Introdução a Redes Neurais

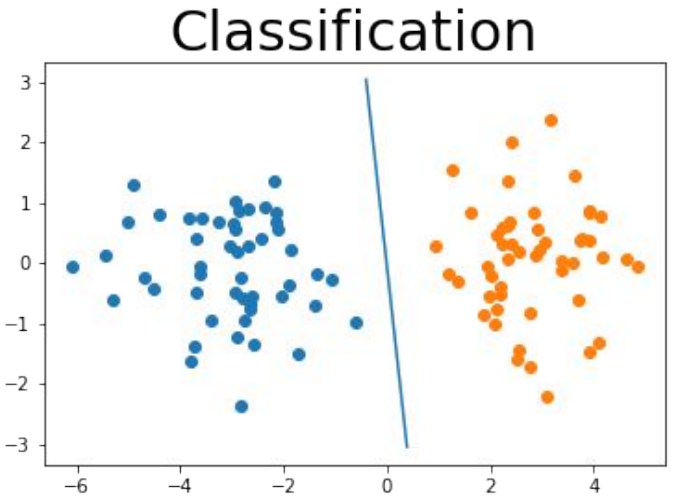
Arquiteturas MLP e CNN

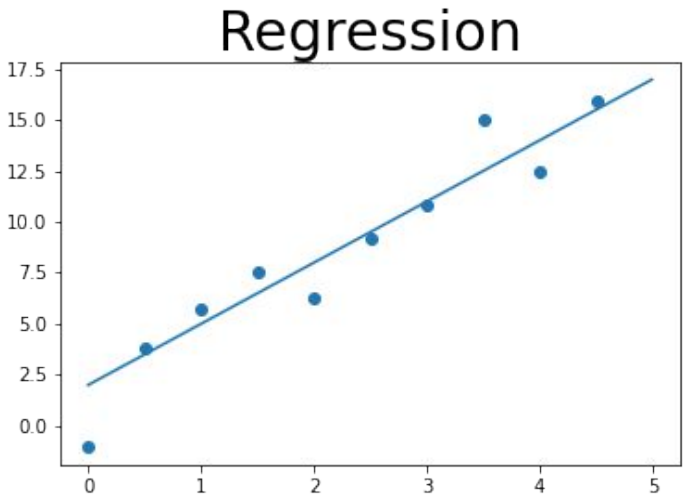
Joris GUERIN

Concurso para Professor Adjunto em Aprendizado de Máquina
Departamento de Informática e Matemática Aplicada (DIMAP)
Universidade Federal do Rio Grande do Norte (UFRN)

Pré-requisitos

Problemas de aprendizagem





Pré-requisitos

Funções de perda

Classificação: **Cross-Entropy**

$$L_{CE}(F, X, y) = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(F(x_i))$$

Regressão: **Mean Squared Error**

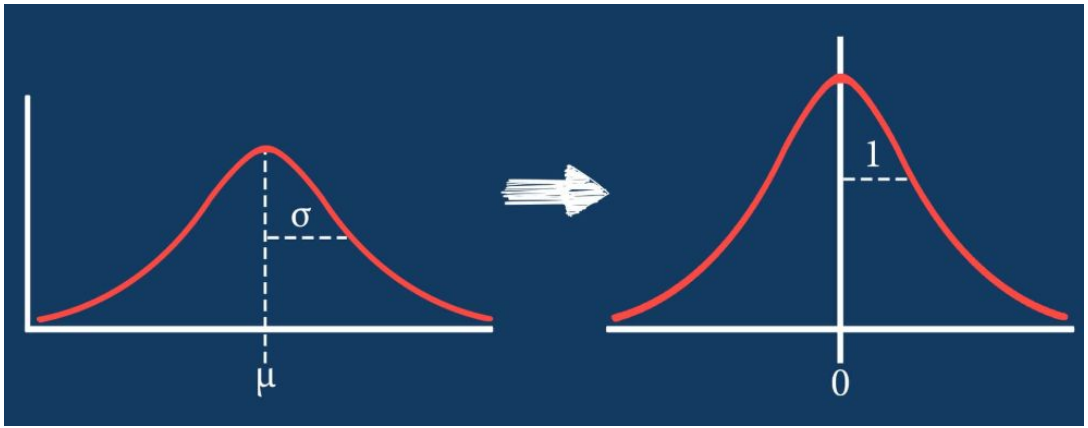
$$L_{MSE}(F, X, y) = -\frac{1}{N} \sum_{i=1}^{N} ||y_i - F(x_i)||^2$$

Pré-requisitos

Modelos lineares

$$F(x) = \omega^T x + b$$

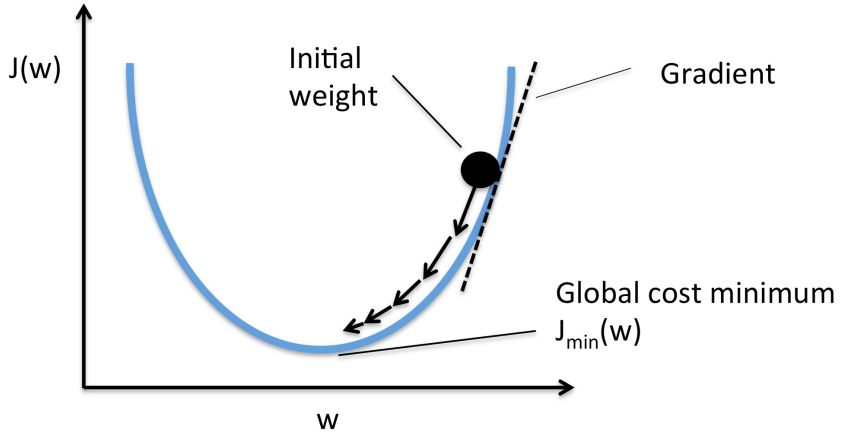
Pré-processamento de dados: **padronização**



Source: *https://towardsdatascience.com/normalization-vs-standardization-quantitative-analysis-a91e8a79cebf?gi=b399a3728261*

Pré-requisitos

Treinamento: Otimização iterativa - **Gradiente Descendente**



Source: *http://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/*

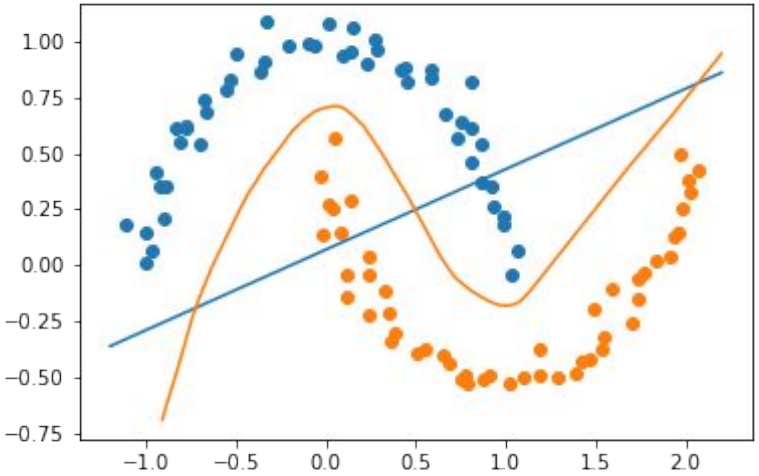
Avaliação de classificadores: **Acurácia** $\frac{N_{correct}}{N_{total}}$

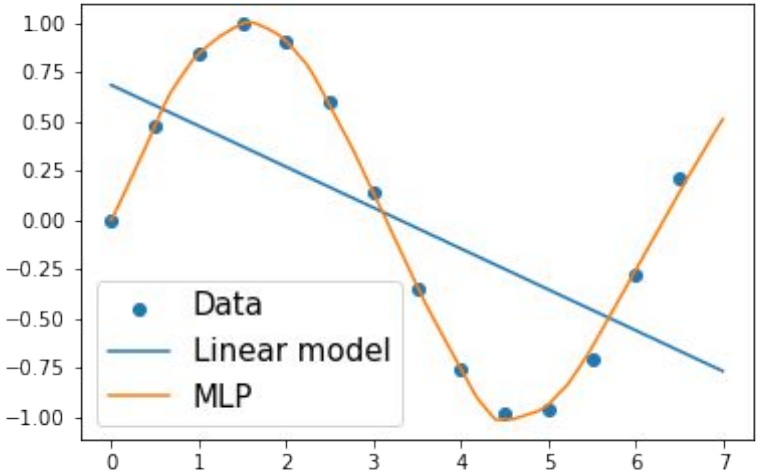
Multi-Layer Perceptron

Uma primeira arquitetura de rede neural artificial

Motivações

Modelização de funções não-lineares





Motivações

Por que MLP entre outros modelos não-lineares?

Teoria: Teorema da aproximação universal

Prática: Permite resolver problemas muito complexos

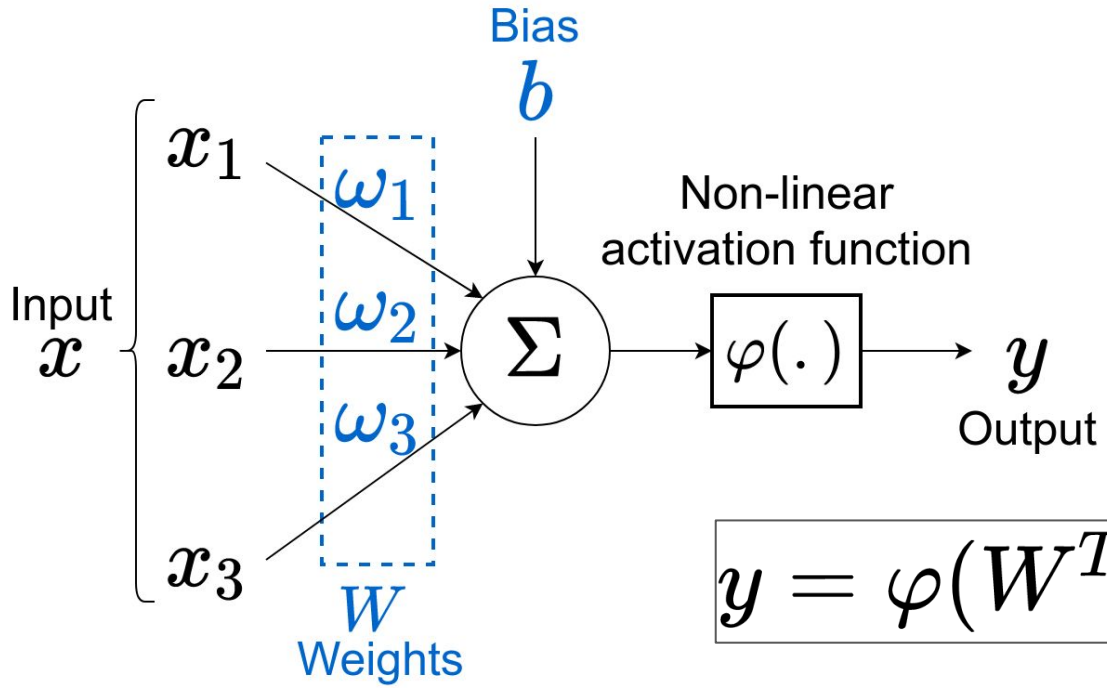
Implementação: Combinação de funções simples

