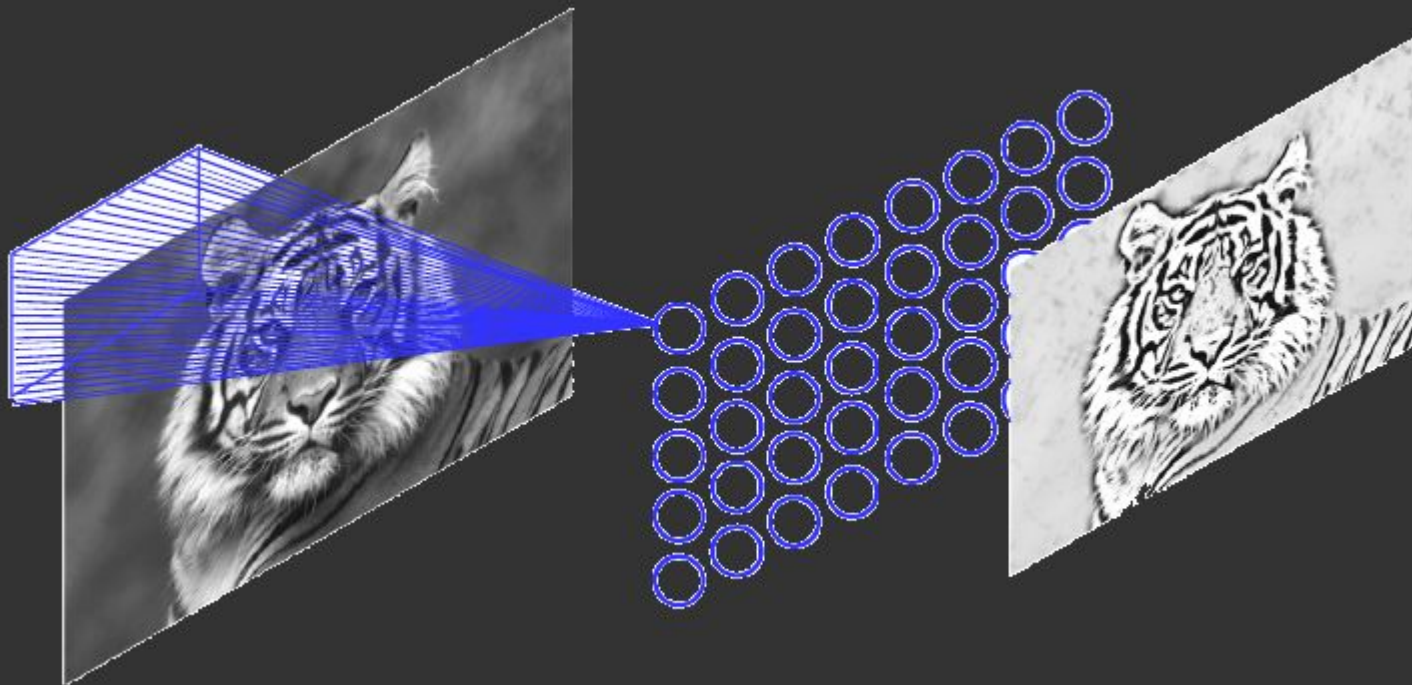


# Redes Neurais Convolucionais

- O filtro, agora, tem tamanho único e fixo (não depende mais do tamanho da matriz  $X$ ) - e a saída é outra matriz.
- A função de ativação permanece.
- Mas o que seria a matriz  $W$ ?



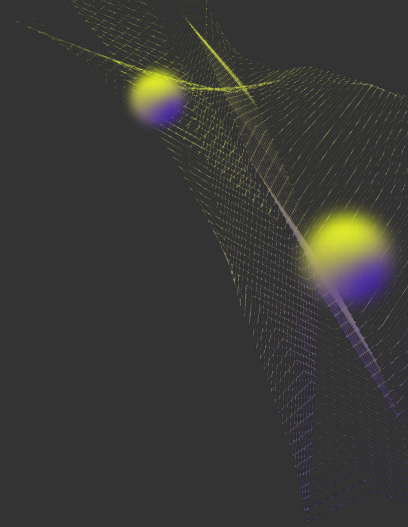
# Redes Neurais Convolucionais





# Redes Neurais Convolucionais

- Na literatura de PDI/VC podemos encontrar vários filtros criados **manualmente** – Canny, Sobel, Gaussiano, etc
  - Todos são boas opções para detectar características fundamentais em imagens





# Redes Neurais Convolucionais

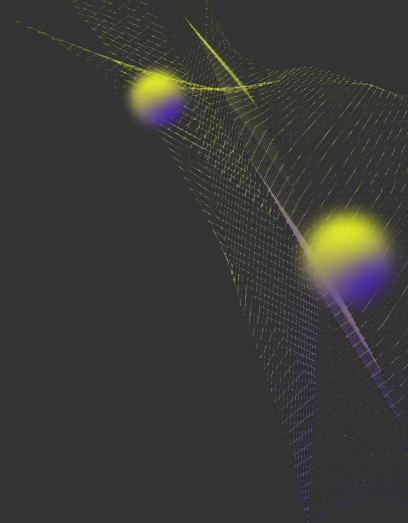


- Na literatura de PDI/VC podemos encontrar vários filtros criados **manualmente** – Canny, Sobel, Gaussiano, etc
  - Mas não necessariamente para tarefas de mais “alto nível”, como reconhecer objetos;



# Redes Neurais Convolucionais

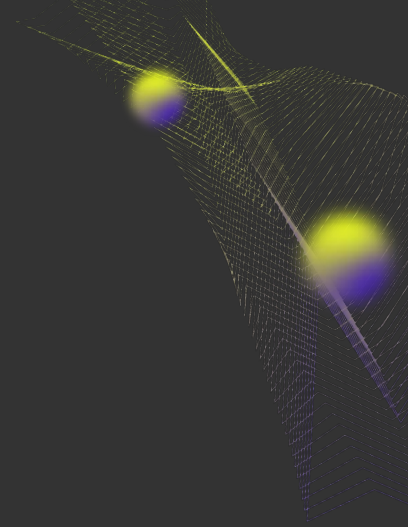
- Na Aula 5, vimos que as técnicas de DL aprendem as operações com os dados
  - Então, nossa RNC deverá aprender um filtro convolucional em seu processo de treinamento (backprop), assim como a RNA “padrão” aprende seus pesos;





# Redes Neurais Convolucionais

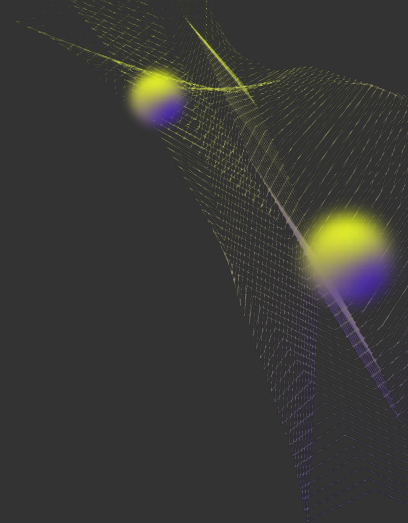
- Numa CNN, os pesos dos filtros são compartilhados por toda a imagem, o que não acontecia em camadas totalmente conectadas (cada entrada tinha um peso distinto)





# Redes Neurais Convolucionais

- Isso reduz, e muito, o custo computacional da rede neural;
- Vamos a uma comparação:



