

# **GEX1082 - Tópicos Especiais em Computação XXXIII**

## **Deep Learning**

**Redes Neurais Convolucionais**



1100/1101 - CIÊNCIA DA COMPUTAÇÃO

Prof. Dr. Giancarlo D. Salton

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
ooooo

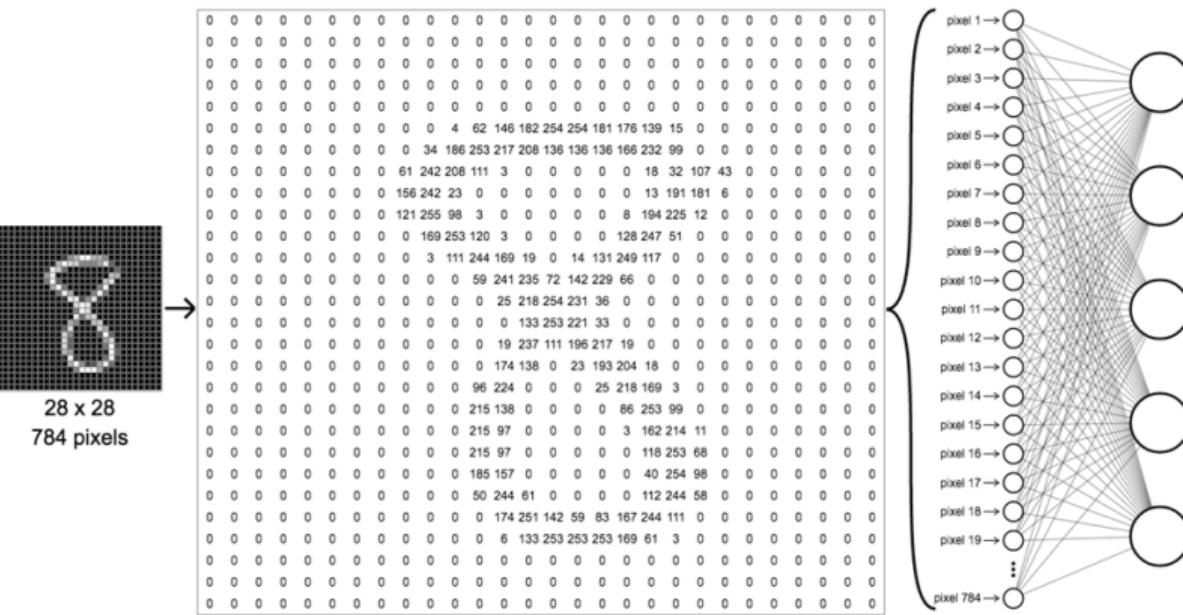
## Redes Neurais Convolucionais

## Representações Distribuídas

## Regularização

## Métodos adaptativos de Gradiente Descendente

## Resumo



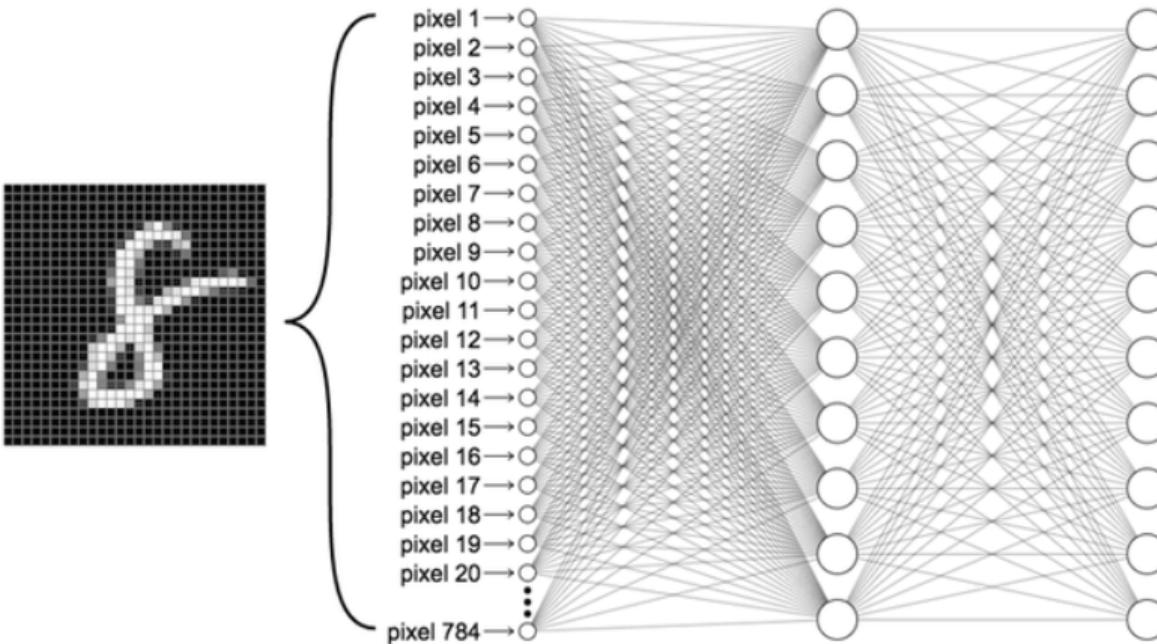
ConvNets  
oooooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooooooo

Resumo  
oooooo



ConvNets

●ooooooooo

Feature Maps

oooooooooooo

Regularization

oooooooooooooo

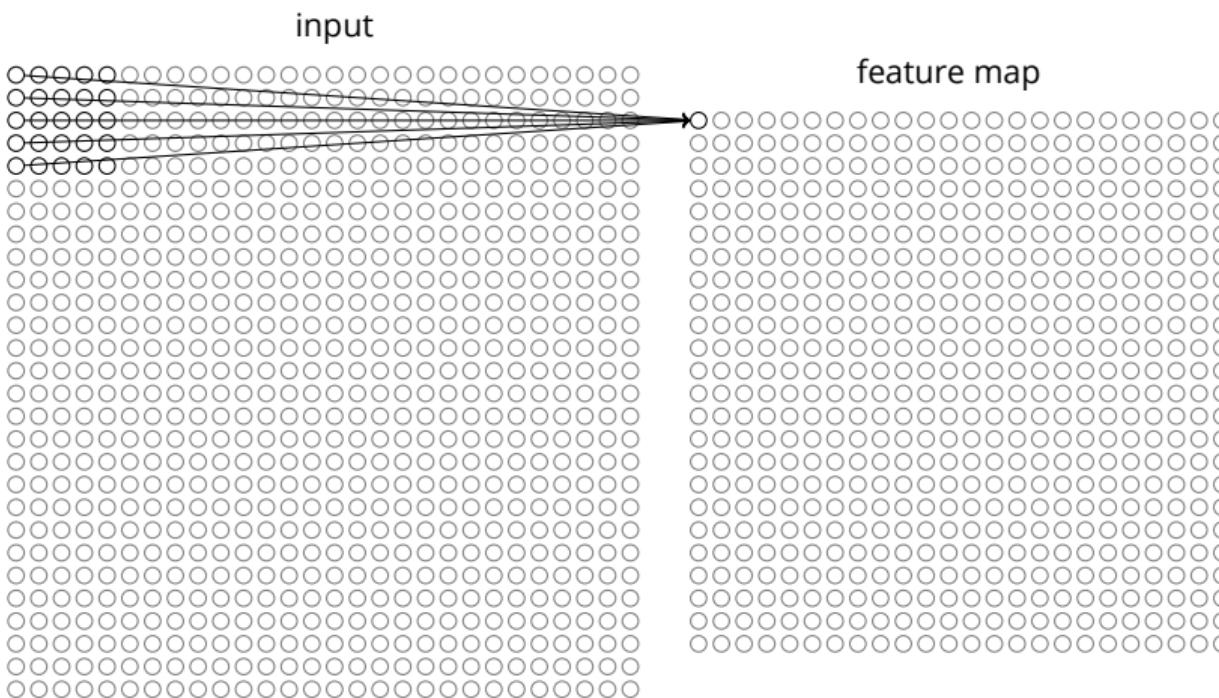
Optimizers

oooooooooooooooooooo

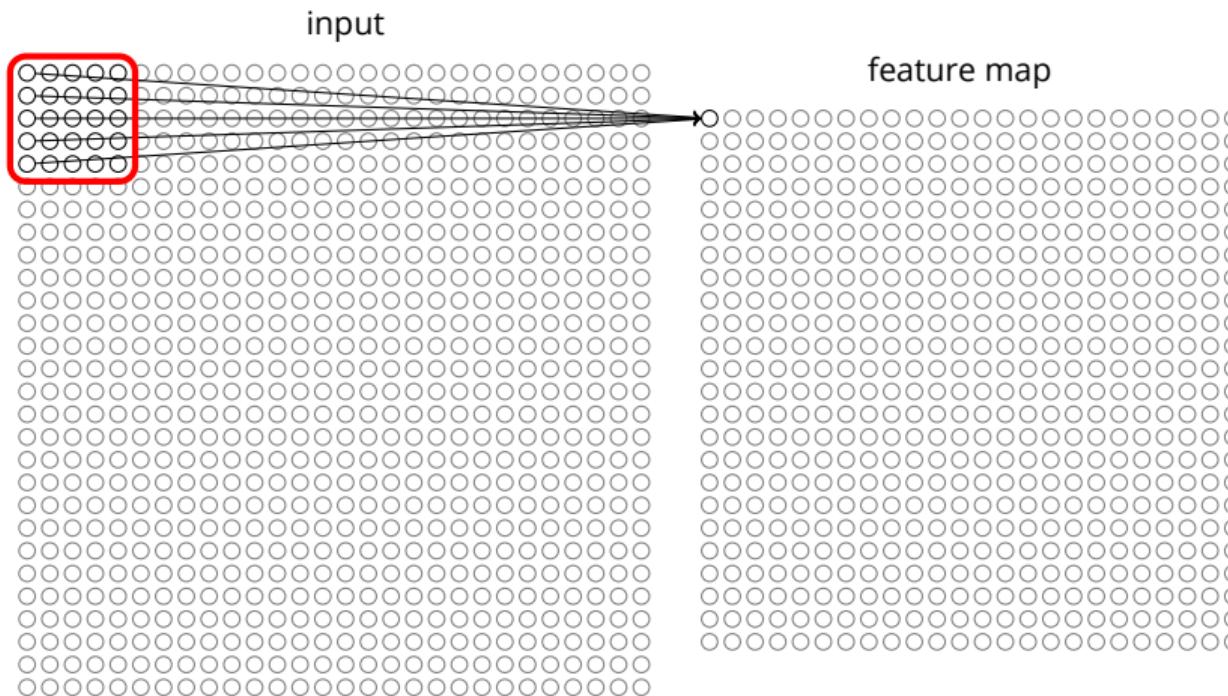
Resumo

ooooo

# Redes Neurais Convolucionais



$$z = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) * input(N_i, k)$$



$$z = bias(C_{out_j}) + \sum_{k=0}^{C_{in}-1} weight(C_{out_j}, k) * input(N_i, k)$$

## ConvNets

## Feature Maps



## Regularization

---

## Optimizers



## Resumo

ConvNets

○○○●○○○○

Feature Maps

○○○○○○○○○○

Regularization

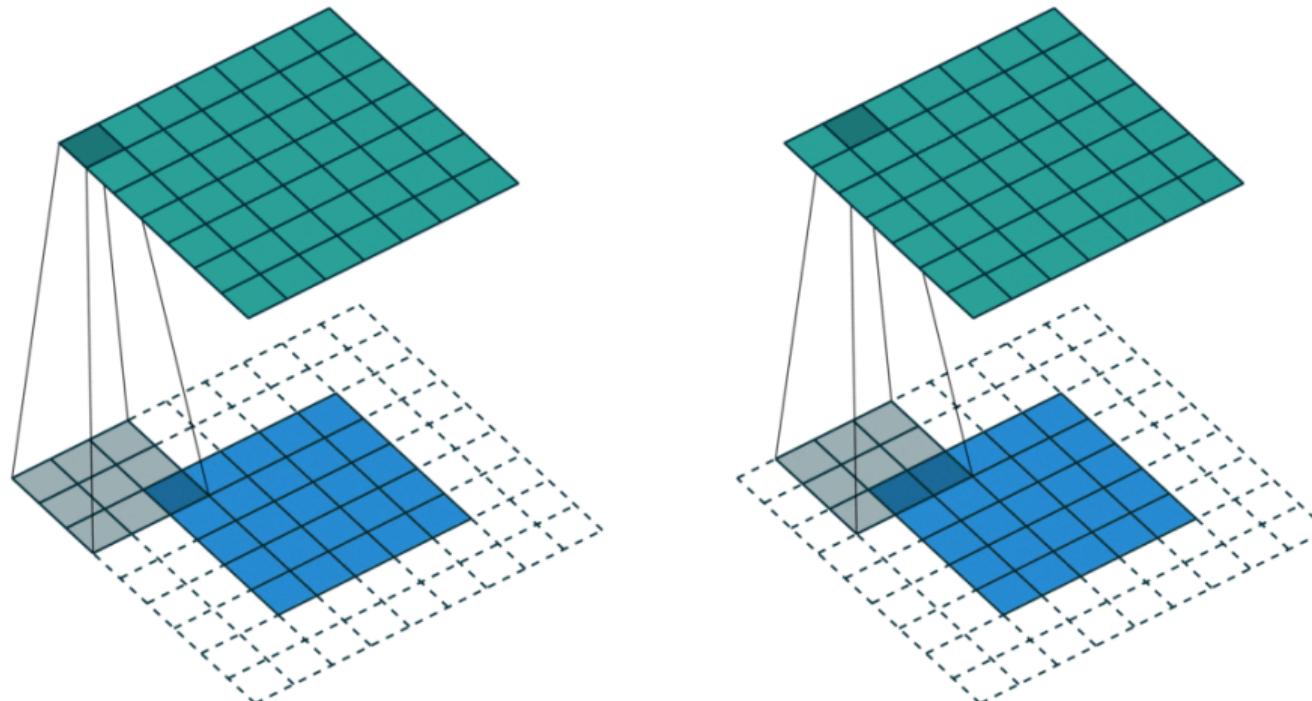
○○○○○○○○○○○○

Optimizers

○○○○○○○○○○○○○○○○

Resumo

○○○○○



ConvNets  
oooooooo●ooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooo

Resumo  
ooooo

- As *conv nets* e as redes *feedforward* são igualmente organizadas em camadas
- A diferença reside no fato de que as primeiras camadas das *conv nets* são compostas pelas convoluções
  - ✓ As convoluções extraem *features* que servirão de entrada para as camadas finais da rede
  - ✓ As camadas finais, por sua vez, são compostas por 1 ou mais camadas “tradicionais”

- Entre uma camada convolucional e a próxima camada, utilizamos uma operação de *pooling*
- As operações de *pooling* se comportam de forma semelhante as convoluções
  - ✓ cada vez que um *feature map* é aplicado em um pedaço da imagem, o *pooling* é aplicado a ativação dos neurônios daquele *feature map*
- *Poolings* mais comuns:
  - ✓ *max pooling*: seleciona o valor mais alto obtido pelo *feature map*
  - ✓ *mean pooling*: obtém a média dos valores obtidos pelo *feature map*