→ Fácil de usar o gradiente descendente

→ Fácil de otimizar o treinamento rápido com GPU

Arquitetura MLP

Modelo de neurônio simples



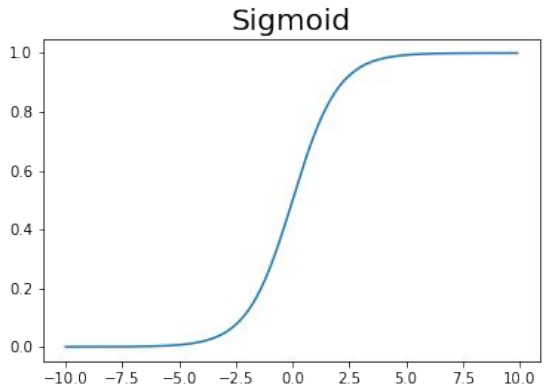
$$y = \varphi(W^T x + b)$$

Arquitetura MLP

Funções de ativação comuns

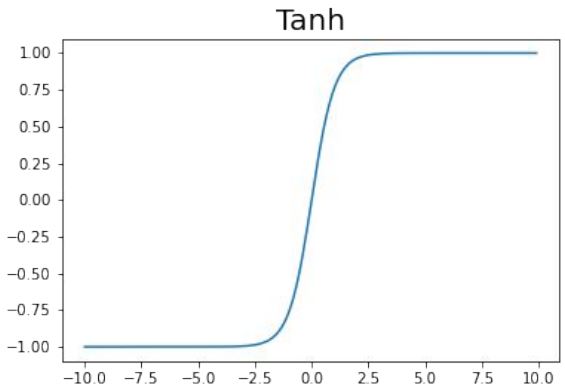
Sigmoid

$$\varphi(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



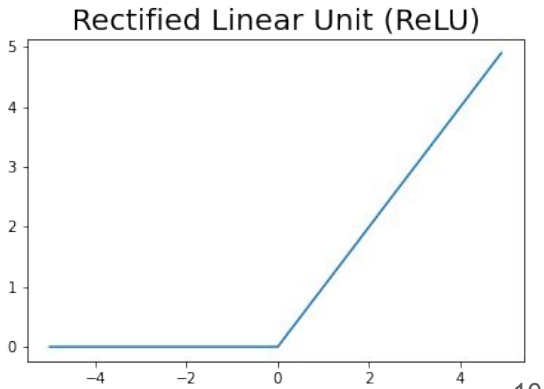
Tanh

$$\varphi(x) = 1 - \frac{2}{1+e^{2x}}$$



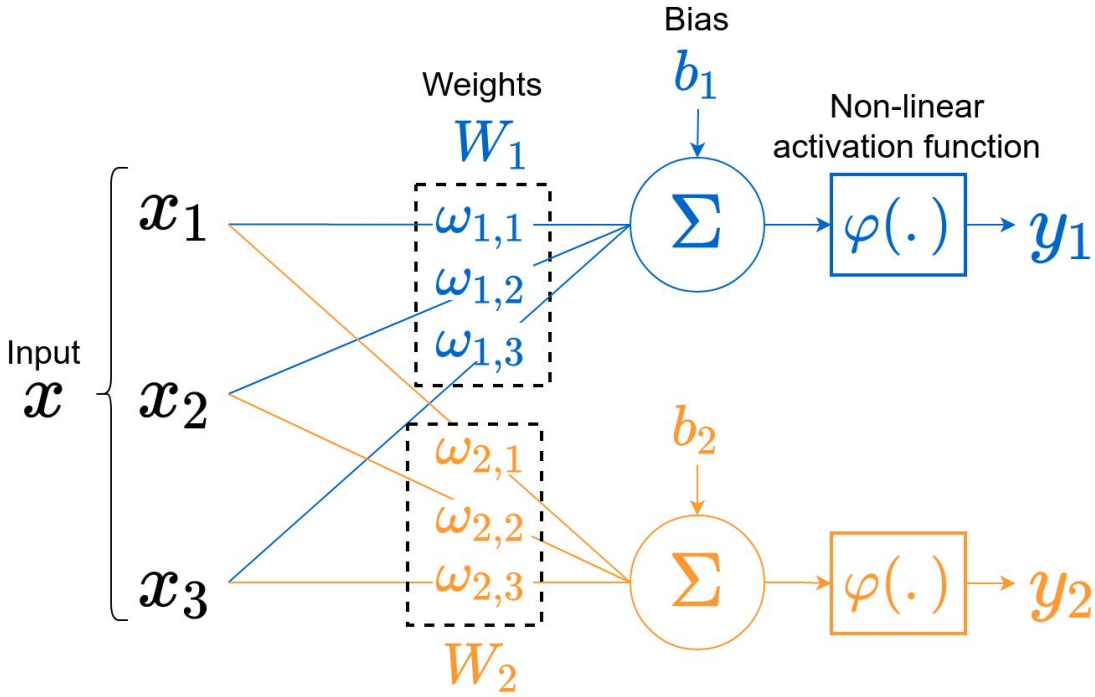
ReLU

$$\varphi(x) = \max(0, x)$$



Arquitetura MLP

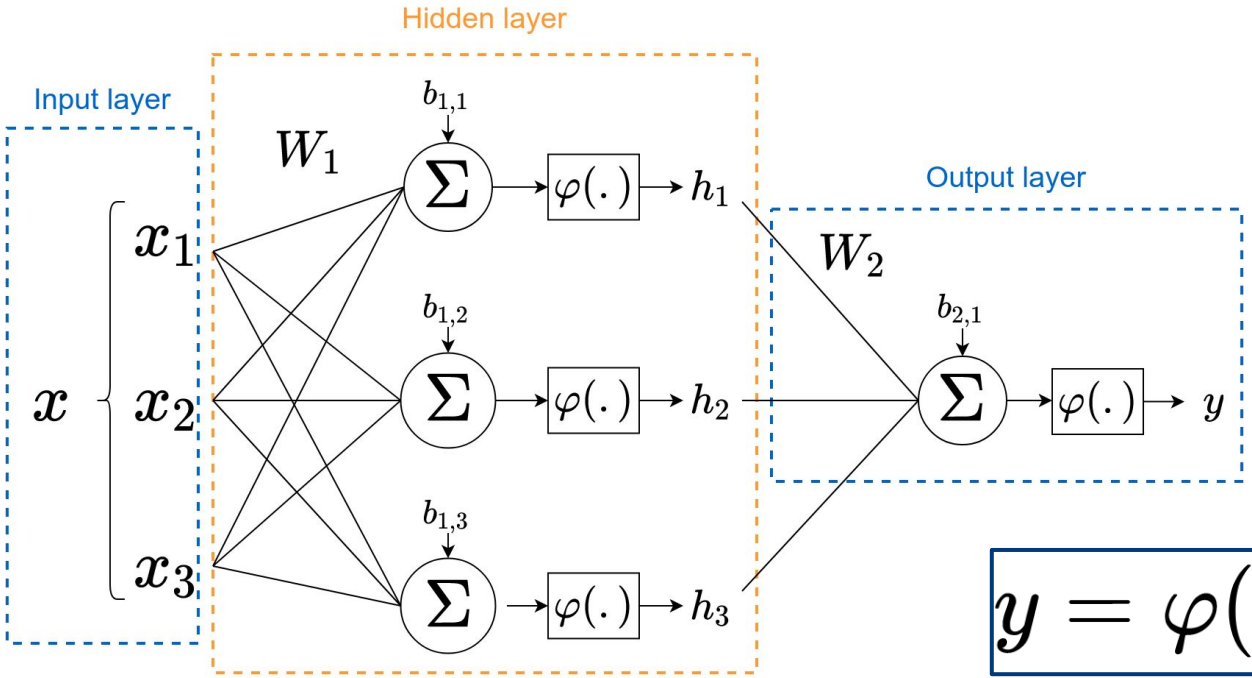
Modelo de camada de neurônios (*layer*)



$$\forall i \in \{1, 2\},$$
$$y_i = \varphi(W_i^T x + b_i)$$

Arquitetura MLP

Modelo de Multi-Layer Perceptron (MLP)



$$y = \varphi(W_2^T h + b_{2,1})$$

Restrições sobre a arquitetura

O tamanho da **camada de entrada** é a dimensão dos dados

O tamanho da **camada de saída** depende dos rótulos

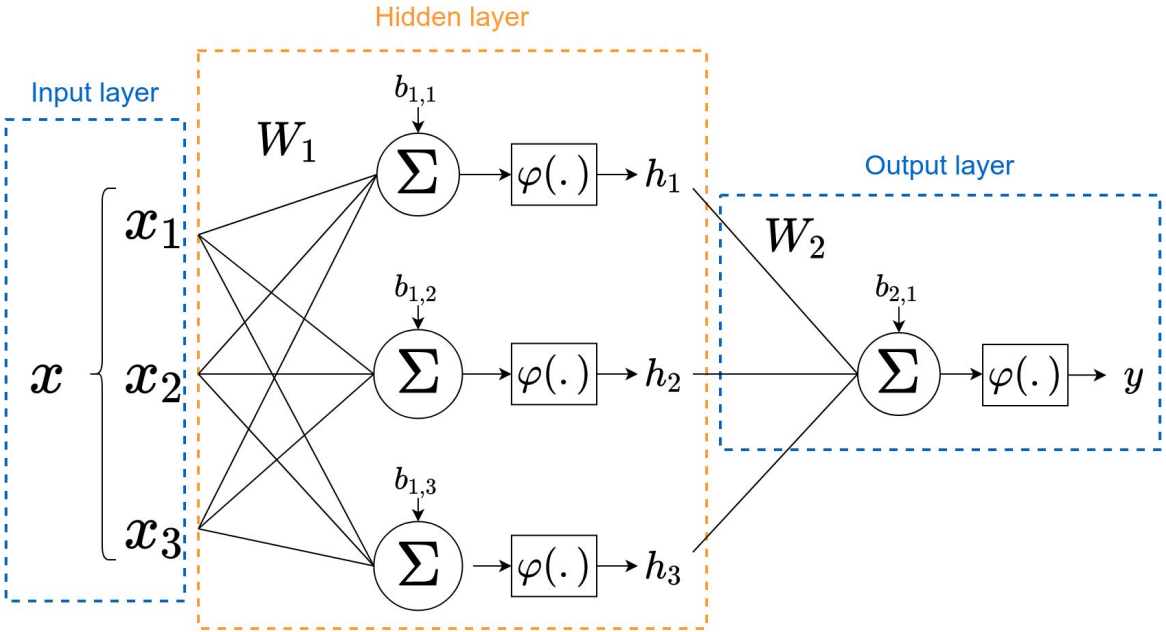
- Classificação: N neurônios de saída = N categorias
- Regressão: N neurônios de saída = N valores a estimar (geralmente 1)

A **função de ativação** da camada de saída depende da faixa de valores

- Exemplo: regressão de variáveis resposta num intervalo fixo: Sigmoid

O **número de parâmetros** não deve ser muito maior do que o número de amostras → Regra empírica

Calcular número de parâmetros



$$N_{i,j}^{\text{params}} = d_i^{\text{in}} + 1$$

$$N_i^{\text{params}} = \sum_j N_{i,j}^{\text{params}} = N_i^{\text{neurons}} \times (d_i^{\text{in}} + 1)$$

$$N^{\text{params}} = \sum_i N_i^{\text{params}}$$

$$[3 \times (3+1)] + [1 \times (3+1)] = 16$$

Regra empírica para construção de arquitetura MLP

Começar com **2 hidden layers**

Em cada camada: usar **potências de 2** para números de neurônios

(2, 4, 8, 16, 32, etc.)

- Primeira camada ~ **metade do número de features**
- Camadas seguintes: **metade da camada anterior**

Este esquema dá uma boa arquitetura para começar o processo de treinamento, mas geralmente o desenvolvimento de uma rede neural é um **processo iterativo** onde se precisa testar muitas configurações.