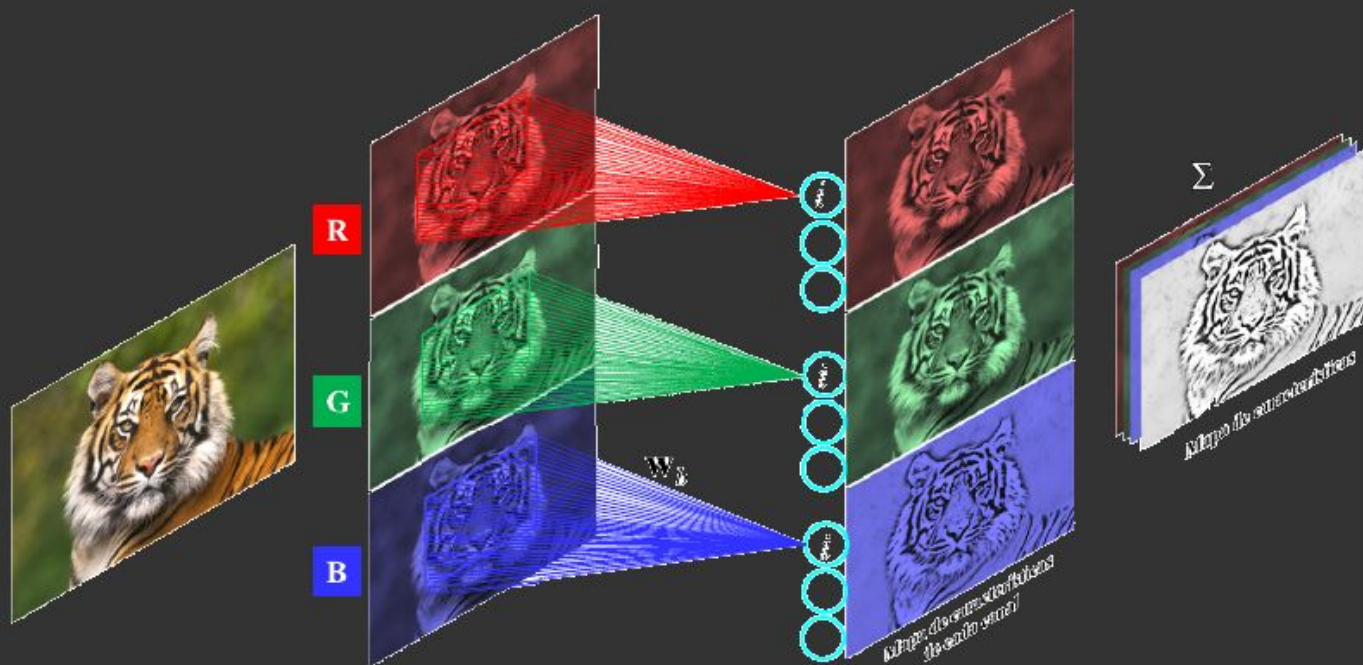
# Redes Neurais Convolucionais

| Tamanho da Imagem | Tipo e Tamanho da Camada | Número de parâmetros                              |
|-------------------|--------------------------|---|
| 64x64             | TC - 100 neurônios       | $64 \cdot 64 \cdot 100 = 409.600$                 |
| 64x64             | Conv - Filtro 11x11      | $11 \cdot 11 = 121$                               |
| 640x480           | TC - 100 neurônios       | $640 \cdot 480 \cdot 100 = 30.720.000$            |
| 640x480           | Conv - Filtro 11x11      | $11 \cdot 11 = 121$                               |
| 1920x1080x3       | TC - 100 neurônios       | $1920 \cdot 1080 \cdot 3 \cdot 100 = 622.080.000$ |
| 1920x1080x3*      | Conv - Filtro 11x11x3*   | $11 \cdot 11 \cdot 3 = 363^*$                     |

Obs: Em geral, são usados 1 matriz por canal de cor em imagens RGB



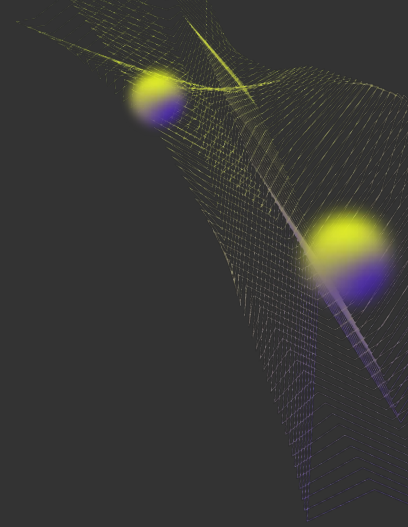
# Redes Neurais Convolucionais





# Redes Neurais Convolucionais

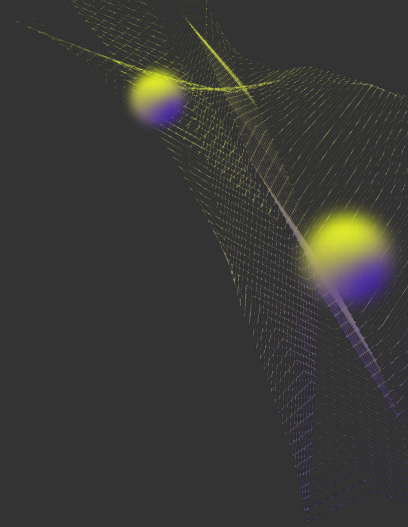
- Devido ao reduzido número de parâmetros que uma camada convolucional necessita, é comum usarmos múltiplas convoluções na mesma camada!
- O resultado é um tensor com múltiplos canais...





# Redes Neurais Convolucionais

- Ex:
  - Entrada - imagem tons de cinza
  - 10 filtros 11x11
  - Cada um resulta em uma matriz 2D
  - Resultado: Imagem com 10 canais