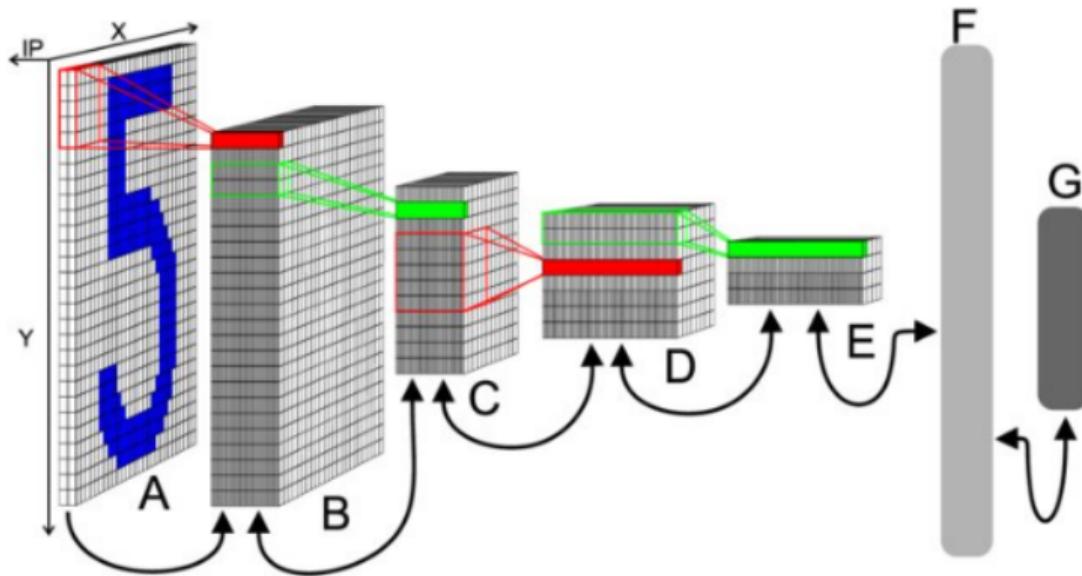
ConvNets  
oooooooo●○

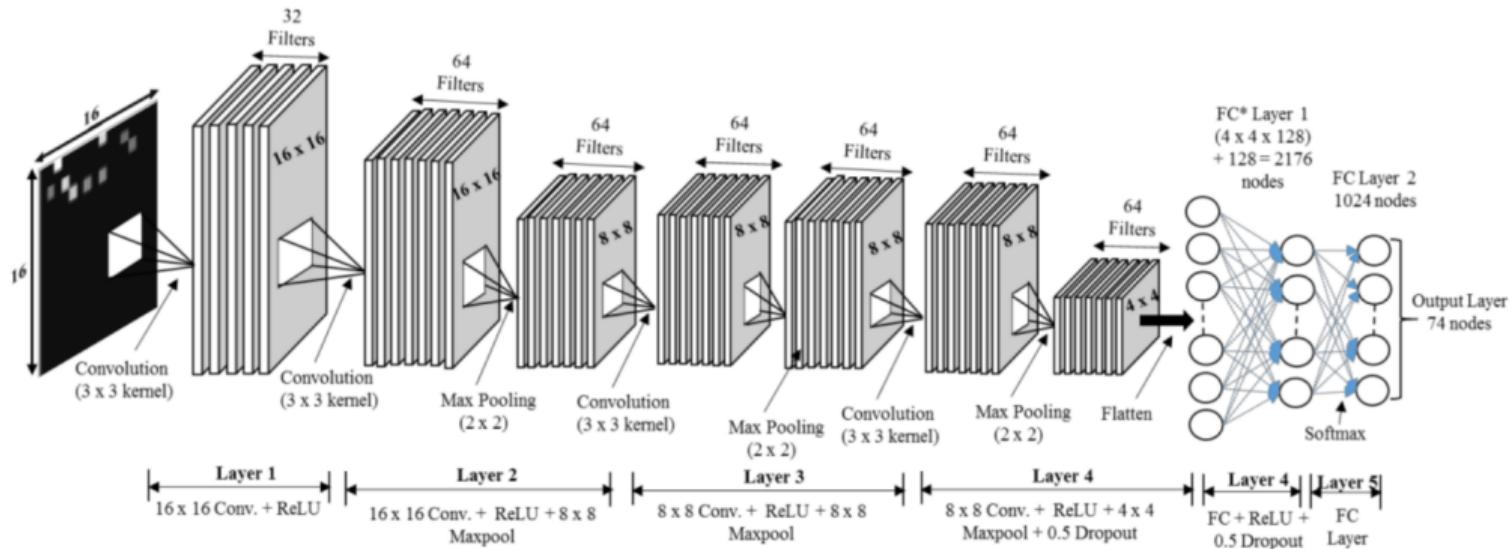
Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooooooo

Resumo  
ooooo





\*FC = Fully Connected

ConvNets  
oooooooooo

Feature Maps  
●oooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooooooo

Resumo  
ooooo

# Representações Distribuídas

ConvNets

oooooooooooo

Feature Maps

○●oooooooooooo

Regularization

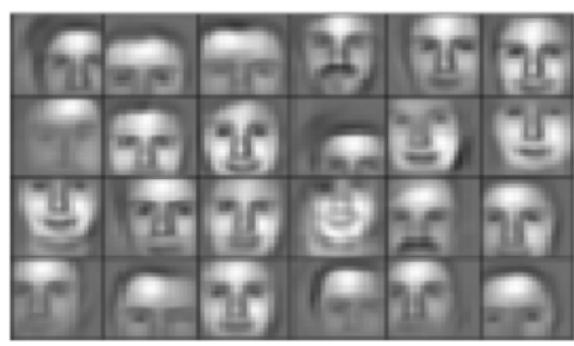
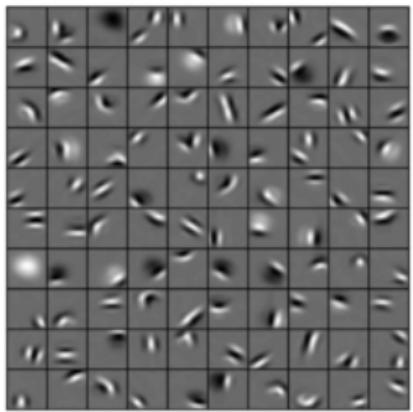
oooooooooooooooo