Exemplo

Treinamento de um MLP para o dataset **Wine**

13 FEATURES *Propriedades químicas do vinho*
Alcohol
Malic acid
Ash
Alkalinity of ash
Magnesium
Total phenols
Etc.

178 AMOSTRAS

Diferentes produtores da mesma região na Itália

3 CATEGORIAS

Diferentes espécies de uva usadas

Exemplo

178 Amostras
13 Features
3 Categorias

- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Input layer

Tamanho
?

- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Input layer

Tamanho

13

Output layer

Tamanho

?

- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Input layer

Tamanho
13

Output layer

Tamanho
3
Ativação
?

- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Número de camadas escondidas: ?

Input layer

Tamanho
13

Output layer

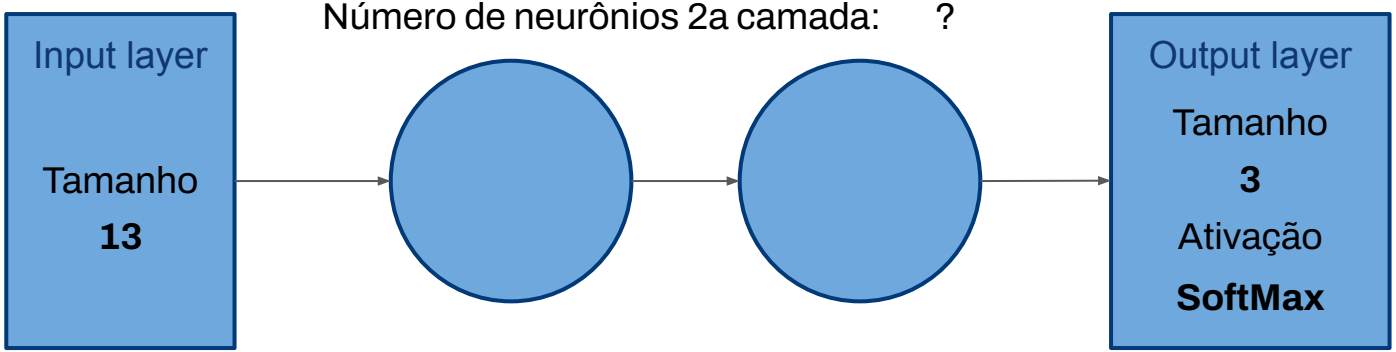
Tamanho
3
Ativação
SoftMax

- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Número de camadas escondidas: 2
Número de neurônios 1a camada: ?
Número de neurônios 2a camada: ?

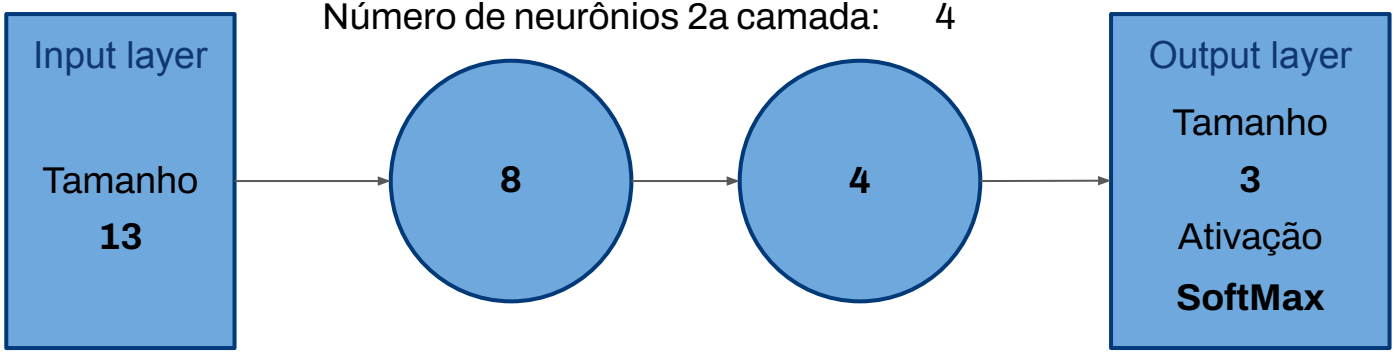


- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Número de camadas escondidas: 2
Número de neurônios 1a camada: 8
Número de neurônios 2a camada: 4

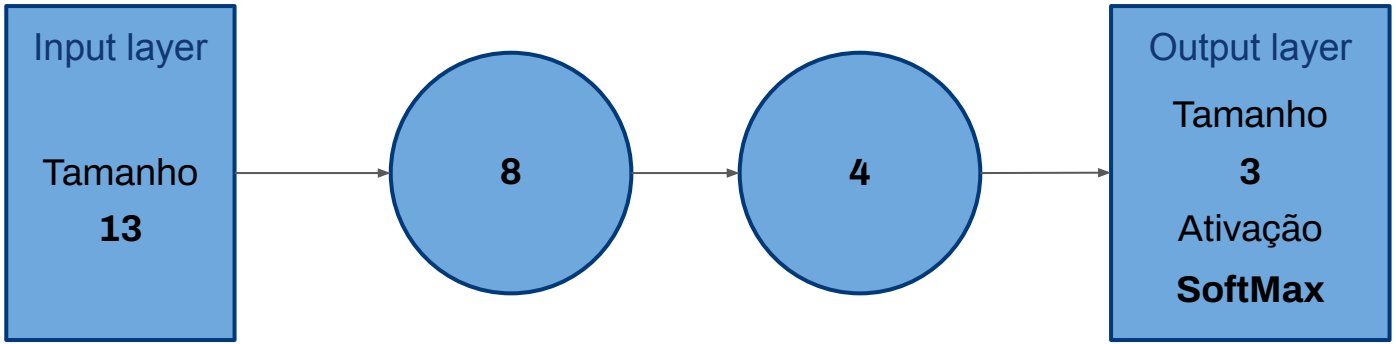


- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Número de parâmetros: ?

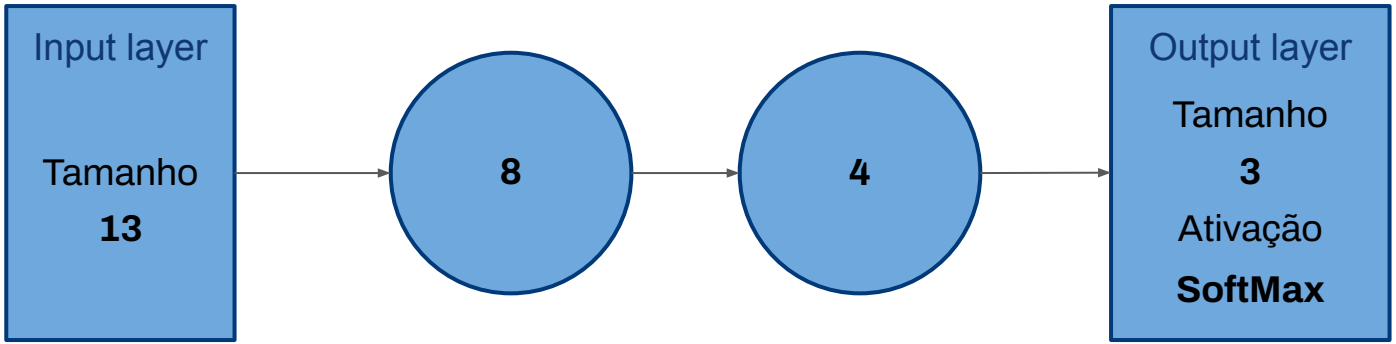


- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo

178 Amostras
13 Features
3 Categorias

Número de parâmetros: 8x(13+1) + 4 x (8+1)
= **148**



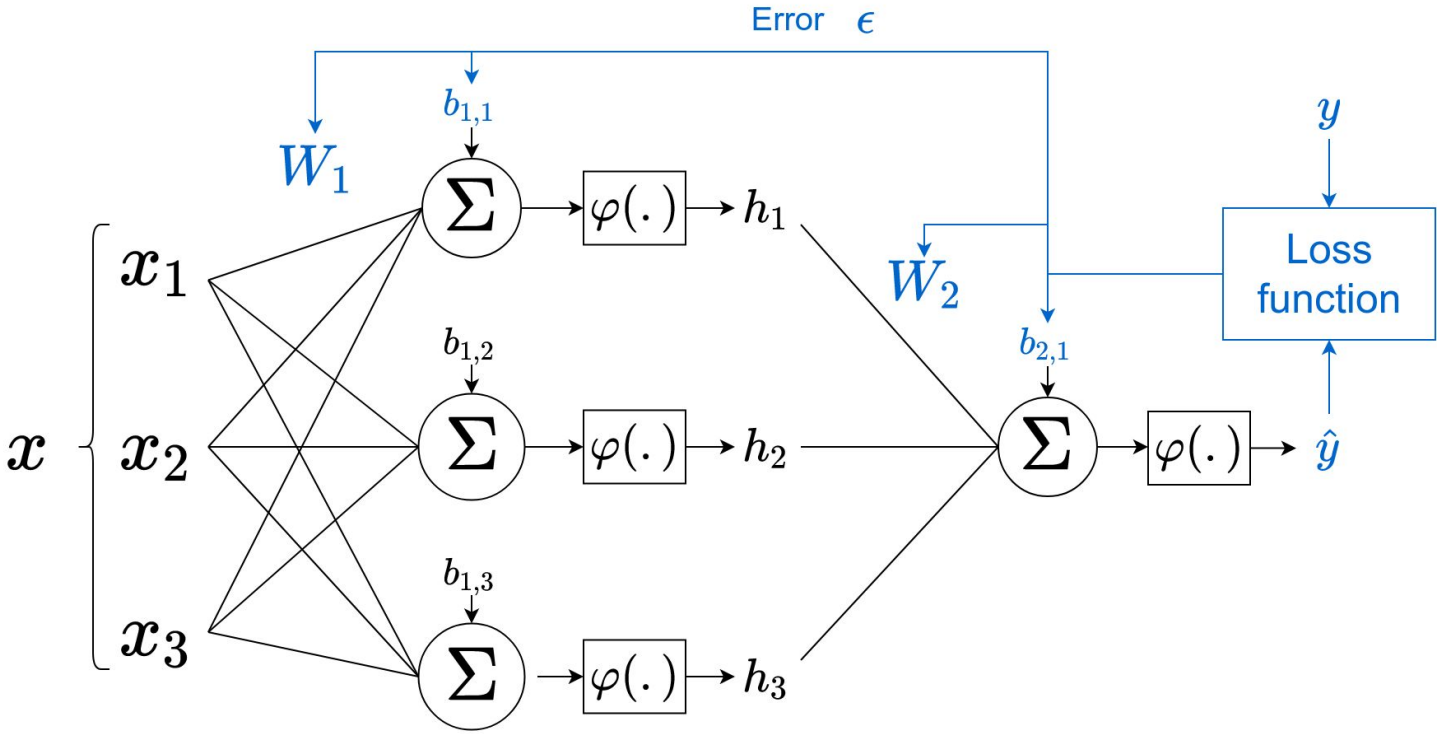
- Começar com 2 hidden layers
- Em cada camada: usar potências de 2 para números de neurônios (2, 4, 8, 16, 32, etc.)
- Primeira camada ~ a metade do número de features
- Camadas seguintes: metade da camada anterior

Exemplo de implementação


```python
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Load data
data = load_wine()
X = data.data
y = data.target

# Split into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Preprocess data
scaler  = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test  = scaler.transform(X_test)

# Build MLP
mlp = MLPClassifier(hidden_layer_sizes=[8, 4], max_iter=10000)
# Train MLP
mlp.fit(X_train, y_train)
# Predict on test set
y_pred = mlp.predict(X_test)

# Compute accuracy score
print("Accuracy MLP: ", accuracy_score(y_pred, y_test))
```

```
Accuracy MLP:  1.0
```