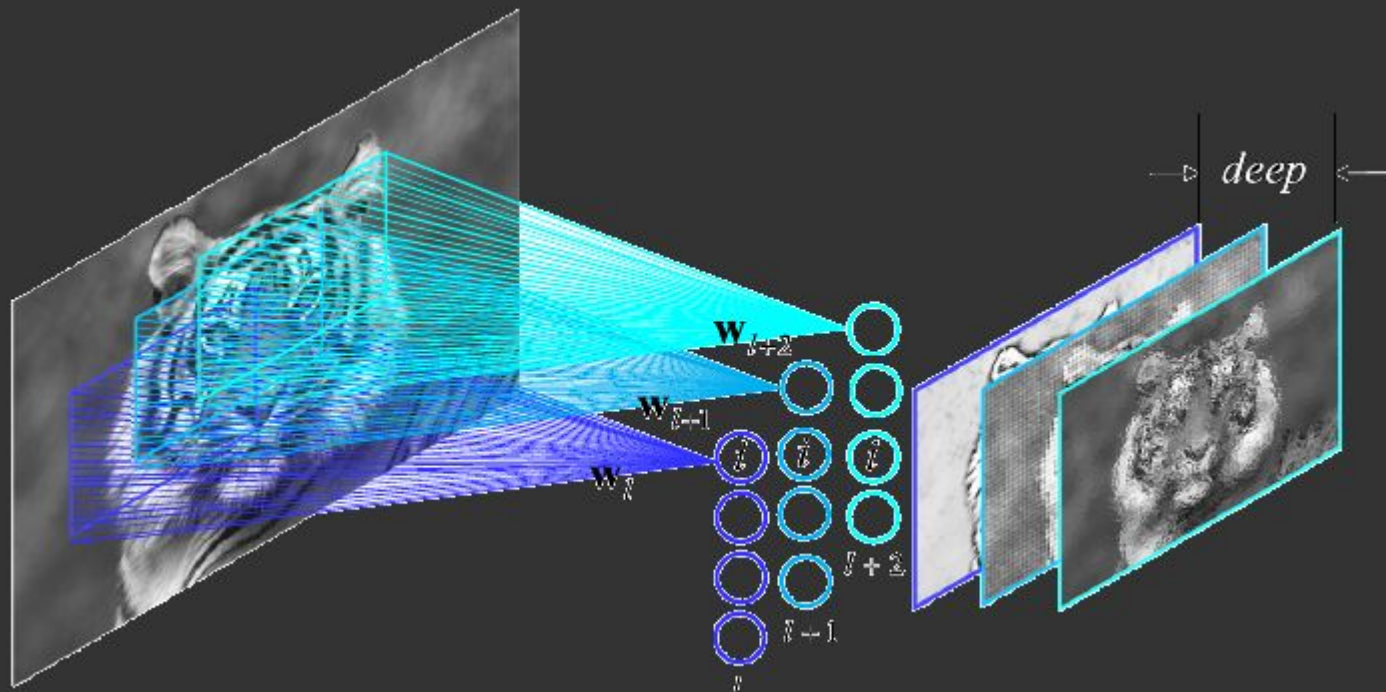




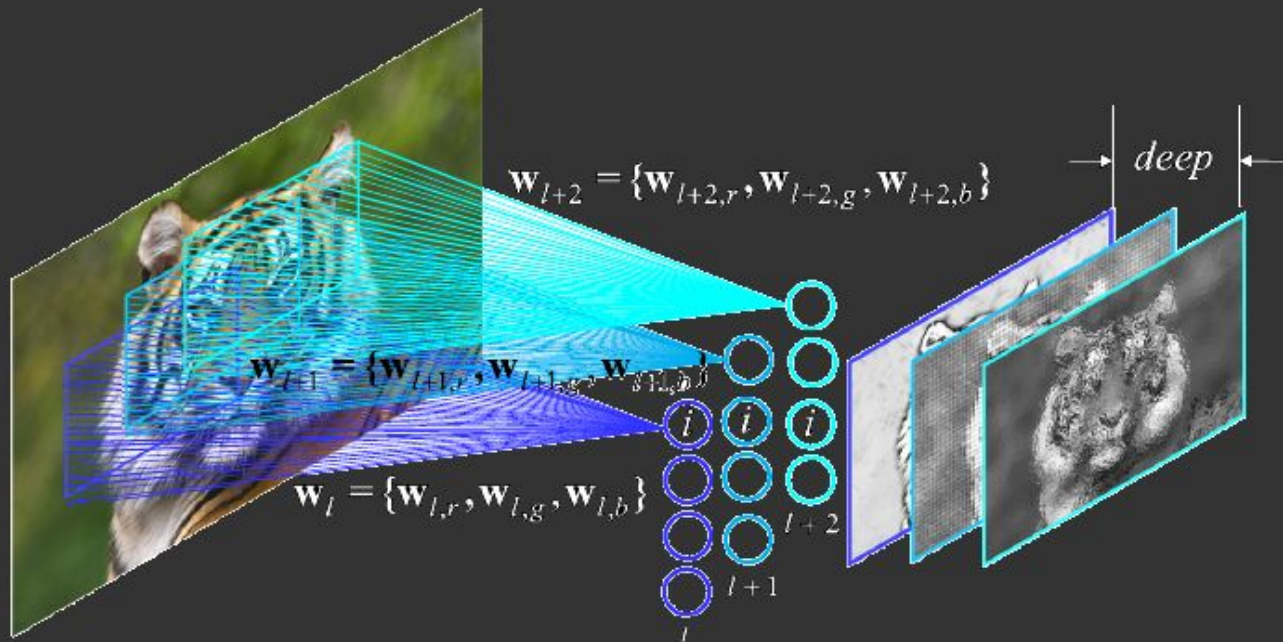
# Redes Neurais Convolucionais

- Ex:
  - Entrada - imagem RGB (3 canais)
  - 10 filtros 11x11x3
  - Cada um resulta em uma matriz 2D
  - Resultado: Imagem com 10 canais

# Redes Neurais Convolucionais



# Redes Neurais Convolucionais





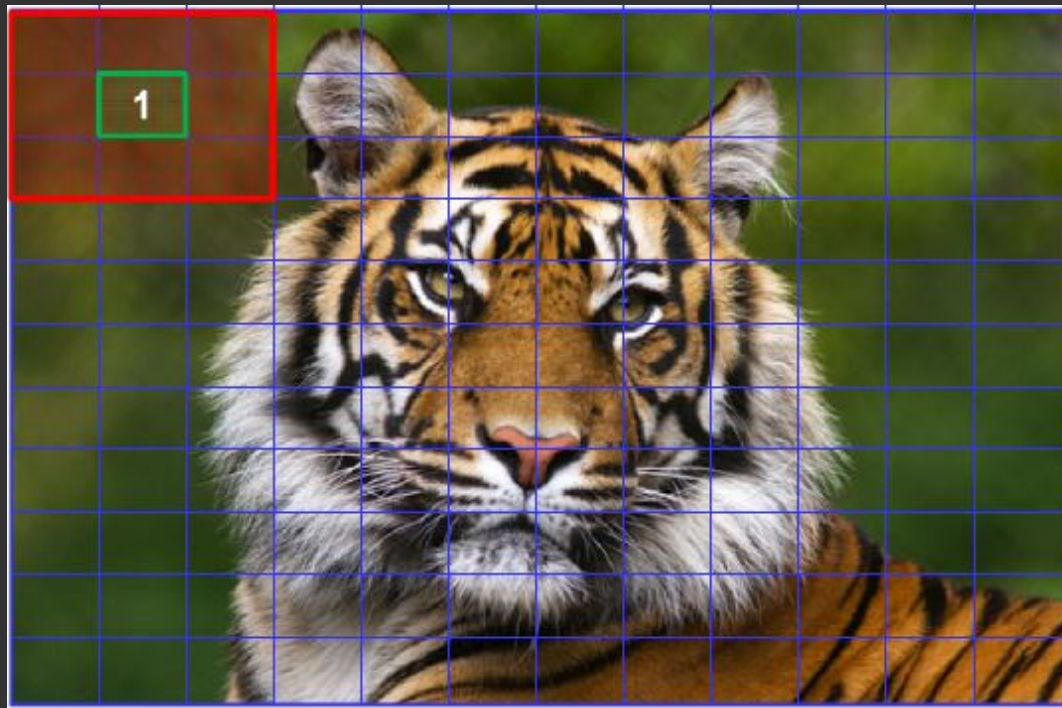
# Redes Neurais Convolucionais



- Vamos a um exemplo da operação de convolução de uma imagem RGB com 12 pixels de largura e 11 de altura por um filtro 3x3.
- Neste exemplo, para simplificar, vamos representar o filtro por um quadrado.

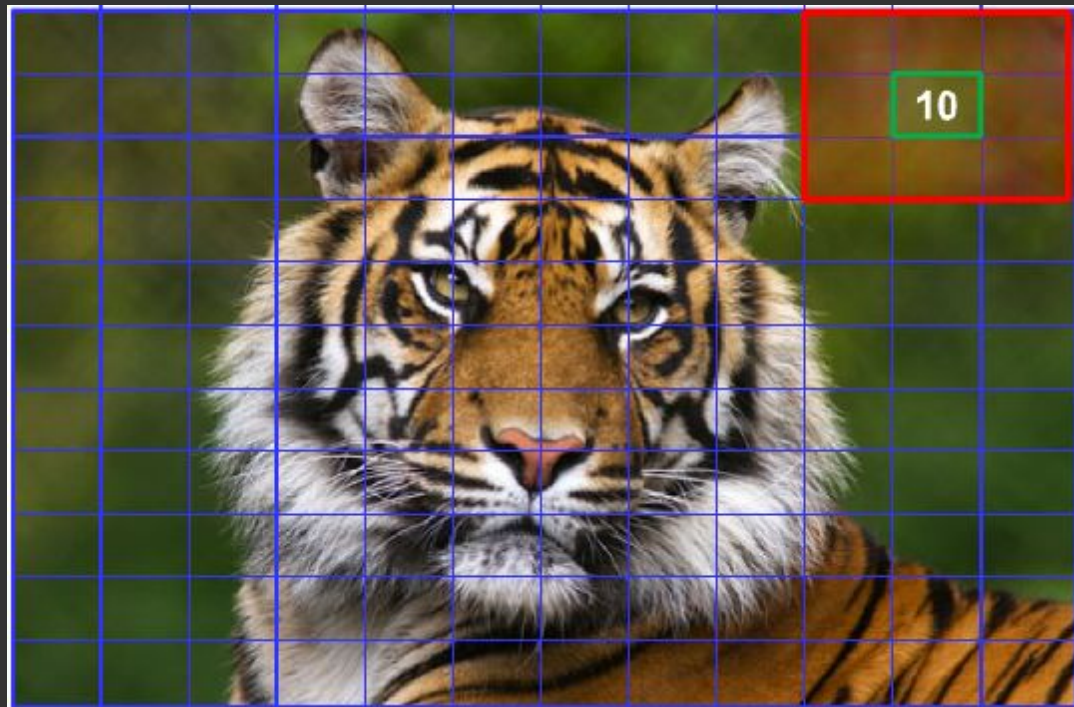


# Redes Neurais Convolucionais





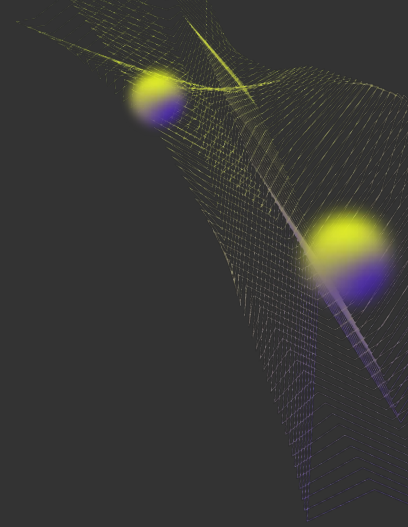
# Redes Neurais Convolucionais





# Redes Neurais Convolucionais

- Nos slides anteriores, pudemos ver que, ao processar a primeira linha da imagem (12 pixels), geramos uma linha com 10 “resultados” de convolução;
- A imagem resultante é menor que a original...