Optimizers

oooooooooooooooooooo

Resumo

oooooo



ConvNets

oooooooooo

Feature Maps

○○●○○○○○○

Regularization

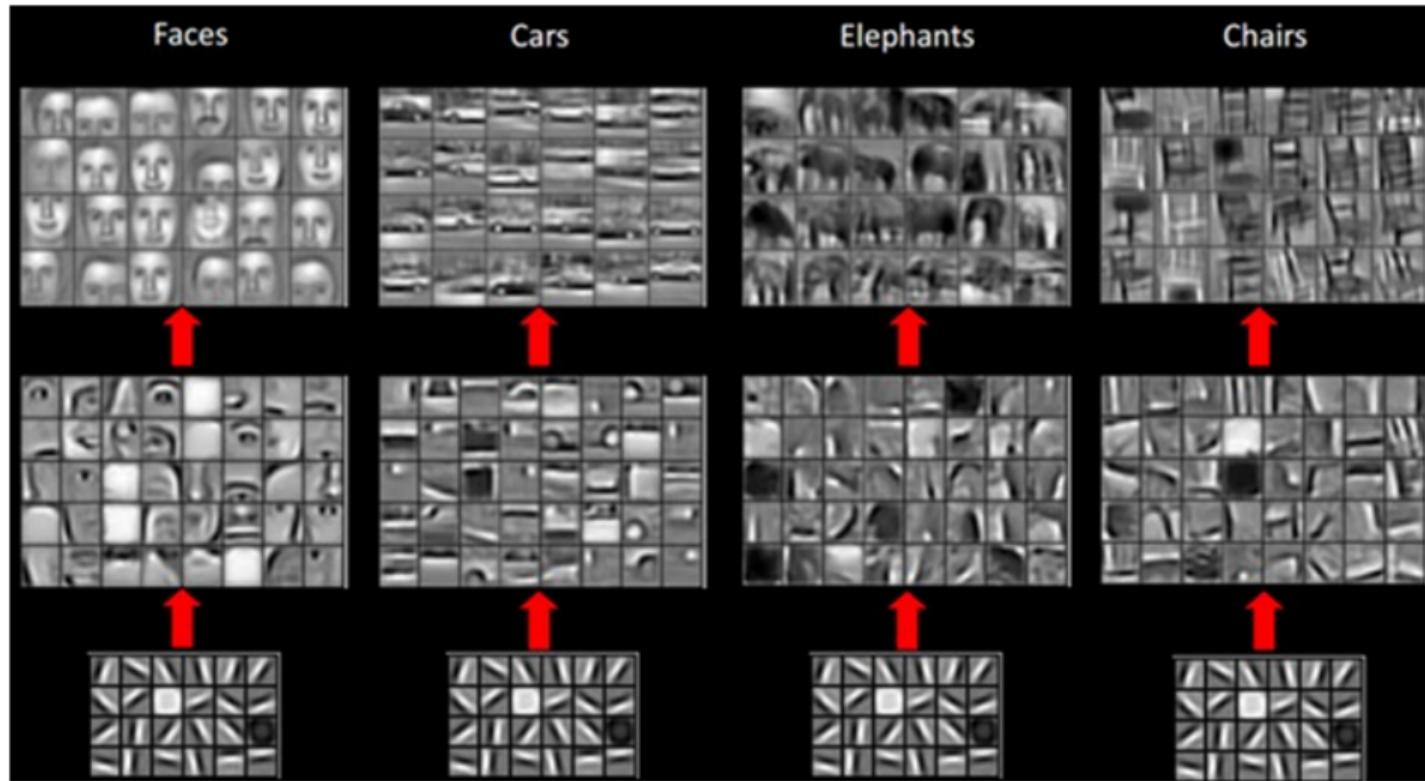
oooooooooooo

Optimizers

oooooooooooooooo

Resumo

○○○○○



ConvNets  
oooooooooooo

Feature Maps  
ooo●oooooooo

Regularization  
oooooooooooooooo

Optimizers  
oooooooooooooooooooo

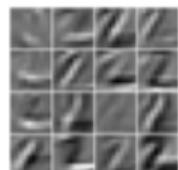
Resumo  
oooooo



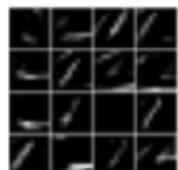
conv1



relu1



conv2



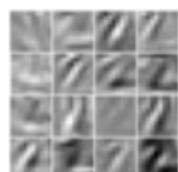
relu2



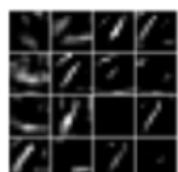
conv1



relu1



conv2



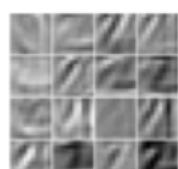
relu2



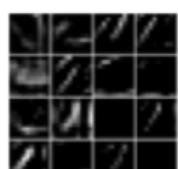
conv1



relu1



conv2



relu2

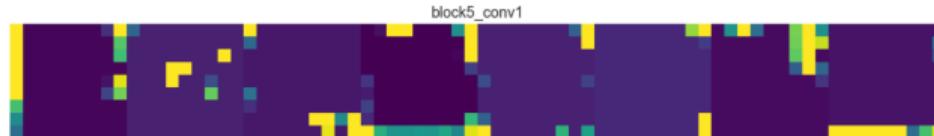
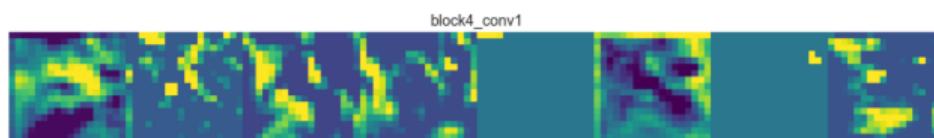
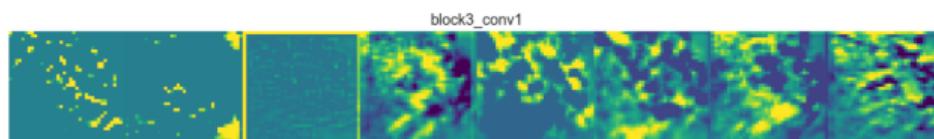
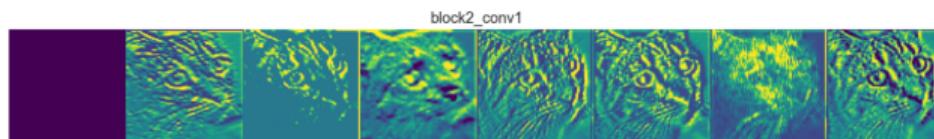
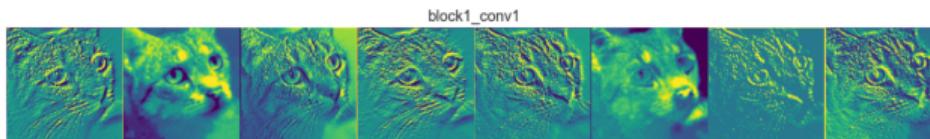
ConvNets  
oooooooooooo

Feature Maps  
oooo●oooooooo

Regularization  
oooooooooooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
oooooo



- As *features* extraídas ao final da hierarquia são então fornecidas como entrada para uma ou mais camadas tradicionais
  - ✓ Embora este passo seja opcional, é altamente recomendado que seja executado
- Por sua vez, as saídas destas camadas são posteriormente fornecidas para a camada de saída
- Por isso, dizemos que as camadas anteriores as camadas tradicionais são chamadas de *extratores de features*
  - ✓ A camada de saída da rede neural é então chamada de *classificador*.

- Esta hierarquia de *features* é o que torna as redes neurais interessantes
- Quanto mais longa a hierarquia, maior a chance de se extrair *features* que podem fazer a diferença entre um bom resultado e um resultado ainda melhor
- Nós ainda devemos considerar os canais de cores utilizados na formação da imagem
  - ✓ Quantos mais canais de cores, mais filtros devemos utilizar!
- No entanto, treinar estes modelos requer um *dataset* grande e um poder computacional maior ainda

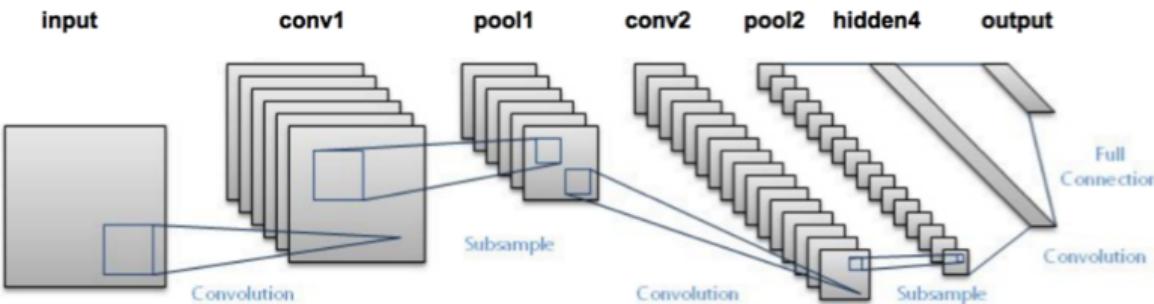
ConvNets  
oooooooooo

Feature Maps  
oooooooo●ooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooo

Resumo  
ooooo



- LeNet-5 (1998)

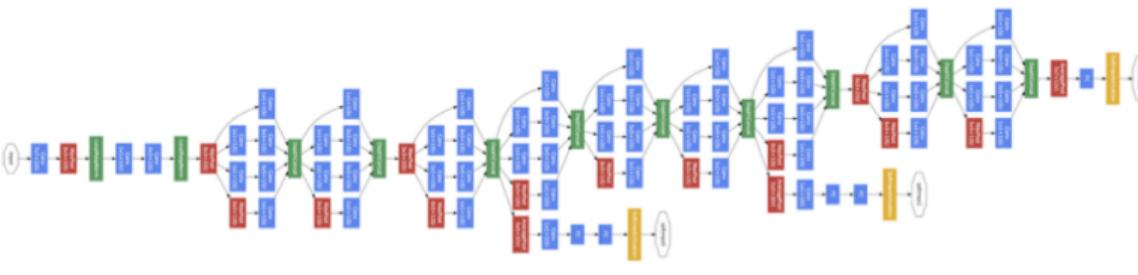
ConvNets  
oooooooooo

Feature Maps  
oooooooo●o

Regularization  
oooooooooooo

Optimizers  
oooooooooooo

Resumo  
ooooo



Convolution  
Pooling  
Softmax  
Other

- GoogleNet/Inception (2014)