Descrição básica do processo de treinamento



Conclusão intermediária

- Apresentamos a arquitetura MLP e mostramos como pode ser desenvolvida

- Muitos elementos importantes sobre treinamento de redes neurais não foram discutidos ainda (próximas aulas)

Inicialização Otimização Regularização Avaliação

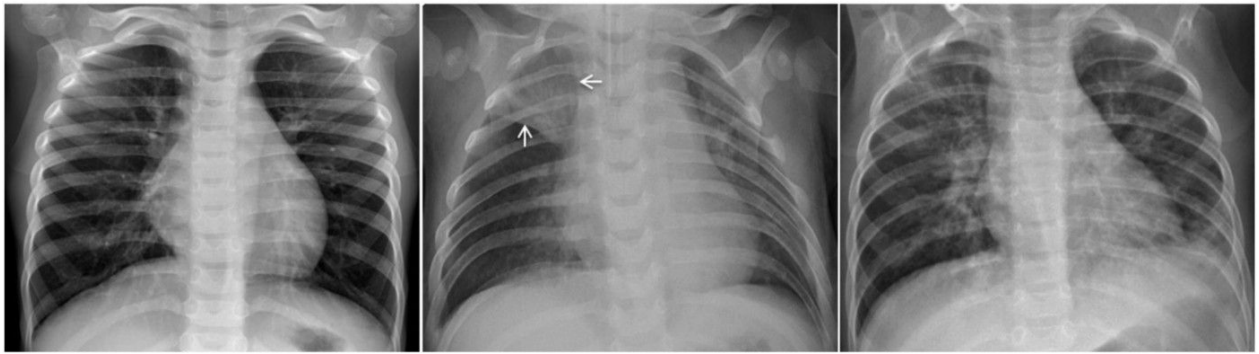
Implementação eficiente Implantação industrial

Convolutional Neural Network

Rede neural artificial para processamento de imagens

Motivações: aprendizado para análise de imagens

Classificação de imagens



54640904_133d1a4fd9.jpg

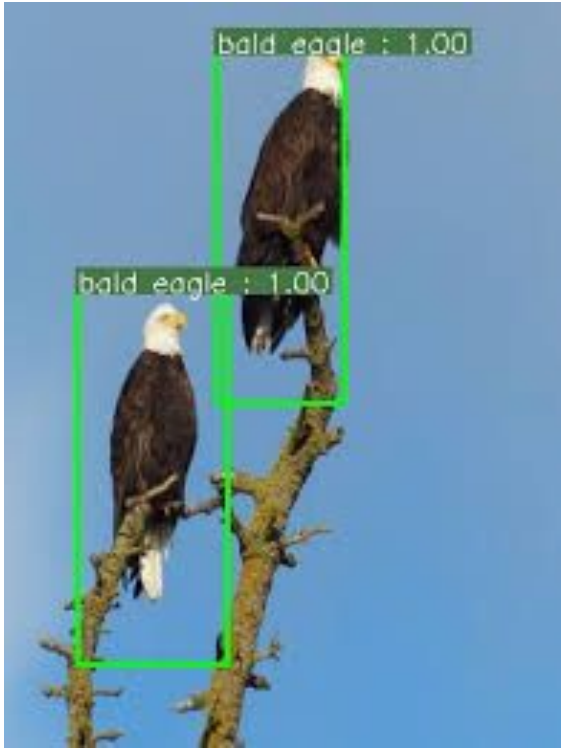
fly agaric	0.999999
common stinkhorn	9.50538e-07
giant puffball	2.51632e-08
earthstar	1.56964e-08
scarlet elf cup	3.09601e-09

Evaluation time (1-image): 0.228s

Source 1: *https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0276-2*
Source 2: *http://fungai.org/2017/12/13/train-a-basic-wild-mushroom-classifier/*

Motivações: aprendizado para análise de imagens

Detecção de objetos



Source 1: https://wider-challenge.org/2019.html
Source 2: https://www.codeproject.com/Articles/1227156/A-Closer-Look-at-Object-Detection-Recognition-and

Motivações: aprendizado para análise de imagens

Super resolution & Image denoising

Source 1: https://80.lv/articles/google-s-new-approach-to-image-super-resolution/
Source 2: http://people.tuebingen.mpg.de/burger/neural_denoising/

Motivações: aprendizado para análise de imagens

Pose recognition

Source: *https://www.analyticsvidhya.com/blog/2021/05/pose-estimation-using-opencv/*

Motivações: aprendizado para análise de imagens

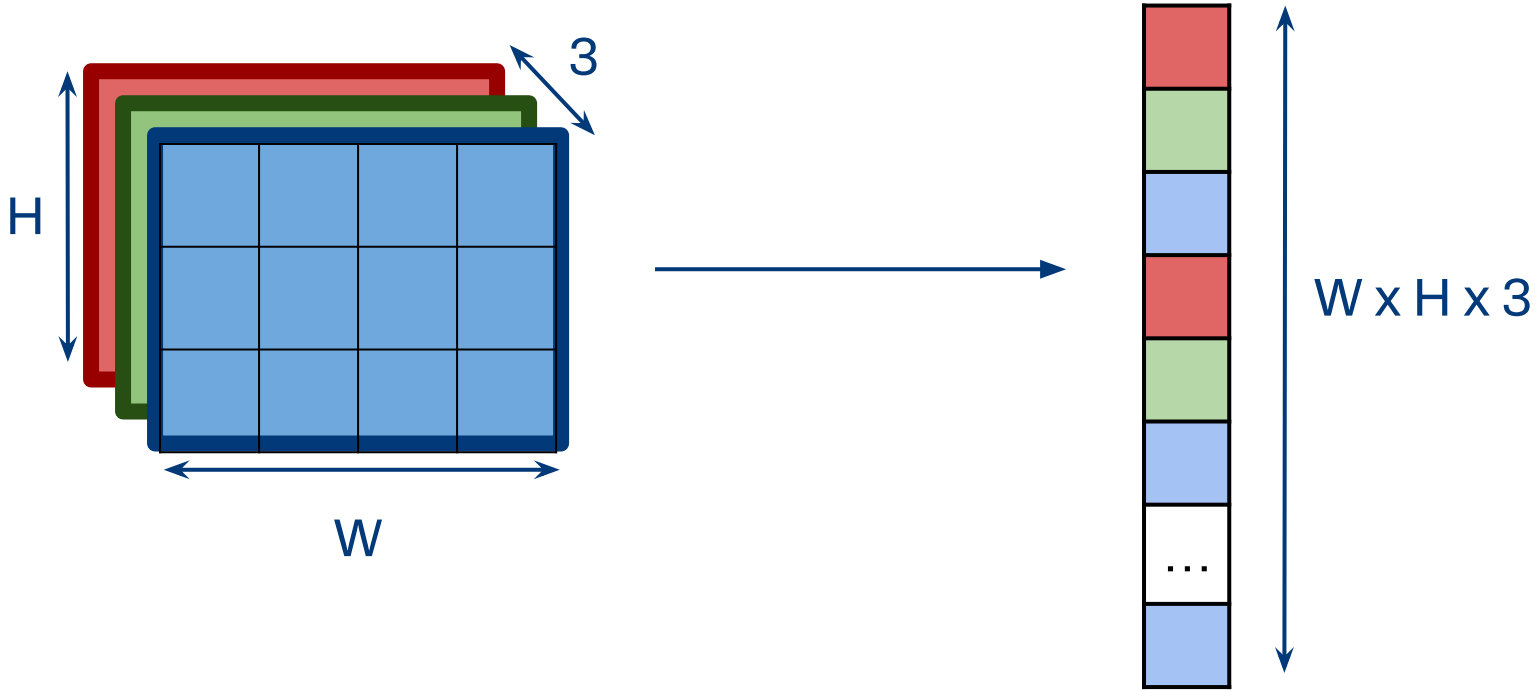
Geração de imagens & Geração de arte

Source 1: https://www.updateordie.com/2019/02/18/estas-pessoas-nao-existem-foram-criadas-por-inteligencia-artificial/
Source 2: https://www.researchgate.net/publication/330155023_CAN_MACHINES_PAINT

Motivações: problemas com MLP para imagens

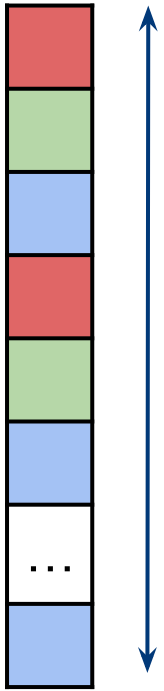
Classificar imagens usando MLP

Uma imagem pode ser representada por um tensor (Matrice 3D) de dimensão WxHx3



Motivações: problemas com MLP para imagens

- Explosão do número de parâmetros necessários



Imagem

300 x 300 x 3
= 270K inputs

MLP com uma camada de 10K neurônios

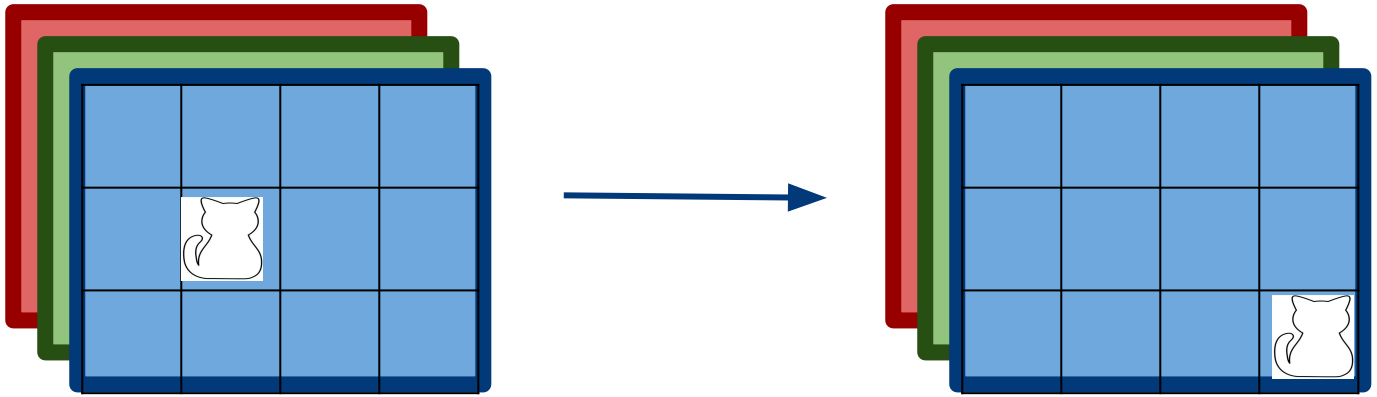
2.7B parâmetros
→ 10.8 GB

Não é suficiente!