# Redes Neurais Convolucionais



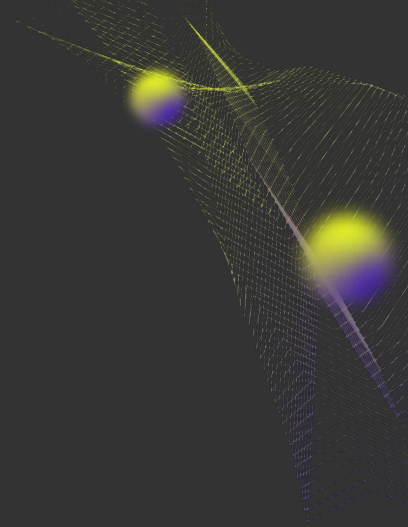
- A operação de convolução gera imagens menores que a entrada.
- O nº de linhas resultante segue a equação (o mesmo vale para colunas):

$$lin_{result} = \frac{lin_{orig} - nlin_{filtro}}{1} + 1$$



# Redes Neurais Convolucionais

- Isso significa que não podemos usar filtros muito grandes em redes muito profundas sem adotar uma estratégia para “reduzir” esse problema;
- Veremos tais estratégias na próxima seção!





# Redes Neurais Convolucionais



- Resumindo:
  - Camadas convolucionais aplicam convolução em imagens na tentativa de resolver os problemas que a MLP possui quando processa esse tipo de dado;
  - O filtro convolucional é treinado com o backpropagation!



# Redes Neurais Convolucionais



- Resumindo:
  - Uma rede convolucional profunda substitui o pipeline clássico de visão computacional (não precisamos escolher um descritor!)
  - Porém, precisamos de muitos dados...



# Redes Neurais Convolucionais



- Resumindo:
  - Embora a aplicação mais comum de RNC seja com imagens, ela pode ser aplicada em outros sinais com característica de vizinhança, como por exemplo, sinais temporais;



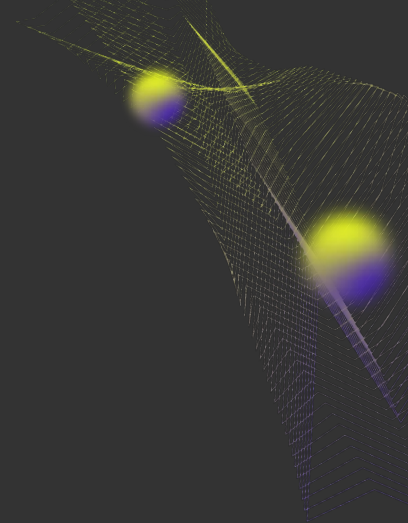
### 3. Parâmetros da CNN e Pooling





# Parâmetros da CNN e Pooling

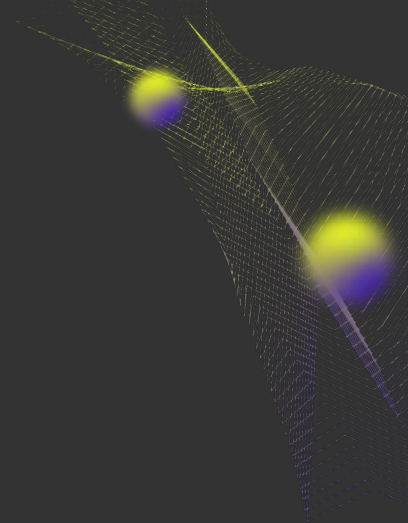
- A camada convolucional possui alguns parâmetros. Os mais óbvios são:
  - Número de canais da entrada
  - Número de canais da saída (nº de filtros)
  - Tamanho do filtro





# Parâmetros da CNN e Pooling

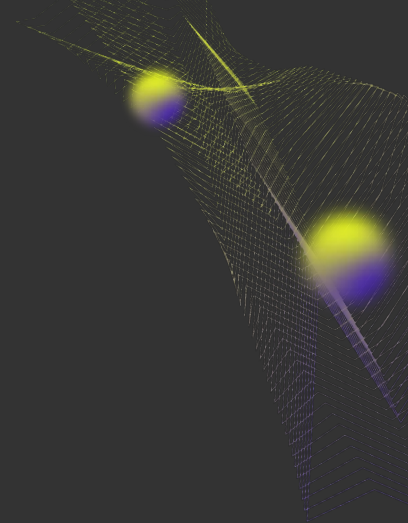
- Temos também outros parâmetros que interferem no tamanho da saída;
- Na última seção, vimos que o tamanho da saída da convolução 2D varia de acordo com o tamanho do filtro (e da entrada);





# Parâmetros da CNN e Pooling

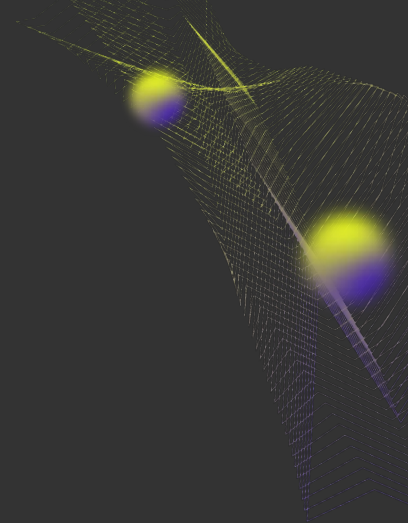
- Os 3 principais são:
  - Padding
  - Stride
  - Dilation





# Parâmetros da CNN e Pooling

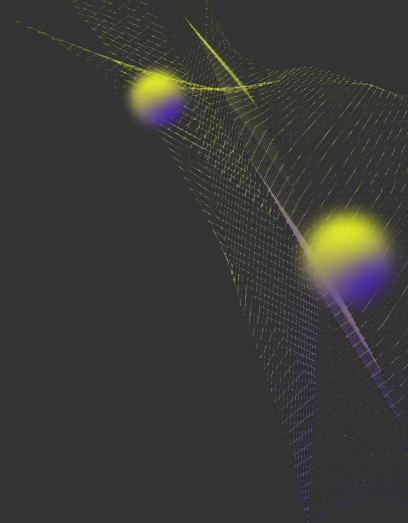
- O parâmetro padding controla um pré-processamento feito na imagem original:
  - uma “borda” é criada (ou não) expandindo o tamanho na imagem