- Por isso, existem iniciativas em competições abertas recomendando que os vencedores compartilhem seus modelos pré-treinados
- Muitos pesquisadores e empresas na área de visão computacional costumam compartilhar o código fonte e várias versões pré-treinadas de seus modelos
- Desta forma, podemos utilizar os modelos da forma que estão, adaptá-los ou utilizar parte dos mesmos para criar outros modelos
  - ✓ A utilização de parte dos modelos, especialmente os extratores de *features*, já é prática comum na área

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
●oooooooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
ooooo

# Regularização

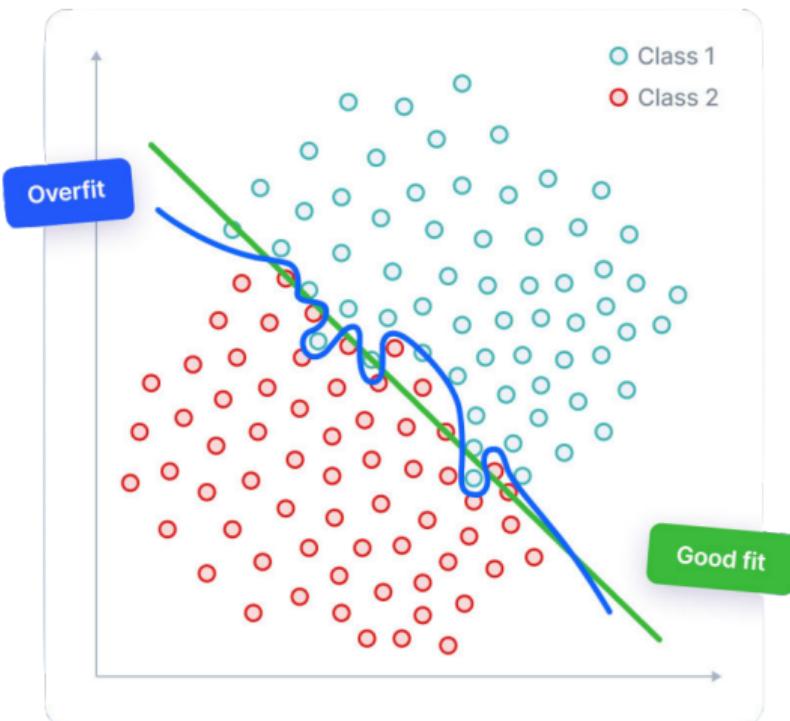
ConvNets  
oooooooooooo

Feature Maps  
oooooooooooo

Regularization  
o●oooooooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
oooooo



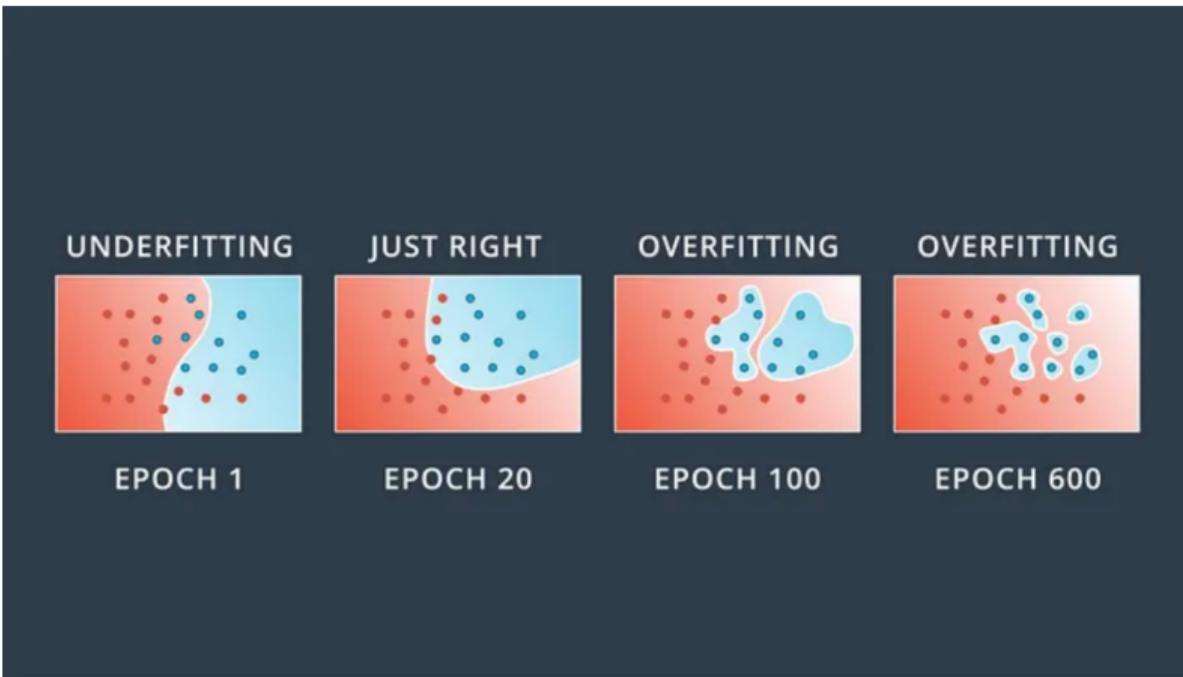
ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oo●oooooooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
ooooo



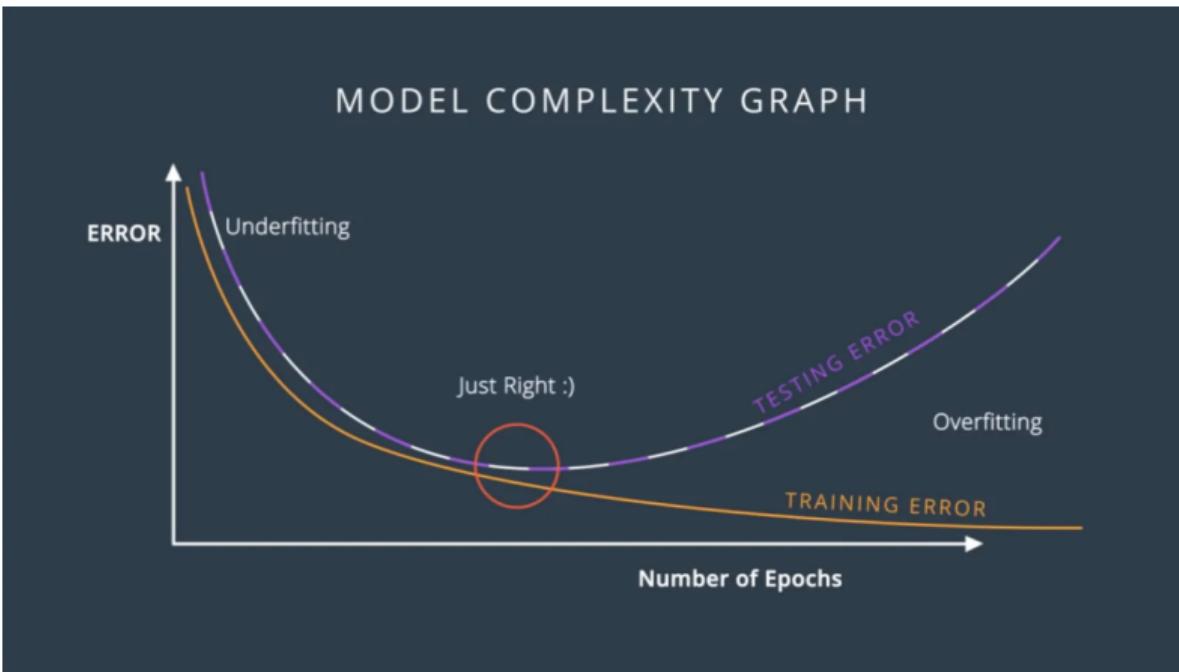
ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooo●oooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
ooooo