Float = 32 bits

Motivações: problemas com MLP para imagens

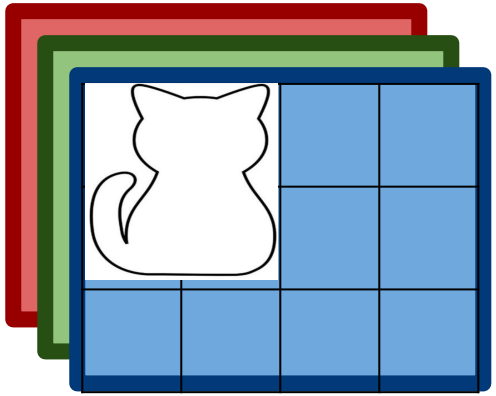
- Número de parâmetros necessários
- **Invariância à translação**

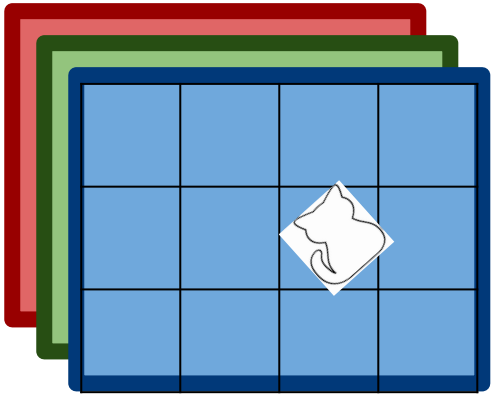


Motivações: problemas com MLP para imagens

- Número de parâmetros
- Invariância à translação
- **Invariância à escala**
- **Invariância à rotação**

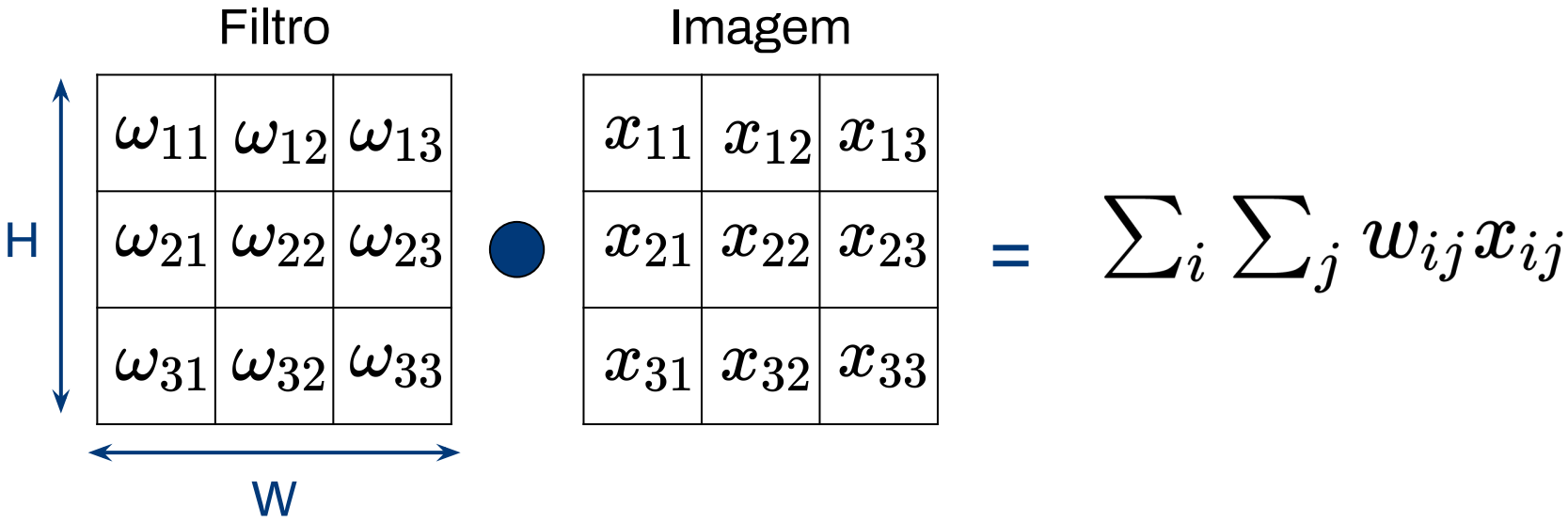
Convolutional Neural Networks (CNN)