# Parâmetros da CNN e Pooling

- Geralmente há 3 possibilidades:
  - “valid”
  - “same”
  - número inteiro



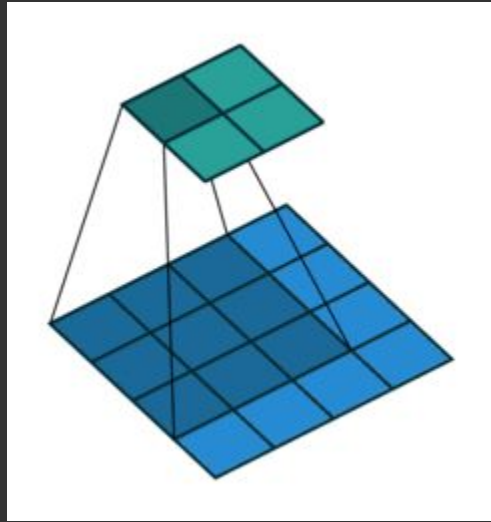


# Parâmetros da CNN e Pooling

- Geralmente há 3 possibilidades:
  - “valid”
    - Não é feito nenhum padding



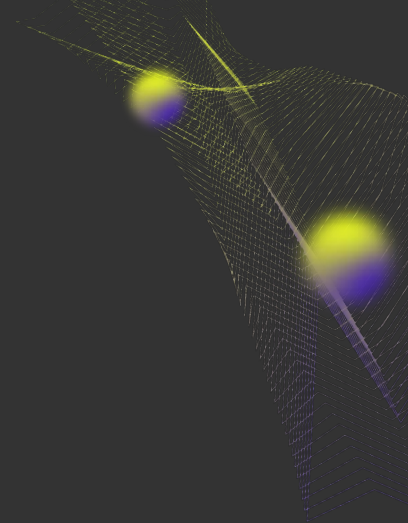
# Parâmetros da CNN e Pooling





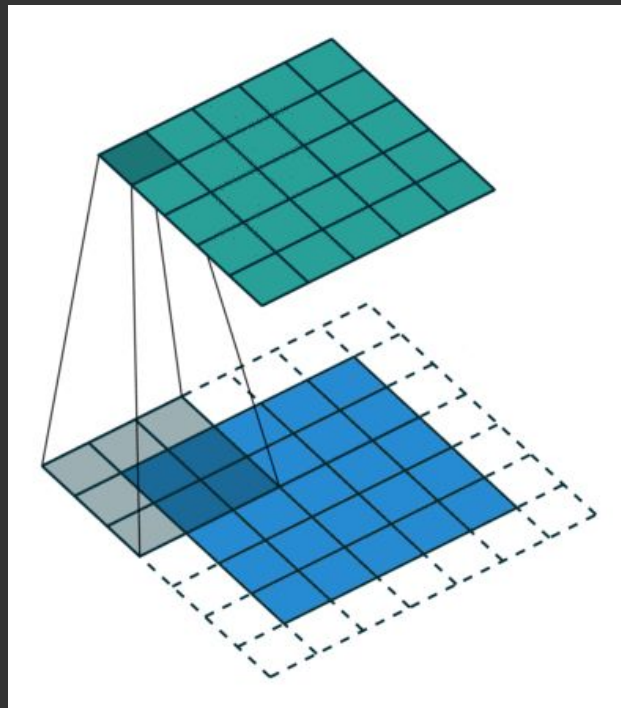
# Parâmetros da CNN e Pooling

- Geralmente há 3 possibilidades:
  - “same”
    - É feito um cálculo automático de forma a criar uma borda que resulte uma saída com tamanho igual ao da entrada





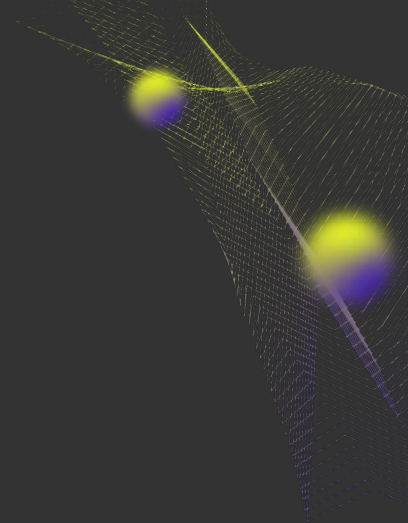
# Parâmetros da CNN e Pooling





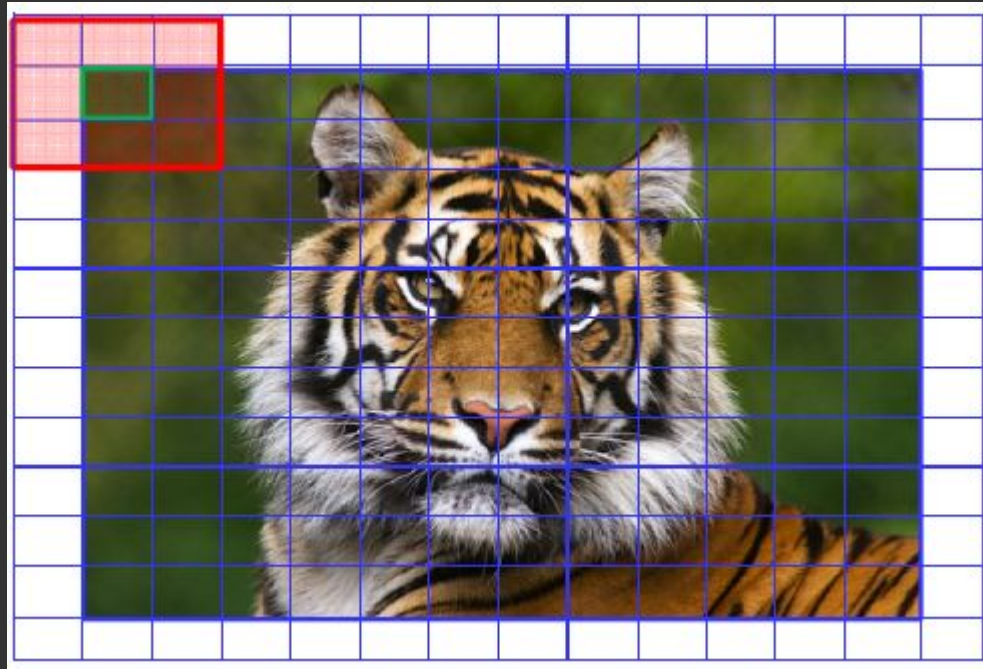
# Parâmetros da CNN e Pooling

- Geralmente há 3 possibilidades:
  - número inteiro  $N$ 
    - É criada uma borda com  $N$  pixels em volta da imagem



# Parâmetros da CNN e Pooling

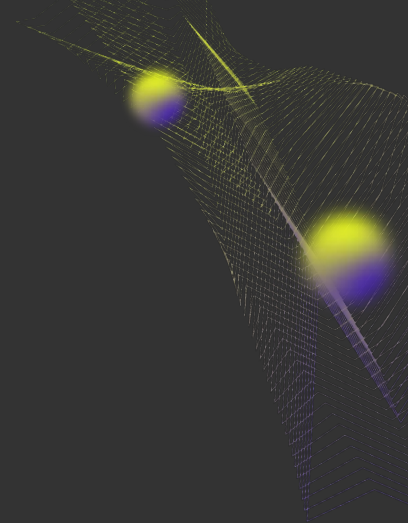
- $N = 1$





# Parâmetros da CNN e Pooling

- Independente do modo de *padding*, é possível definir o valor da borda:
  - número fixo (geralmente zeros)
  - refletido
  - circular
  - replicar



# Parâmetros da CNN e Pooling

- Zero

|   |     |     |     |     |     |   |
|---|-----|-----|-----|-----|-----|---|
| 0 | 0   | 0   | 0   | 0   | 0   | 0 |
| 0 | 236 | 180 | 2   | 89  | 65  | 0 |
| 0 | 84  | 242 | 162 | 105 | 174 | 0 |
| 0 | 152 | 104 | 9   | 40  | 200 | 0 |
| 0 | 100 | 206 | 43  | 123 | 248 | 0 |
| 0 | 108 | 246 | 33  | 125 | 224 | 0 |
| 0 | 84  | 163 | 34  | 163 | 213 | 0 |
| 0 | 0   | 0   | 0   | 0   | 0   | 0 |

# Parâmetros da CNN e Pooling

- Refletido

|     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|
| 242 | 235 | 180 | 2   | 89  | 65  | 105 |
| 180 | 236 | 180 | 2   | 89  | 65  | 65  |
| 242 | 84  | 242 | 162 | 105 | 174 | 174 |
| 104 | 152 | 104 | 9   | 40  | 200 | 200 |
| 206 | 100 | 206 | 43  | 123 | 248 | 248 |
| 243 | 108 | 246 | 33  | 125 | 224 | 224 |
| 84  | 84  | 163 | 34  | 163 | 213 | 213 |
| 246 | 84  | 163 | 34  | 163 | 213 | 125 |

# Parâmetros da CNN e Pooling

- Circular

|     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|
| 213 | 84  | 163 | 34  | 163 | 213 | 84  |
| 65  | 236 | 180 | 2   | 89  | 65  | 180 |
| 174 | 84  | 242 | 162 | 105 | 174 | 242 |
| 200 | 152 | 104 | 9   | 40  | 200 | 104 |
| 248 | 100 | 206 | 43  | 123 | 248 | 206 |
| 224 | 108 | 246 | 33  | 125 | 224 | 243 |
| 213 | 84  | 163 | 34  | 163 | 213 | 84  |
| 65  | 236 | 180 | 2   | 89  | 65  | 236 |