ConvNets

oooooooooooo

Feature Maps

oooooooooooo

Regularization

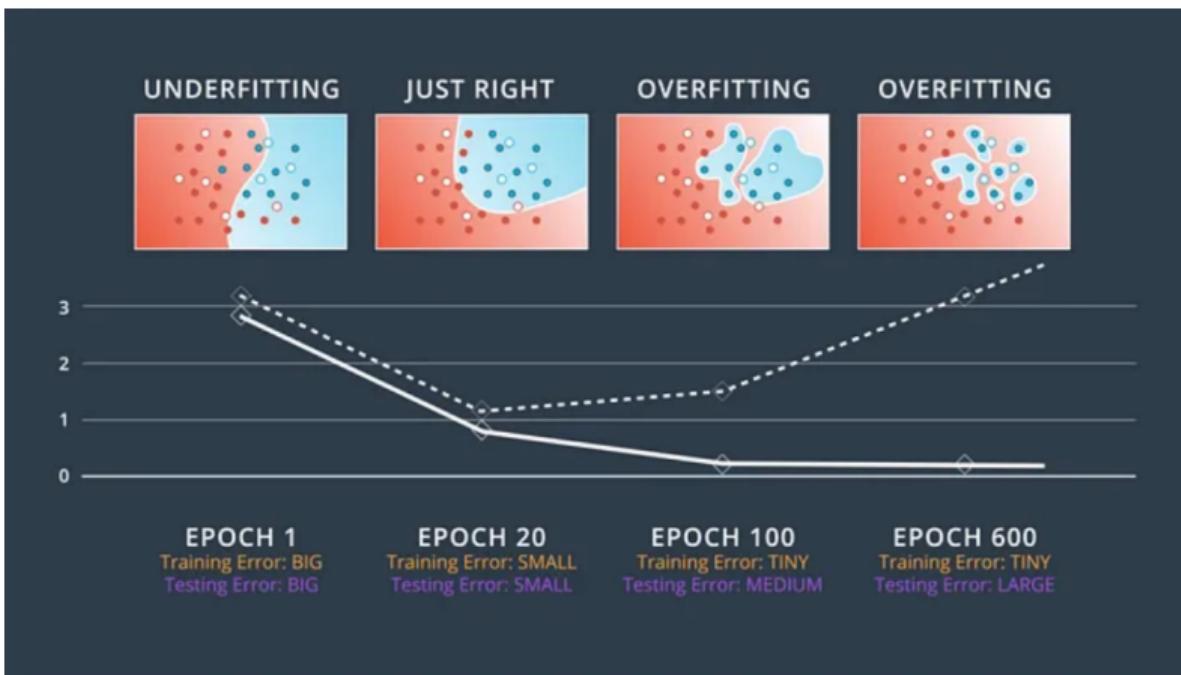
oooo●oooooooo

Optimizers

oooooooooooooooooooo

Resumo

ooooo



ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooo●oooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
ooooo



ConvNets

oooooooooo

Feature Maps

oooooooooooo

Regularization

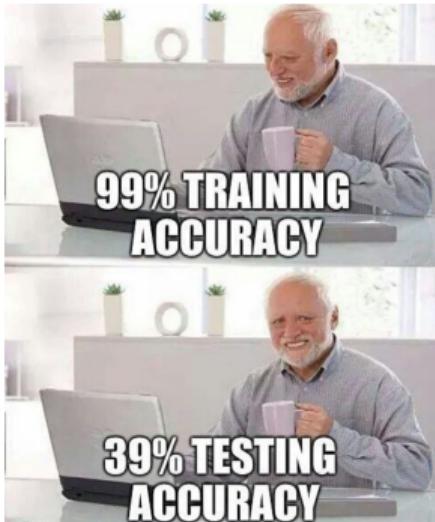
oooooooo●oooooooo

Optimizers

oooooooooooooooooooo

Resumo

ooooo



My model on training data



My model on test dataset



Find the next number of the sequence

1, 3, 5, 7, ?

Correct solution

217341

because when

$$f(x) = \frac{18111}{2} x^4 - 90555 x^3 + \frac{633885}{2} x^2 - 452773 x + 217331$$

f(1)=1

f(2)=3 much solution

f(3)=5 wow very logic

f(4)=7

f(5)=217341

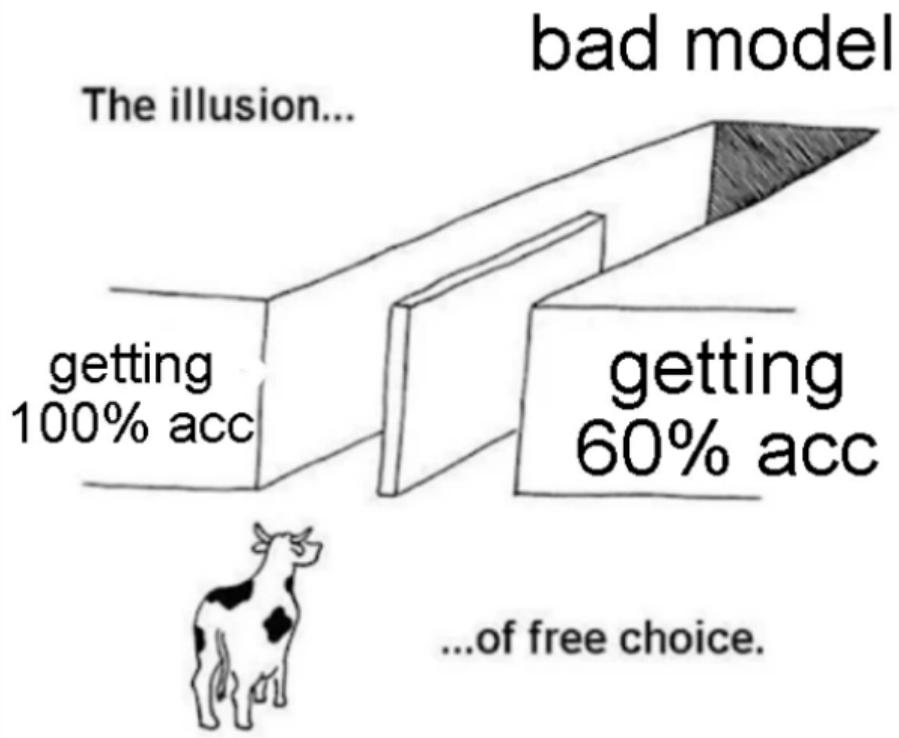
such function

many maths

WOW



UNIVERSIDADE  
FEDERAL DA  
FRONTEIRA SUL



## Regularização

- Termo adicionado na *loss function* para “dificultar” a otimização
- Objetivo: evitar que a rede neural entre em *overfitting*
  - ✓  $\mathcal{L}_1$ : busca tornar a maioria dos parâmetros = 0 usando uma constante

$$\mathbf{C} = -\frac{1}{n} \sum_x [y \log a^L + (1 - y) \log(1 - a^L)] + \frac{\lambda}{n} \sum_w |w|$$

- ✓  $\mathcal{L}_2$ : busca tornar os parâmetros próximos de 0 usando valores proporcionais aos  $w$

$$\mathbf{C} = -\frac{1}{n} \sum_x [y \log a^L + (1 - y) \log(1 - a^L)] + \frac{\lambda}{n} \sum_w |w|$$

- ✓ *ElasticNet* ( $\mathcal{L}_1 + \mathcal{L}_2$ )

## Regularização

- Termo adicionado na *loss function* para “dificultar” a otimização
- Objetivo: evitar que a rede neural entre em *overfitting*
  - ✓  $\mathcal{L}_1$ : busca tornar a maioria dos parâmetros = 0 usando uma constante
  - ✓  $\mathcal{L}_2$ : busca tornar os parâmetros próximos de 0 usando valores proporcionais aos  $w$
  - ✓ *ElasticNet* ( $\mathcal{L}_1 + \mathcal{L}_2$ )