Arquitetura CNN

Filtro de convolução e produto escalar de matrizes

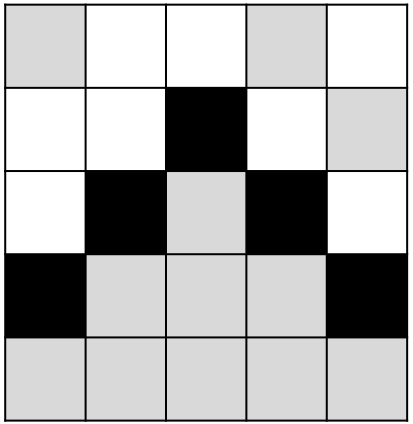


Extensão 3D $\rightarrow \sum_i \sum_j \sum_k w_{ijk} x_{ijk}$

Arquitetura CNN

Exemplo: filtros de convolução para detectar orelha de gato

Imagem em escala de cinza



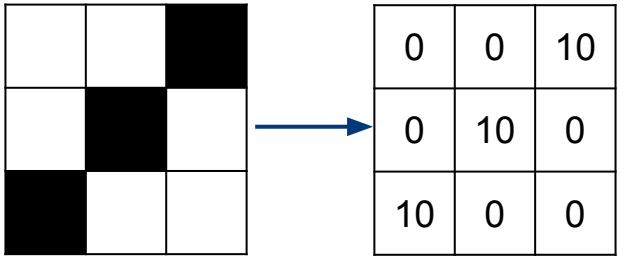
representação numérica

3	0	0	3	0
0	0	10	0	3
0	10	3	10	0
10	3	3	3	10
3	3	3	3	3

Arquitetura CNN

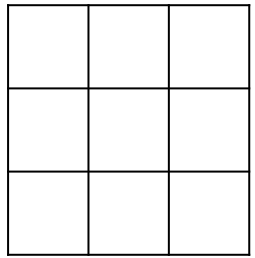
Etapa 1: detecção de bordas inclinadas

Padrão procurado



0	0	10
0	10	0
10	0	0

Filtro de convolução



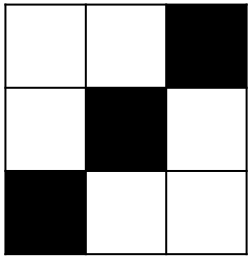
=

Valor alto quando padrão, se não, valor baixo

Arquitetura CNN

Etapa 1: detecção de bordas inclinadas

Padrão procurado

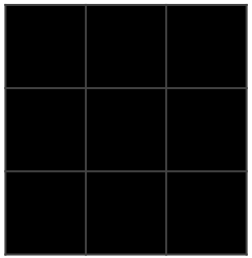


0	0	10
0	10	0
10	0	0

Filtro de convolução

0	0	1
0	1	0
1	0	0

= 30



10	10	10
10	10	10
10	10	10

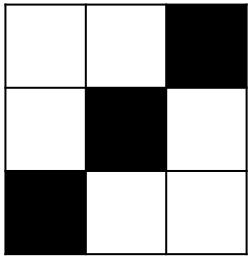
0	0	1
0	1	0
1	0	0

= 30

Arquitetura CNN

Etapa 1: detecção de bordas inclinadas

Padrão procurado

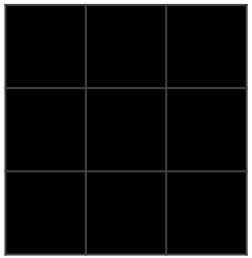


0	0	10
0	10	0
10	0	0

Filtro de convolução

-1	-1	1
-1	1	-1
1	-1	-1

= 30



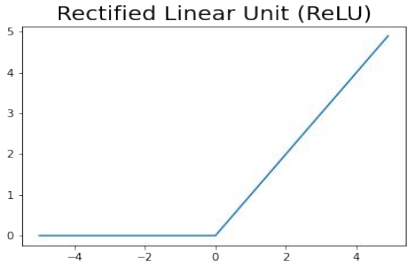
10	10	10
10	10	10
10	10	10

-1	-1	1
-1	1	-1
1	-1	-1

= -30

Arquitetura CNN

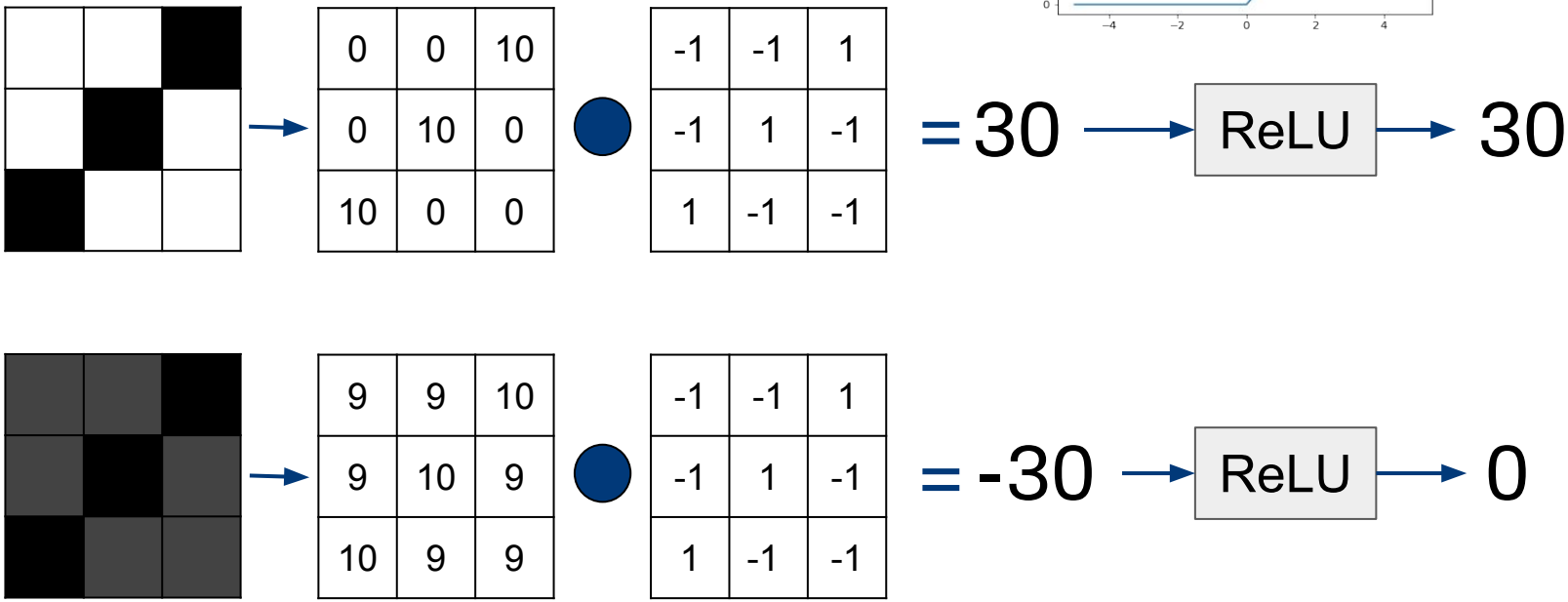
$$\varphi(x) = \max(0, x)$$



Etapa 1: detecção de bordas inclinadas

Padrão procurado

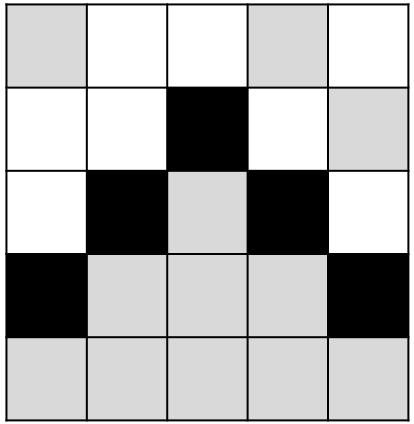
Filtro de convolução



Arquitetura CNN

Aplicar o filtro à imagem toda

Imagem em escala de cinza



Representação numérica

3	0	0	3	0
0	0	10	0	3
0	10	3	10	0
10	3	3	3	10
3	3	3	3	3

Filtro de convolução

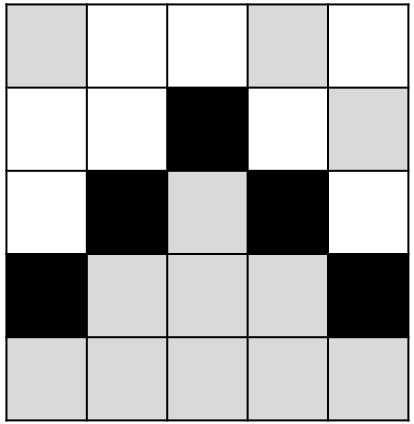
-1	-1	1
-1	1	-1
1	-1	-1

Resultado da convolução

Arquitetura CNN

Aplicar o filtro à imagem toda

Imagem em escala de cinza



Representação numérica

3	0	0	3	0
0	0	10	0	3
0	10	3	10	0
10	3	3	3	10
3	3	3	3	3

Filtro de convolução

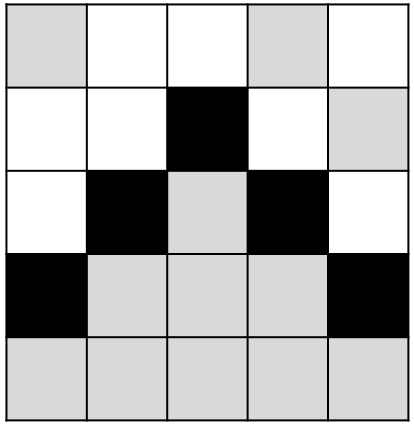
-1	-1	1
-1	1	-1
1	-1	-1

Resultado da convolução

Arquitetura CNN

Aplicar o filtro à imagem toda

Imagem em escala de cinza



Representação numérica

3	0	0	3	0
0	0	10	0	3
0	10	3	10	0
10	3	3	3	10
3	3	3	3	3

Filtro de convolução

-1	-1	1
-1	1	-1
1	-1	-1

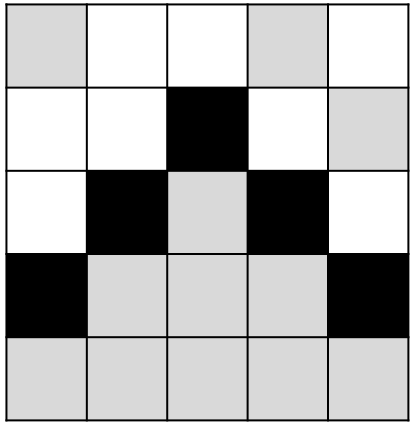
Resultado da convolução

		10		

Arquitetura CNN

Aplicar o filtro à imagem toda

Imagem em escala de cinza



Representação numérica

0	0	0	0	0	0	0
0	3	0	0	3	0	0
0	0	0	10	0	3	0
0	0	10	3	10	0	0
0	10	3	3	3	10	0
0	3	3	3	3	3	0
0	0	0	0	0	0	0

Zero-padding

Filtro de convolução

-1	-1	1
-1	1	-1
1	-1	-1

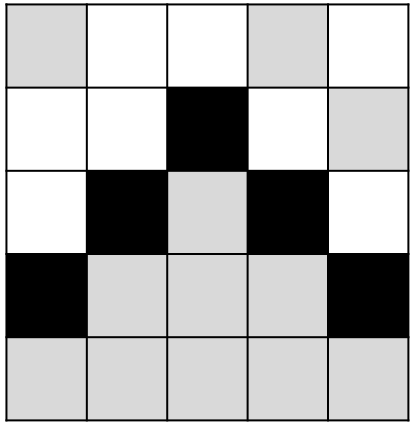
Resultado da convolução

3				
		10		

Arquitetura CNN

Aplicar o filtro à imagem toda

Imagem em escala de cinza



Representação numérica

0	0	0	0	0	0	0
0	3	0	0	3	0	0
0	0	0	10	0	3	0
0	0	10	3	10	0	0
0	10	3	3	3	10	0
0	3	3	3	3	3	0
0	0	0	0	0	0	0

Filtro de convolução

-1	-1	1
-1	1	-1
1	-1	-1

Resultado da convolução

3	-13	-13	10	-6
-13	-26	10	-23	10
-23	21	-30	-10	-20
11	-20	-9	-26	-3
-7	-13	-6	1	-13

Exercício: Verificar os cálculos

Arquitetura CNN

Aplicar a **não-linearidade**

Resultado da convolução

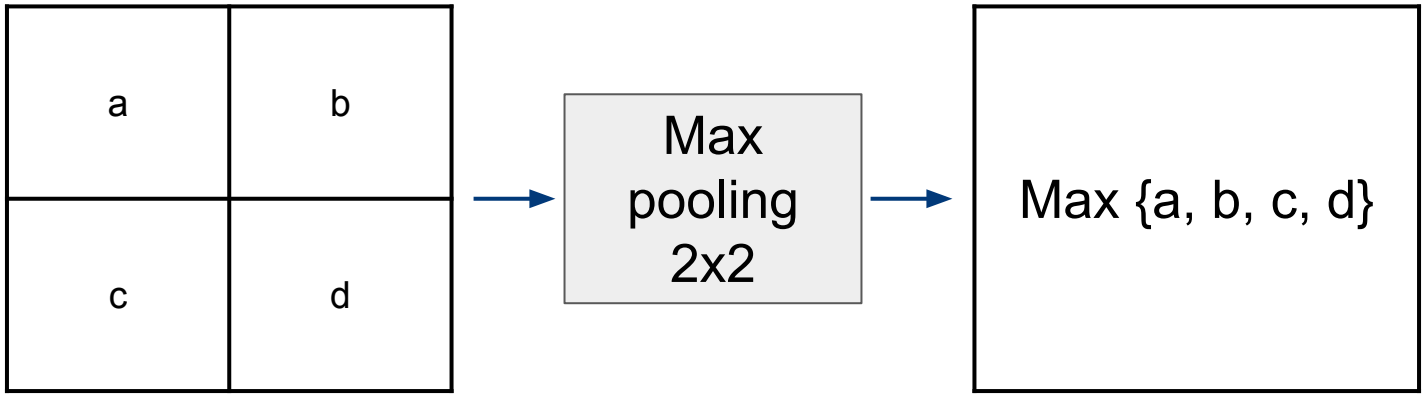
3	-13	-13	10	-6
-13	-26	10	-23	10
-23	21	-30	-10	-20
11	-20	-9	-26	-3
-7	-13	-6	1	-13

Saída da camada de convolução

3	0	0	10	0
0	0	10	0	10
0	21	0	0	0
11	0	0	0	0
0	0	0	1	0

Arquitetura CNN

Operação de **Max-pooling**

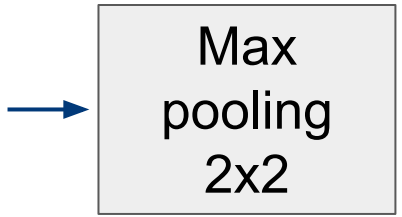


Arquitetura CNN

Operação de **Max-pooling**

Saída da camada de convolução

3	0	0	10	0
0	0	10	0	10
0	21	0	0	0
11	0	0	0	0
0	0	0	1	0

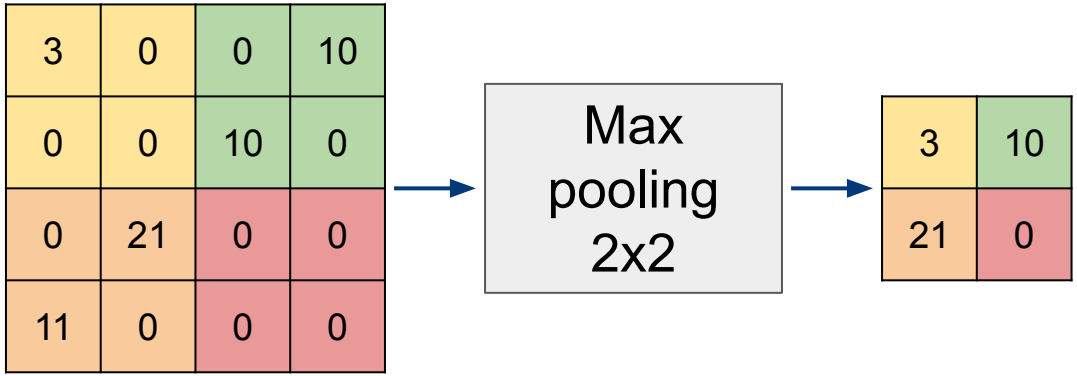


Saída da camada de max pooling

3	10	10
21	0	0
0	1	0

Arquitetura CNN

Por que Max pooling?



- **Número de parâmetros**

  (Tamanho da imagem divido por 4)
- **Invariância à translação**

  (Ativação independente da posição exata)
- Invariância à escala
- Invariância à rotação