# Parâmetros da CNN e Pooling

- Replicado

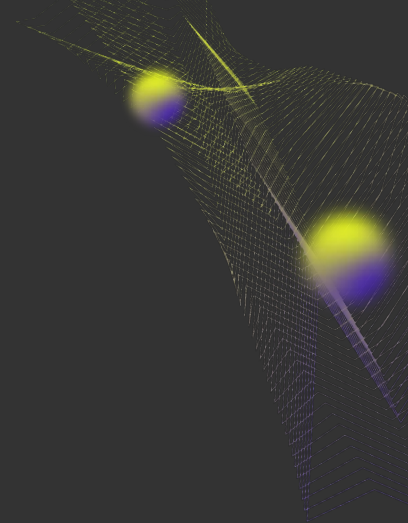
|     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|
| 236 | 236 | 180 | 2   | 89  | 65  | 65  |
| 236 | 236 | 180 | 2   | 89  | 65  | 65  |
| 84  | 84  | 242 | 162 | 105 | 174 | 174 |
| 152 | 152 | 104 | 9   | 40  | 200 | 200 |
| 100 | 100 | 206 | 43  | 123 | 248 | 248 |
| 108 | 108 | 246 | 33  | 125 | 224 | 224 |
| 84  | 84  | 163 | 34  | 163 | 213 | 213 |
| 84  | 84  | 163 | 34  | 163 | 213 | 213 |



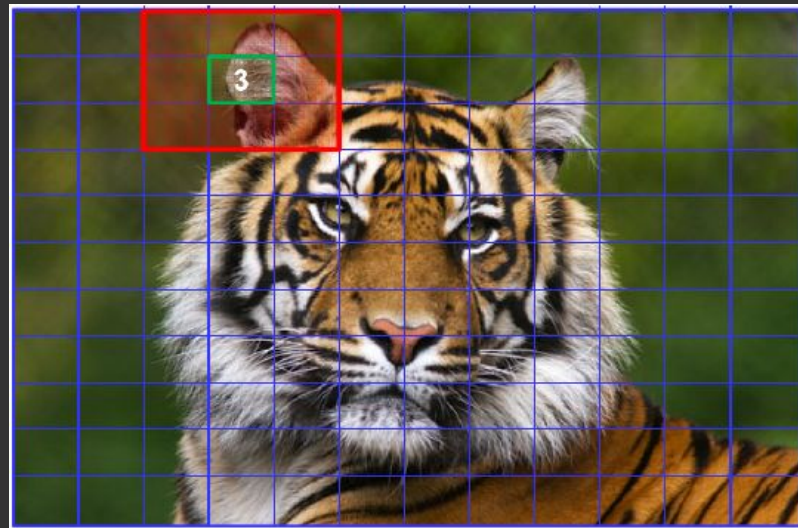
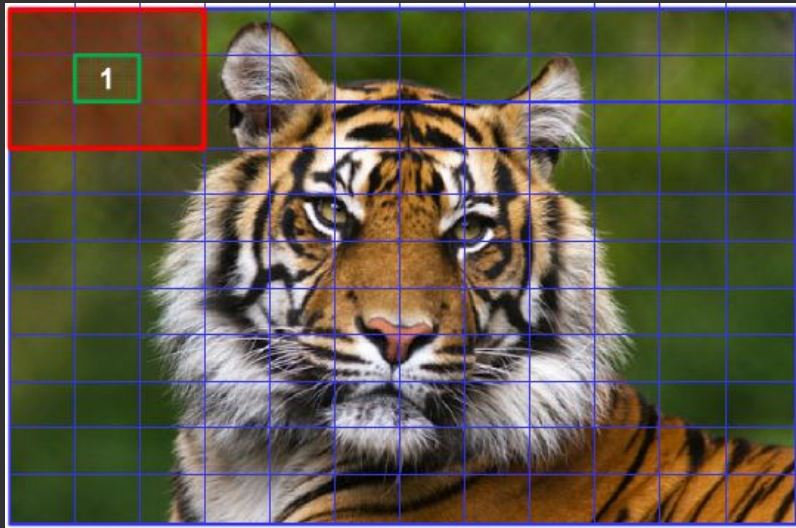


# Parâmetros da CNN e Pooling

- O segundo parâmetro, chamado *stride*, controla a quantidade de “pulos” do filtro convolucional
  - Até agora, o filtro “desliza” na imagem pixel a pixel. Isso significa  $\text{stride} = 1$ .

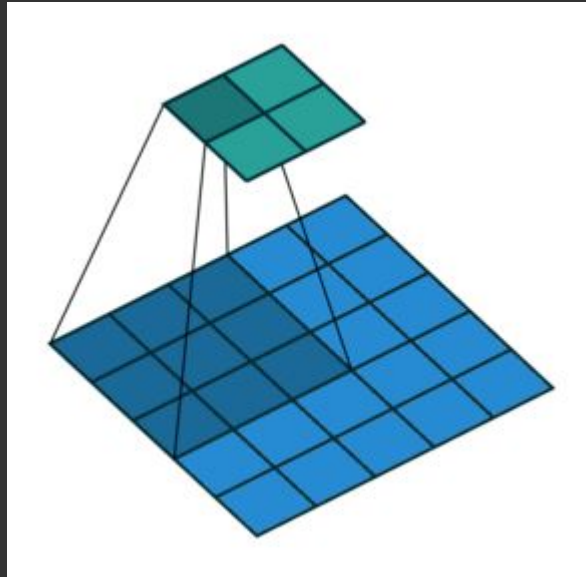


# Parâmetros da CNN e Pooling



# Parâmetros da CNN e Pooling

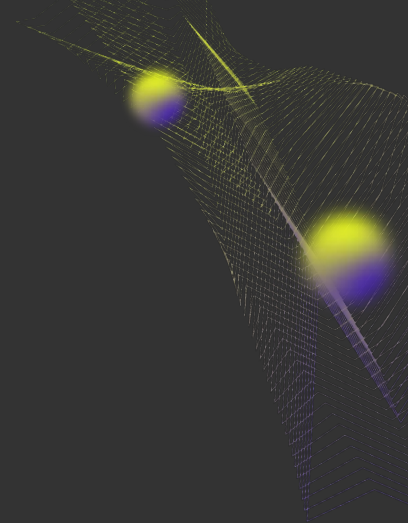
- Stride = 2





# Parâmetros da CNN e Pooling

- O parâmetro stride é usado como forma de reduzir o tamanho da saída e reduzir o processamento da rede;
- Há outras formas, que veremos ainda nesta seção;

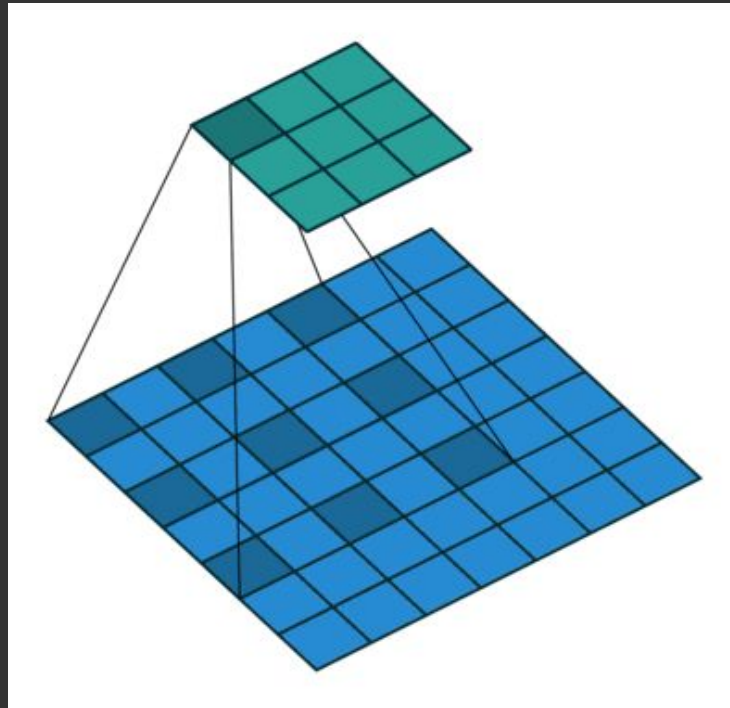




# Parâmetros da CNN e Pooling

- O terceiro parâmetro, chamado *dilation*, controla o espaçamento entre os pixels do filtro convolucional
  - Normalmente, os filtros consideram *pixels* vizinhos (dilatação = 1). Caso quisermos aumentar esse espaçamento, aumentamos esse parâmetro
  - Usado para diminuir tamanho da imagem;