Em  $\mathcal{L}_1$ , adicionamos uma média dos valores absolutos dos parâmetros  $W$  multiplicada por um parâmetro  $\lambda$  que controla o impacto da regularização.

$$\mathbf{C} = -\frac{1}{n} \sum_x [y \log a^L + (1 - y) \log(1 - a^L)] + \frac{\lambda}{n} \sum_w |w|$$

$$\mathbf{C} = -\frac{1}{n} \sum_x [y \log a^L + (1 - y) \log(1 - a^L)] + \frac{\lambda}{n} \sum_w w^2$$

## Regularização

- Termo adicionado na *loss function* para “dificultar” a otimização
- Objetivo: evitar que a rede neural entre em *overfitting*
  - ✓  $\mathcal{L}_1$ : busca tornar a maioria dos parâmetros = 0 usando uma constante

$$\mathbf{C} = -\frac{1}{n} \sum_x [y \log a^L + (1 - y) \log(1 - a^L)] + \frac{\lambda}{n} \sum_w |w|$$

- ✓  $\mathcal{L}_2$ : busca tornar os parâmetros próximos de 0 usando valores proporcionais aos  $w$

$$\mathbf{C} = -\frac{1}{n} \sum_x [y \log a^L + (1 - y) \log(1 - a^L)] + \frac{\lambda}{n} \sum_w |w|^2$$

- ✓ *ElasticNet* ( $\mathcal{L}_1 + \mathcal{L}_2$ )

Em  $\mathcal{L}_2$ , adicionamos uma média dos valores dos parâmetros  $\mathbf{W}$  ao quadrado, multiplicada por um parâmetro  $\lambda$  que controla o impacto da regularização.

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo●oo

Optimizers  
oooooooooooooooooooo

Resumo  
ooooo

## Dropout

- *Dropout*: remove neurônios aleatoriamente durante o *forward pass* e os coloca de volta durante o passo de *backpropagation*

ConvNets  
oooooooooo

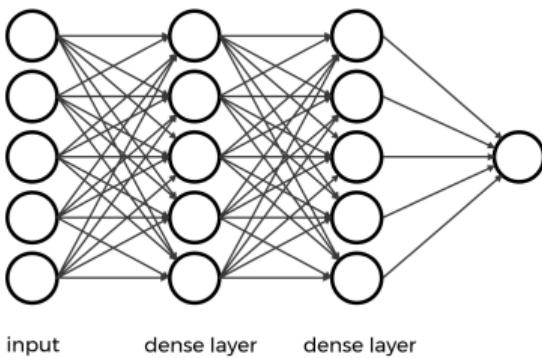
Feature Maps  
oooooooooooo

Regularization  
oooooooooooo●○

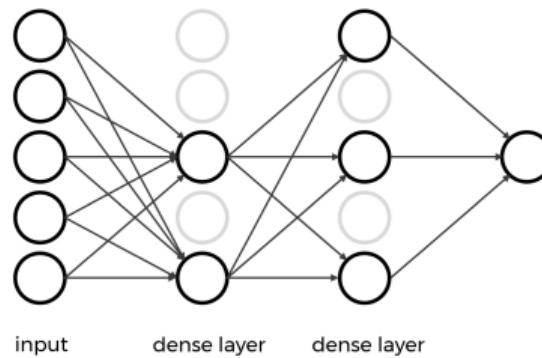
Optimizers  
oooooooooooooooooooo

Resumo  
ooooo

no dropout



with dropout



- Inicialização “Xavier” (ou “Glorot”)

$$weights = \mathcal{N}(0, std)$$

onde:

$$std = gain \times \sqrt{\frac{2}{n_{in} + n_{out}}}$$

e normalmente

$$gain = 1$$

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
●oooooooooooo

Resumo  
ooooo

# Métodos adaptativos de Gradiente Descendente

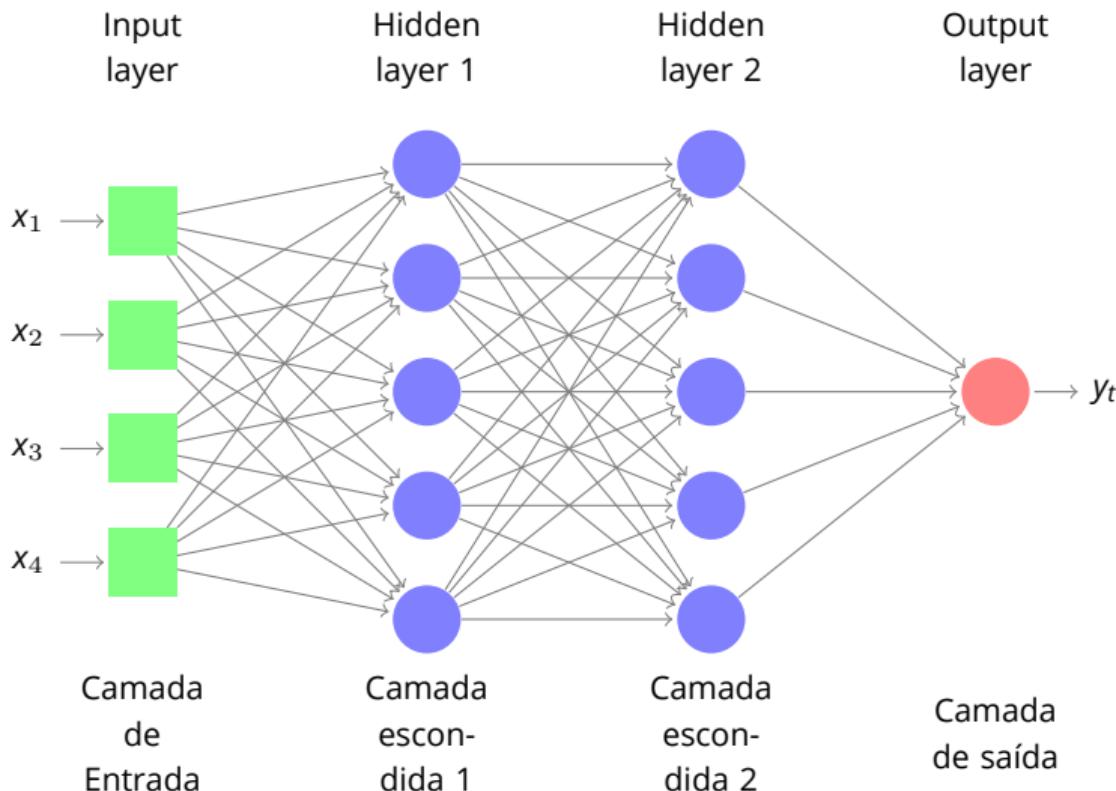
ConvNets  
oooooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
○●oooooooooooo

Resumo  
oooooo



- Supondo uma rede neural com 4 camadas e 1 neurônio em cada camada, a cadeia para se atualizar parâmetros na primeira camada é (usando a atualização de bias como exemplo)

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) w_2 \sigma'(z_2) w_3 \sigma'(z_3) w_4 \sigma'(z_4) \frac{\partial C}{\partial a_4}$$

- A multiplicação por  $\sigma'(z_I)$  pode fazer com que o sinal desapareça quase que por completo!
  - ✓ *Vanishing gradients*
  - ✓ Embora menos frequente, há também o problema de *Exploding gradients*, onde o sinal aumenta rapidamente