Arquitetura CNN

Detecção de orelha de gato

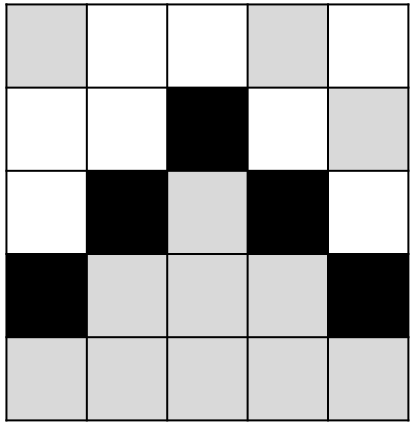
Filtro 1

-1	-1	1
-1	1	-1
1	-1	-1

Filtro 2

1	-1	-1
-1	1	-1
-1	-1	1

Imagem



Representação numérica

3	0	0	3	0
0	0	10	0	3
0	10	3	10	0
10	3	3	3	10
3	3	3	3	3

Nova imagem

Canal 1

3	10	10
21	0	0
0	1	0

Canal 2

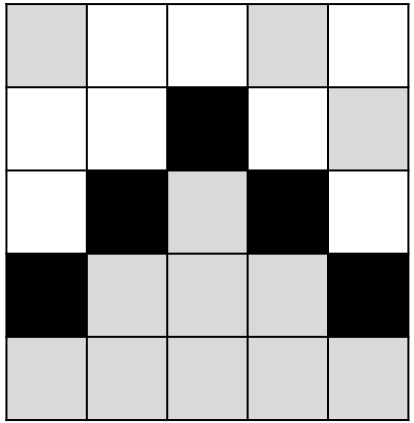
7	10	0
0	18	11
1	0	0

Exercício: Repete o processo de convolução + pooling com o filtro 2

Arquitetura CNN

Filtro de convolução da segunda camada

Imagem



Nova representação

Canal 1

3	10	10
21	0	0
0	1	0

Canal 2

7	10	0
0	18	11
1	0	0

Filtro de convolução 3D

Canal 1

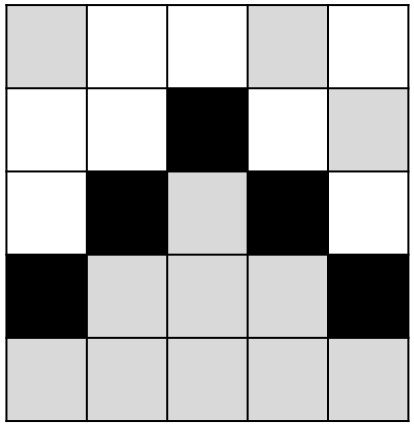
Canal 2

Resultado da convolução

Arquitetura CNN

Filtro de convolução da segunda camada

Imagem



Nova representação

Canal 1

3	10	10
21	0	0
0	1	0

Canal 2

7	10	0
0	18	11
1	0	0

Filtro de convolução 3D

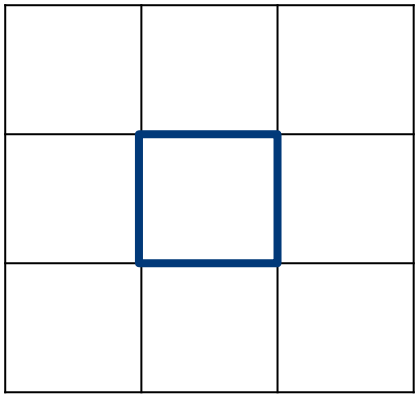
Canal 1

-1	-1	-1
-1	1	-1
-1	-1	-1

Canal 2

-1	-1	-1
-1	-1	1
-1	-1	-1

Resultado da convolução

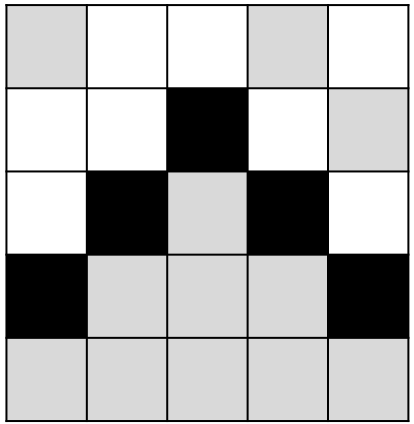


Exercício: Calcular a convolução 3D

Arquitetura CNN

Filtro de convolução da segunda camada

Imagem



Nova representação

Canal 1

3	10	10
21	0	0
0	1	0

Canal 2

7	10	0
0	18	11
1	0	0

Filtro de convolução 3D

Canal 1

-1	-1	-1
-1	1	-1
-1	-1	-1

Canal 2

-1	-1	-1
-1	-1	1
-1	-1	-1

Resultado da convolução

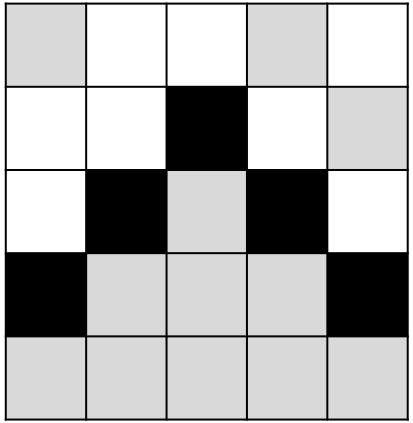
-43	-70	-39
7	-70	-60
-41	-50	-30

Exercício: Calcular a convolução 3D

Arquitetura CNN

Filtro de convolução da segunda camada

Imagem



Resultado da convolução

-43	-70	-39
7	-70	-60
-41	-50	-30

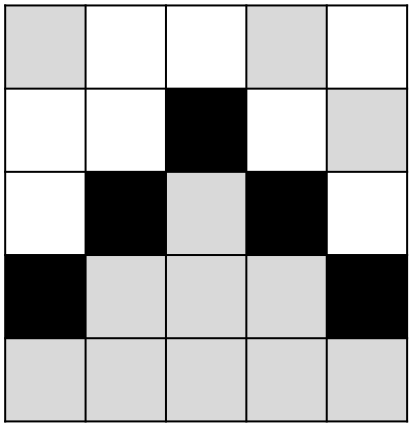
ReLU

0	0	0
7	0	0
0	0	0

Arquitetura CNN

Camada de classificação

Imagem



ReLU

0	0	0
7	0	0
0	0	0

$> 0 \rightarrow$ True

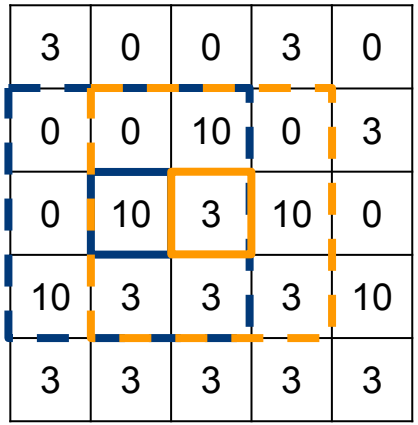
$= 0 \rightarrow$ False

Conclusão: Tem uma orelha de gato na imagem

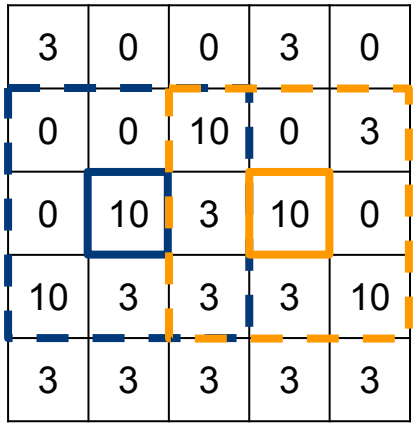
Arquitetura CNN

Outro mecanismo de redução de tamanho: **Stride**

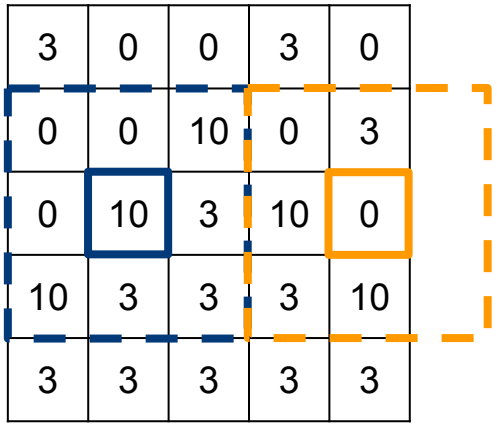
Stride = 1



Stride = 2



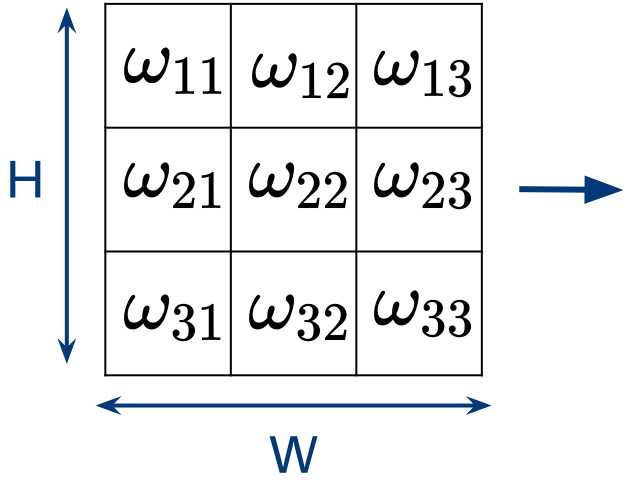
Stride = 3



Arquitetura CNN

Como escolher os filtros na prática?

Filtros de convolução parametrizados e aprendidos



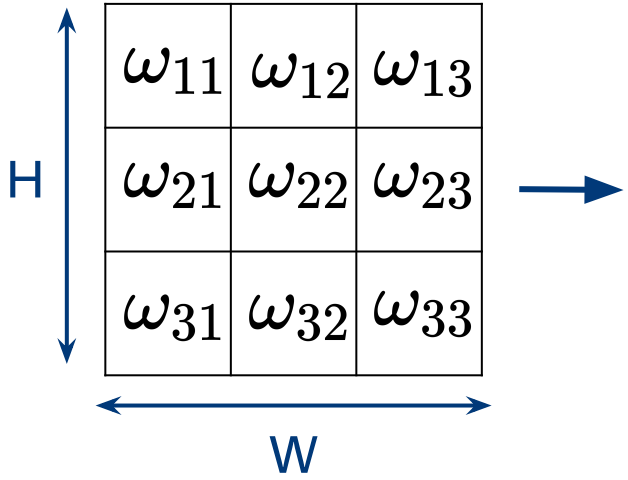
Aplicado à profundidade inteira da imagem de entrada

Número de parâmetros de 1 filtro 3x3 aplicado à uma imagem RGB?

Arquitetura CNN

Como escolher os filtros na prática?

Filtros de convolução parametrizados e aprendidos



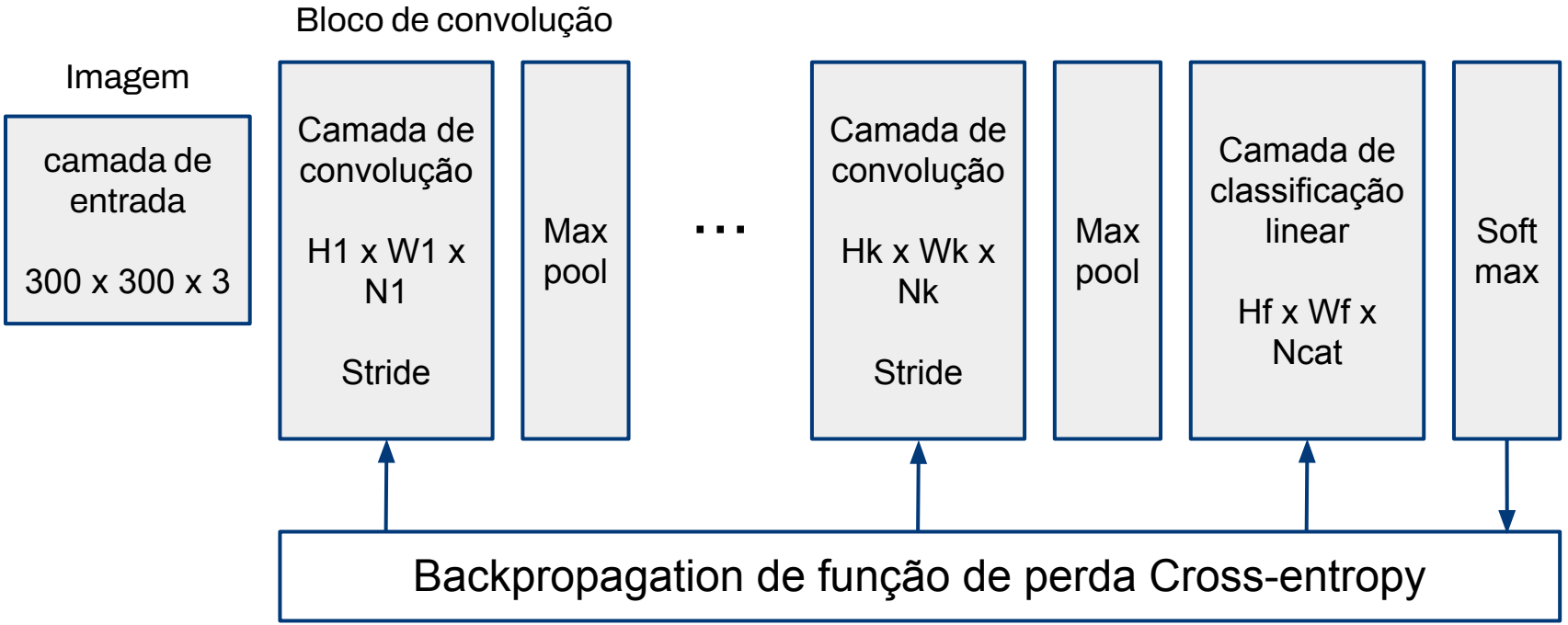
Aplicado à profundidade inteira da imagem de entrada

Número de parâmetros de 1 filtro 3x3 aplicado à uma imagem RGB?