# Parâmetros da CNN e Pooling



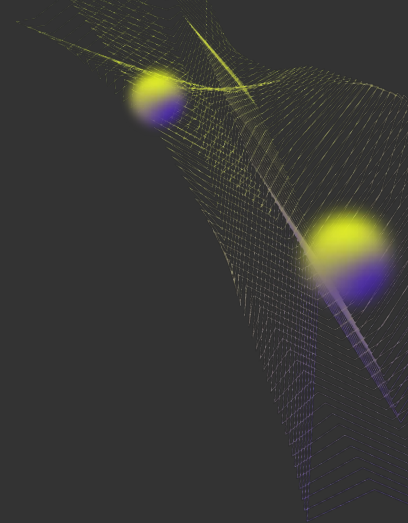


# Parâmetros da CNN e Pooling

- Em geral, o tamanho do resultado de uma convolução é dado por:

$$TS = \frac{(N - F + 2P)}{S} + 1$$

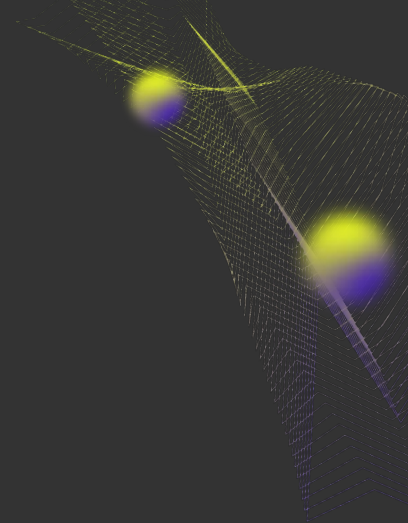
- N é o tamanho da entrada, F é o tamanho do filtro, P é o padding, S é o stride.





# Parâmetros da CNN e Pooling

- Para finalizar a seção, vamos ver uma operação (que gera um novo tipo de camada de RNA) muito utilizado em conjunto com camadas convolucionais
  - Também são usadas para reduzir dimensão
  - Recebem o nome de “pooling”.







# Parâmetros da CNN e Pooling

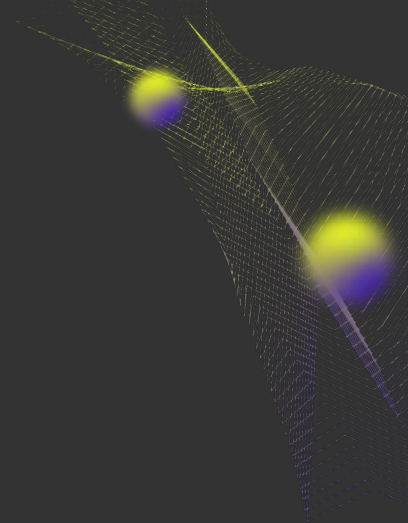


- Uma camada de pooling aplica uma operação em cada uma das matrizes com a finalidade de “comprimir” os resultados e reduzir a dimensão
  - Ex: Supondo que temos 10 canais na saída de uma camada conv, continuaremos a ter 10 canais na saída do pooling, mas com dimensão menor



# Parâmetros da CNN e Pooling

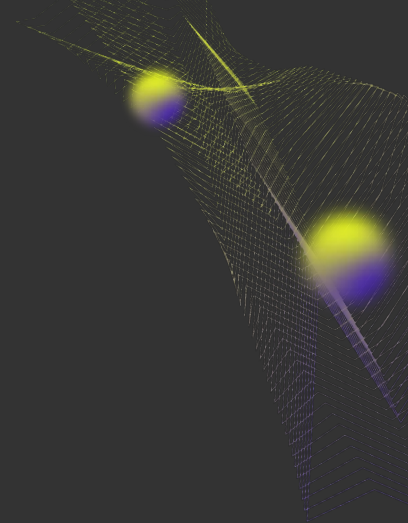
- A ideia do pooling é similar a convolução
  - mas uma operação não linear é aplicada
    - Uma janela “percorre a imagem”  
aplicando uma determinada  
operação matemática





# Parâmetros da CNN e Pooling

- Os dois tipos mais comuns de pooling são:
  - Max pooling
  - Mean/Average pooling



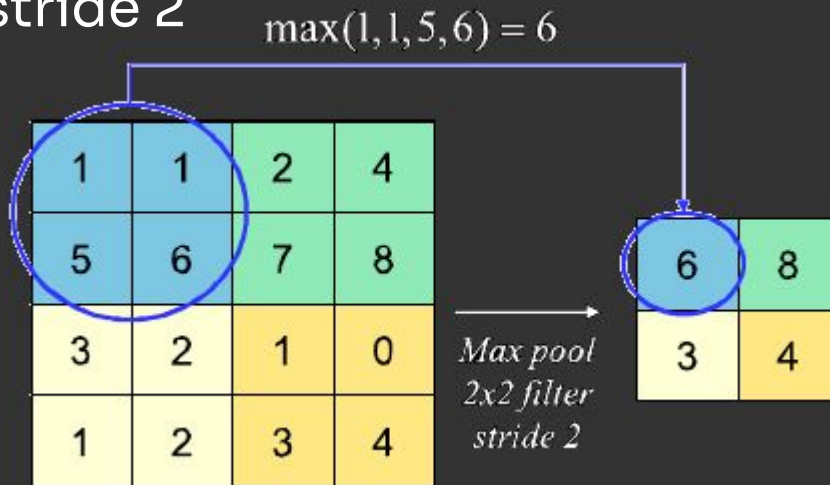
# Parâmetros da CNN e Pooling

- Ex: Max Pooling com uma imagem 4x4, janela 2x2 e stride 2

|   |   |   |   |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 5 | 6 | 7 | 8 |
| 3 | 2 | 1 | 0 |
| 1 | 2 | 3 | 4 |