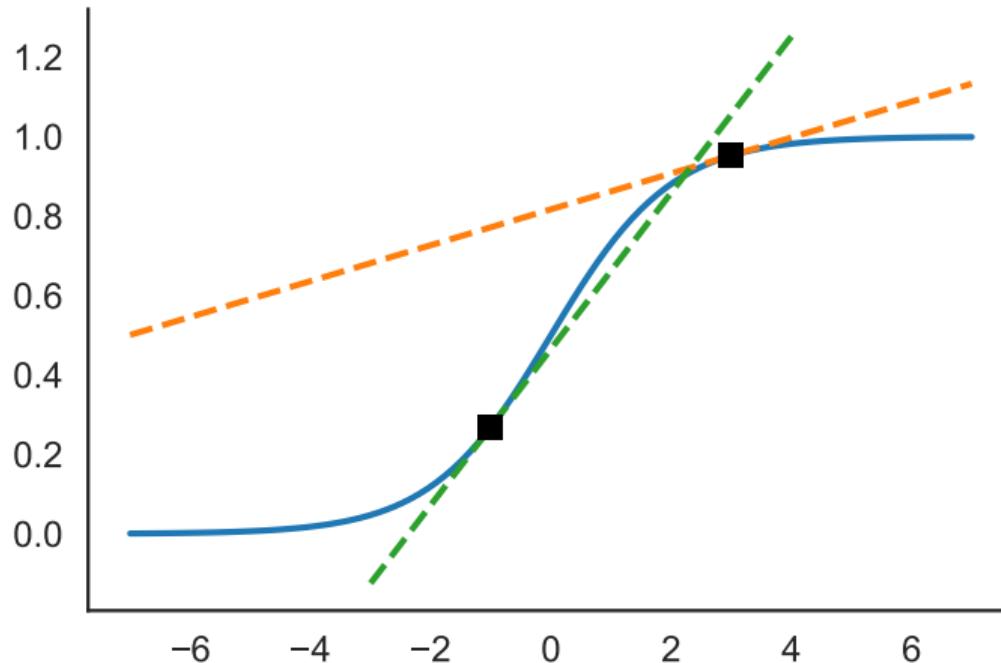
ConvNets  
oooooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
ooo●oooooooooooo

Resumo  
ooooo



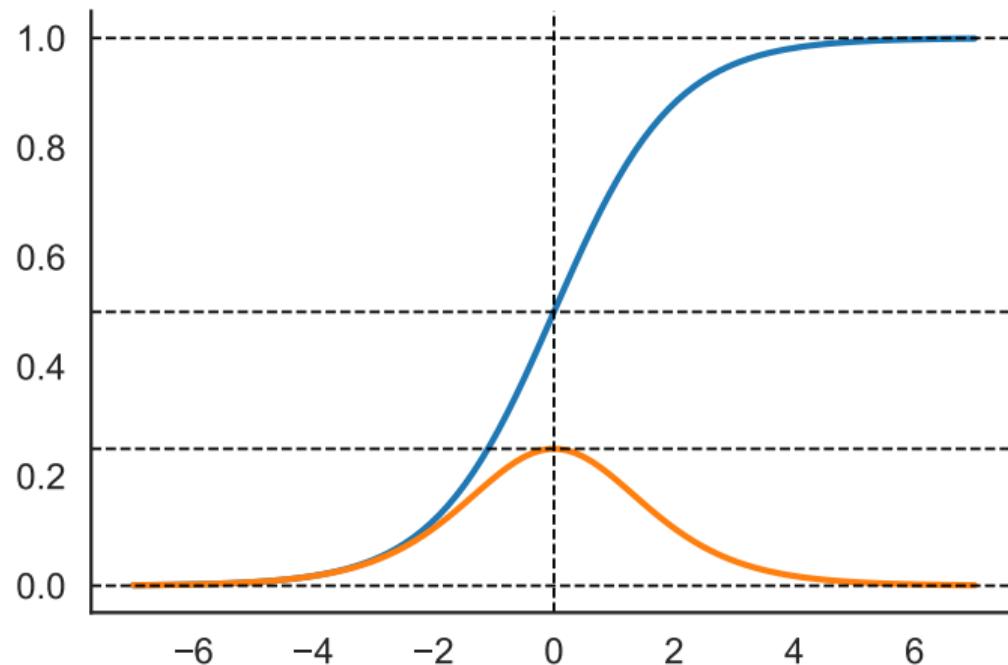
ConvNets  
oooooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooo●oooooooooooo

Resumo  
ooooo



- Embora o método de Gradiente Descendente (SGD) seja o padrão para otimização de redes neurais artificiais, existem outros métodos que estendem sua formulação básica

$$\hat{\theta} = \theta - \alpha \odot \nabla \mathcal{C}$$

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooo●oooooooo

Resumo  
ooooo

- Momentum

$$v_t = \gamma v_{t-1} + \alpha \odot \nabla \mathcal{C}$$

$$\hat{\theta} = \theta - v_t$$

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooo●oooooooo

Resumo  
ooooo

- Nesterov Accelerated Gradient (Nesterov Momentum)

$$v_t = \gamma v_{t-1} + \alpha \odot \nabla \mathcal{C}(\theta - \gamma v_{t-1})$$

$$\hat{\theta} = \theta - v_t$$

- Adaptive Gradient (AdaGrad)

$$\begin{aligned} g_t &= \nabla \mathcal{C} \\ \theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \odot g_t \end{aligned}$$

onde  $G_t$  é uma matriz diagonal que contém os gradientes anteriores (primeiro elevados ao quadrado e depois somados)

- RMSprop

$$\begin{aligned}E[g^2]_t &= \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t\end{aligned}$$

onde  $E[g^2]_t$  é uma matriz diagonal que contém uma “janela” dos gradientes anteriores ao invés de conter todos (mais uma vez, os gradientes são elevados ao quadrado e depois somados)

- Adaptive Learning Rate (AdaDelta)

$$g_t = \nabla \mathcal{C}$$

$$RMS[g]_t = \sqrt{\gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2 + \epsilon}$$

$$RMS[\Delta\theta]_t = \sqrt{\gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma) \Delta\theta_t^2 + \epsilon}$$

$$\Delta\theta_t = - \frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t$$

$RMS[g]_t$  é semelhante ao  $E[g^2]_t$  definido para o RMSProp; e  $RMS[\Delta\theta]_t$  é definida da mesma forma que  $RMS[g]_t$  mas usando os valores anteriores de  $\theta$

- Adaptive Momentum Estimate (Adam)

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

ConvNets

oooooooooo

Feature Maps

oooooooooooo

Regularization

oooooooooooo

Optimizers

oooooooooooo●oo

Resumo

ooooo

ConvNets

oooooooooo

Feature Maps

oooooooooooo

Regularization

oooooooooooo

Optimizers

oooooooooooo●○

Resumo

ooooo

## ■ Recomendações

- ✓ Se os dados forem esparsos (muitos 0s), Adadelta, RMSprop e Adam são as melhores escolhas
- ✓ De maneira geral, SGD provê os melhores resultados
- ✓ contudo, o SGD precisa de mais iterações para atingir tal efeito
- ✓ SGD é menos resistente a inicialização de parâmetros e pode ficar “preso” em “*saddle points*”

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooo

Resumo  
●oooo

# Resumo



A. facke  
@twitter\_account

...

"UsE DroPPoUT aNd VaL/TrAIIn SpIIT tO  
AvOld oVeRFittInG" Bro my model  
barely works on the training data...

2:41 PM · Aug 30, 2022 · Twitter Web App

96 Retweets 88 Quote Tweets 153 Likes



## Training Set



## Validation Set



## Test Set



ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooo

Resumo  
oo●oo

- Redes neurais profundas são os modelos mais avançados de IA no momento
- Ainda é difícil treiná-las e obter bons resultados
- No entanto é fácil encontrar modelos pré-treinados para adaptar à outras necessidades

- Entender os métodos mais recentes que foram utilizados para criá-las facilita o processo de adaptação
- No entanto, o entendimento de como as redes neurais funcionam ainda está na fase da experimentação
  - ✓ Ainda falta uma teoria que explique seu funcionamento e nos guie na criação/otimização de modelos

ConvNets  
oooooooooo

Feature Maps  
oooooooooooo

Regularization  
oooooooooooo

Optimizers  
oooooooooooooooooooo

Resumo  
oooo●

## Redes Neurais Convolucionais

## Representações Distribuídas

## Regularização

## Métodos adaptativos de Gradiente Descendente

## Resumo