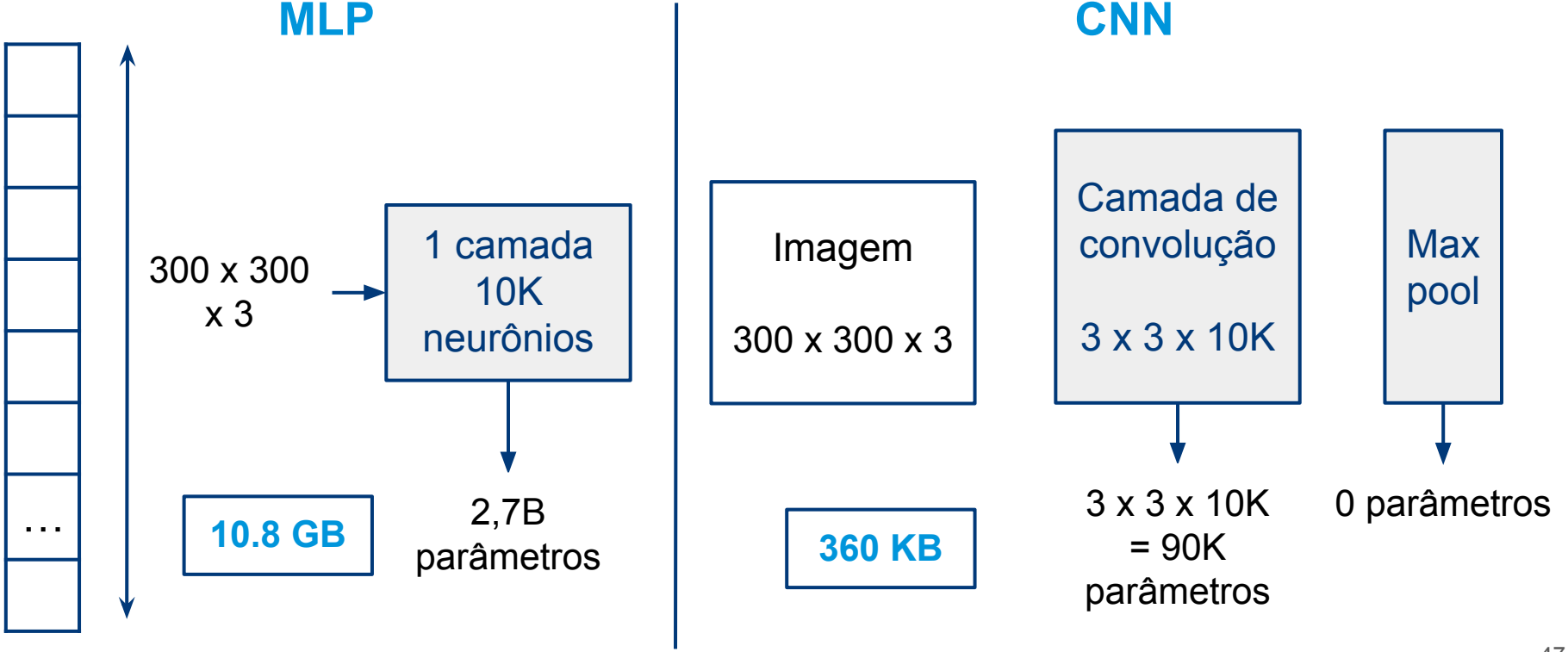
3 x 3 x 3 = 27

Arquitetura CNN

Construção e treinamento



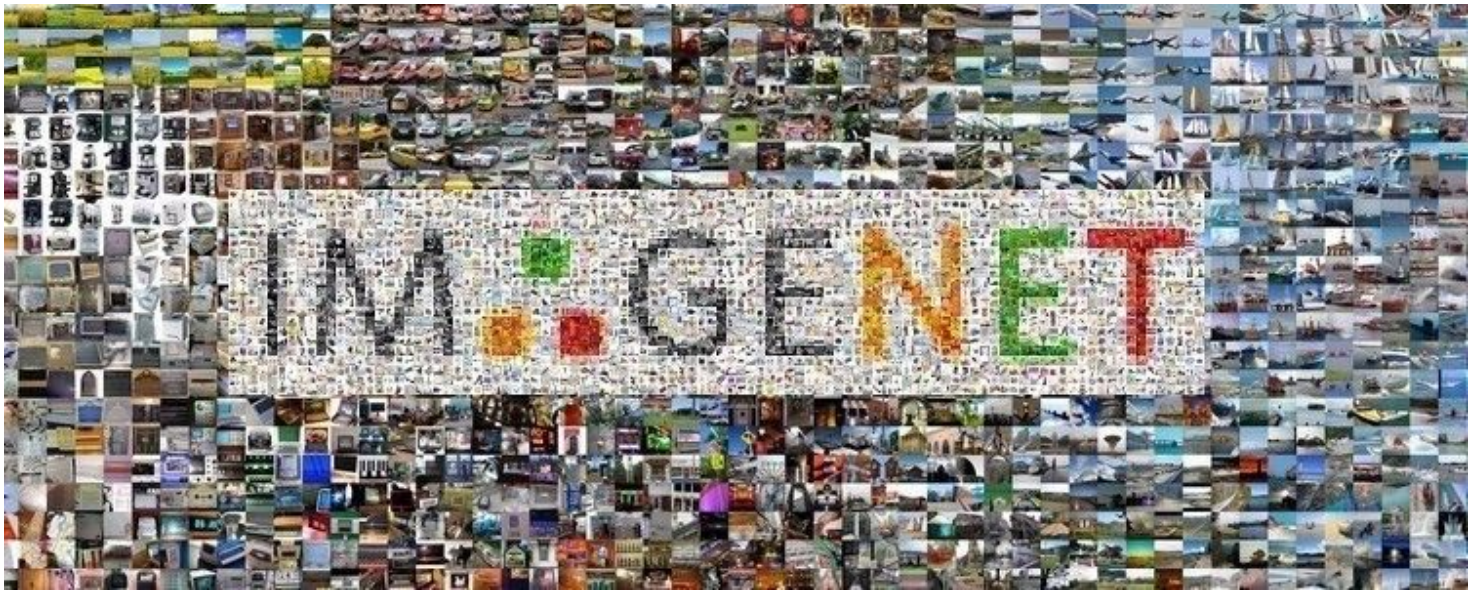
Número de parâmetros: MLP vs CNN



Competição ImageNet (ILSVRC 2012)

Classificar imagens entre 1000 categorias

1.2M imagens de treinamento | 150K imagens de teste

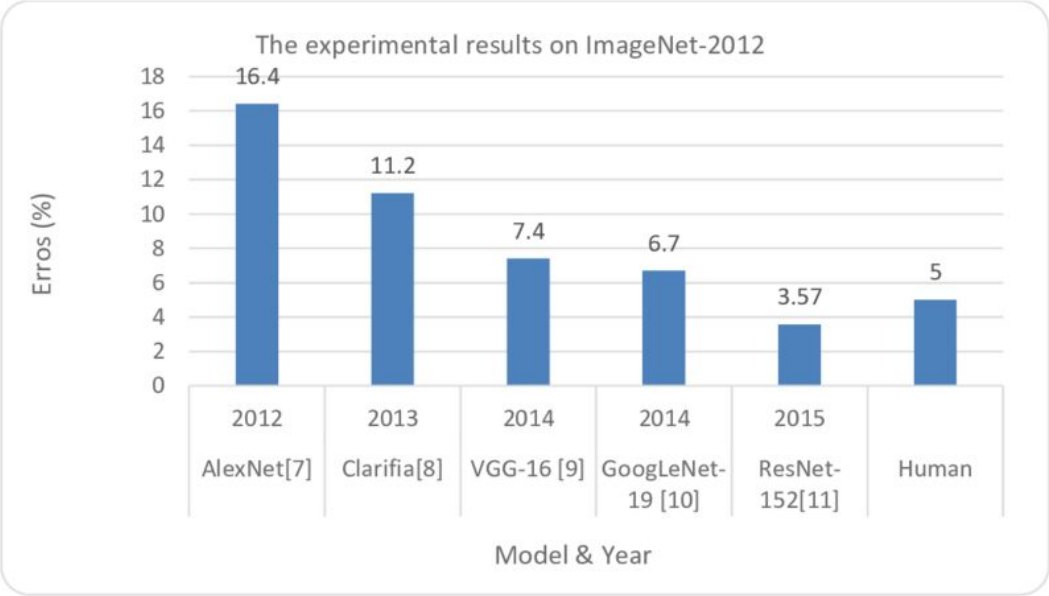


Source: *https://cv.gluon.ai/build/examples_datasets/imagenet.html*

Competição ImageNet (ILSVRC 2012)

Classificar imagens entre 1000 categorias

1.2M imagens de treinamento | 150K imagens de teste



Source: *https://www.researchgate.net/publication/323570864_The_History_Began_from_AlexNet_A_Comprehensive_Survey_on_Deep_Learning_Approaches*

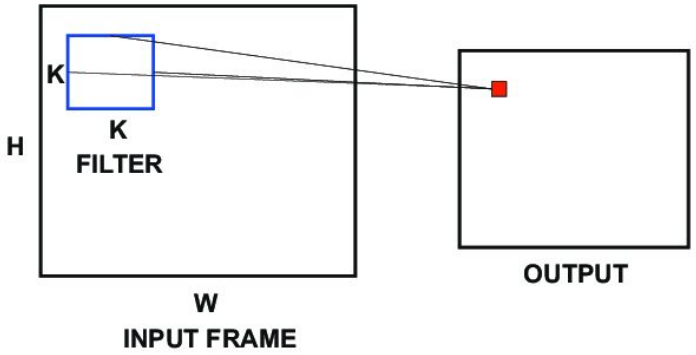
Arquitetura VGG 16

Source: https://www.datacorner.fr/vgg-transfer-learning/
Source 2: https://keras.io/api/applications/

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	0.790	0.945	22,910,480	126	109.42	8.06
VGG16	528	0.713	0.901	138,357,544	23	69.50	4.16
VGG19	549	0.713	0.900	143,667,240	26	84.75	4.38

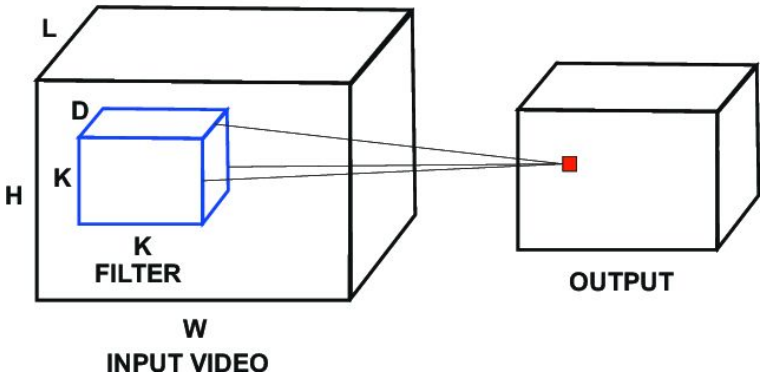
Adaptação da arquitetura CNN para vídeos

Imagens →



Convolução 2D

Vídeos →



Convolução 3D

Source: https://www.researchgate.net/publication/331801448_Combining_Global_and_Local_Convolutional_3D_Networks_for_Detecting_Depression_from_Facial_Expressions

Conclusão

- Apresentamos a arquitetura MLP e mostramos como pode ser desenvolvida → Dados tabulares
- Apresentamos a arquitetura CNN e mostramos como pode ser desenvolvida → Imagens

- Muitos elementos importantes sobre treinamento de redes neurais não foram discutidos ainda (próximas aulas)

Inicialização Otimização Regularização Avaliação

Implementação eficiente Implantação industrial

Fortalecer o conhecimento

- Exercício 1: Desenvolvimento de uma rede MLP
  - Escolher um dataset supervisionado no UCI ML repository (*https://archive.ics.uci.edu/ml/index.php*)
  - Desenvolver uma arquitetura MLP adaptada usando a técnica apresentada.
  - Implementar e testar com a biblioteca Scikit-learn.

- Exercício 2: Desenvolvimento de convoluções à mão
  - Testar as convoluções desenvolvidas nesta aula no exemplo seguinte.
  - Propor uma cadeia de convoluções para detectar este padrão.
  - Testar as convoluções quando os padrões mudam de posição (Invariância à translação)