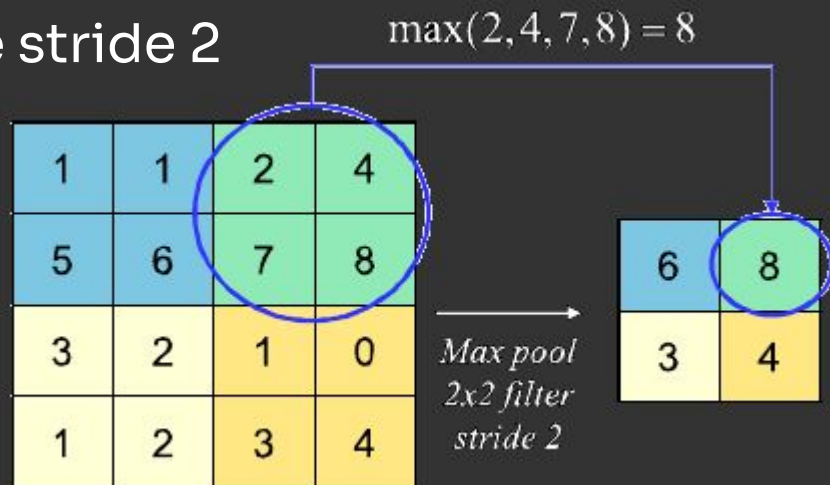
# Parâmetros da CNN e Pooling

- Ex: Max Pooling com uma imagem 4x4 e janela 2x2 e stride 2



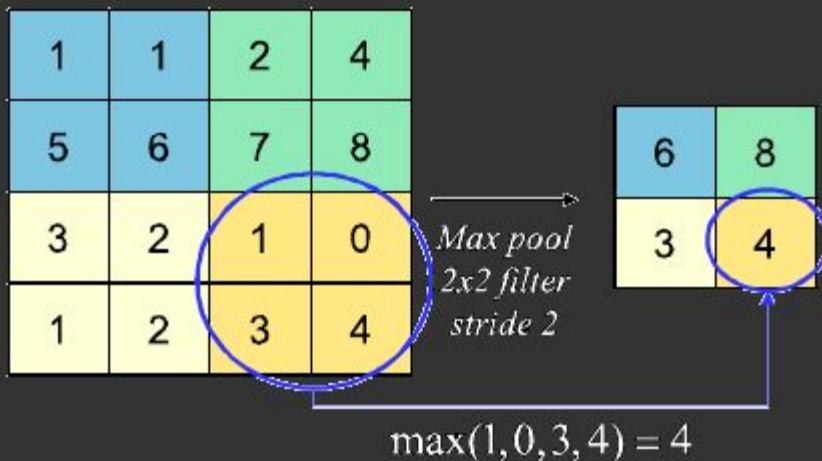
# Parâmetros da CNN e Pooling

- Ex: Max Pooling com uma imagem 4x4 e janela 2x2 e stride 2



# Parâmetros da CNN e Pooling

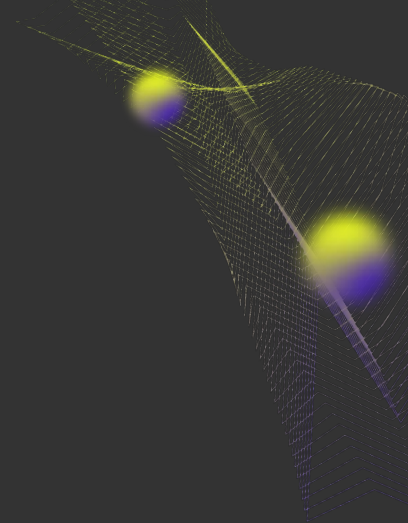
- Ex: Max Pooling com uma imagem 4x4 e janela 2x2 e stride 2





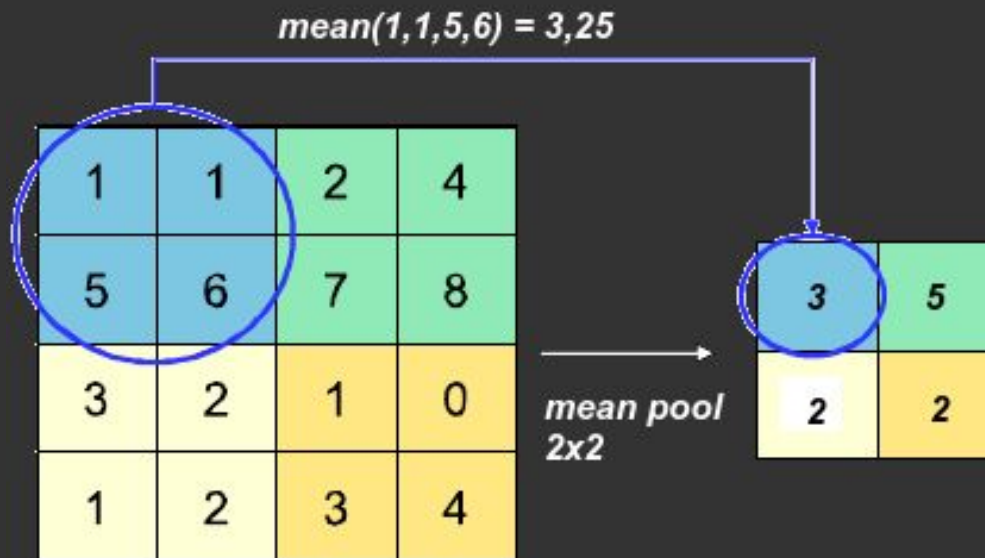
# Parâmetros da CNN e Pooling

- Já a operação de Mean/Average pooling é similar, onde o resultado é a média dos valores da janela sobreposta





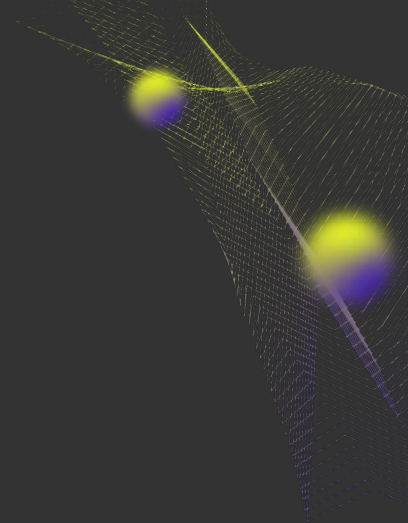
# Parâmetros da CNN e Pooling





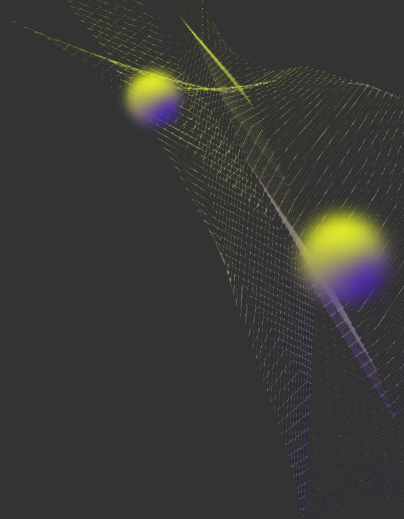
# Parâmetros da CNN e Pooling

- Observações:
  - A imagem de saída fica invariante a pequenas transformações
  - Informações importantes são preservadas na saída
  - Porém com tamanho menor (custo computacional reduzido!)





# Resumo

- Camadas Convolucionais
    - Stride, tamanho do filtro, dilation e padding
  - Camadas de Pooling
    - max ou avg pooling
  - Camadas Totalmente Conectadas
    - Geralmente usada no final de uma CNN para problemas de classificação
- 

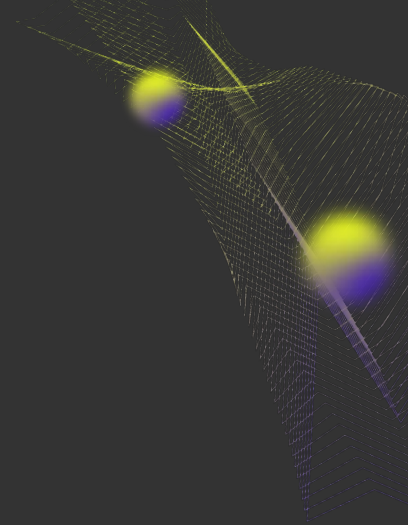


## 4. Pytorch e CNNs



# Pytorch e CNNs

- Existem diversas bibliotecas que implementam RNAs e CNNs
  - Pytorch é uma delas!
  - Não vamos reinventar a roda, vamos usá-la
- Material de estudo disponibilizado no AVA.



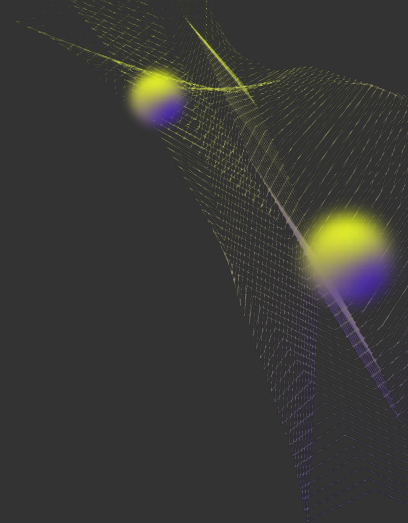


## 5. Laboratório 06



## Laboratório 6

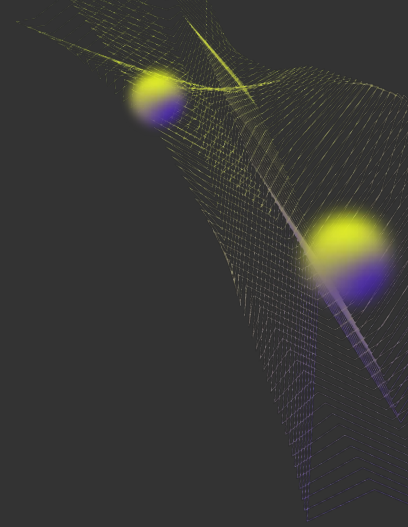
- No 6º laboratório da disciplina, vocês irão ter contato com uma rede neural convolucional simples, usando PyTorch!
- No Moodle!





# Laboratório 6

- Após o Laboratório 6, temos o EA3 da disciplina!





INTELIGÊNCIA  
ARTIFICIAL &  
CIÊNCIA DE DADOS

**Bruno Légora Souza da Silva**

Professor do Departamento de  
Informática/UFES

*[bruno.l.silva@ufes.br](mailto:bruno.l.silva@ufes.br)*