Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Reconhecimento de Padrões/Objetos

André Tavares da Silva

andre.silva@udesc.br

Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

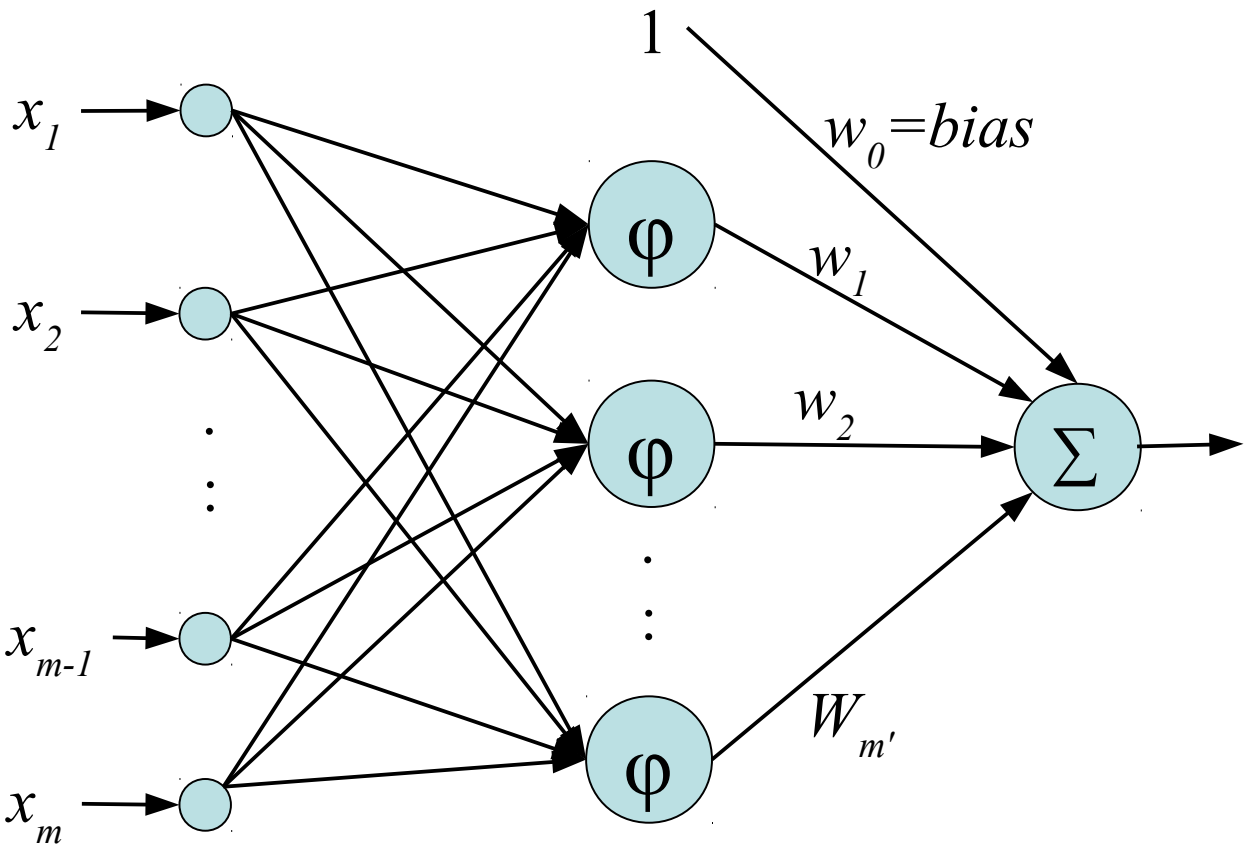
Rede RBF

(*Radial Basis Function*)

RNA RBF

- Uma rede neural com Função de Ativação de Base Radial (RBF) consiste em um modelo neural multicamadas, capaz de aprender padrões complexos e resolver problemas não-linearmente separáveis.
- A arquitetura de uma rede RBF tem três camadas: camada de entrada, na qual os padrões são apresentados à rede; a camada intermediária (única) que aplica uma transformação não linear do espaço de entrada para o espaço escondido (alta dimensionalidade); e camada de saída que fornece a resposta da rede ao padrão apresentado.

RNA RBF



entradas

Camada escondida
(funções de base radial)

Camada de saída

Teorema de Cover

- Um problema de classificação de padrões que "cai" num espaço de alta dimensão é mais provável ser linearmente separável do que em espaço de baixa dimensão (teorema de Cover da separabilidade de padrões): razão porquê a dimensão do espaço escondido de uma rede RBF ser alta.
- Quando se tem padrões linearmente separáveis (em uma dimensão maior), o problema de classificação torna-se mais simples.

O problema do XOR (novamente)

- Construiremos um classificador de padrões que produza 0 (zero) para entradas (1,1) ou (0,0) e 1 para (0,1) e (1,0).
- Para camada oculta usaremos as funções gaussianas:

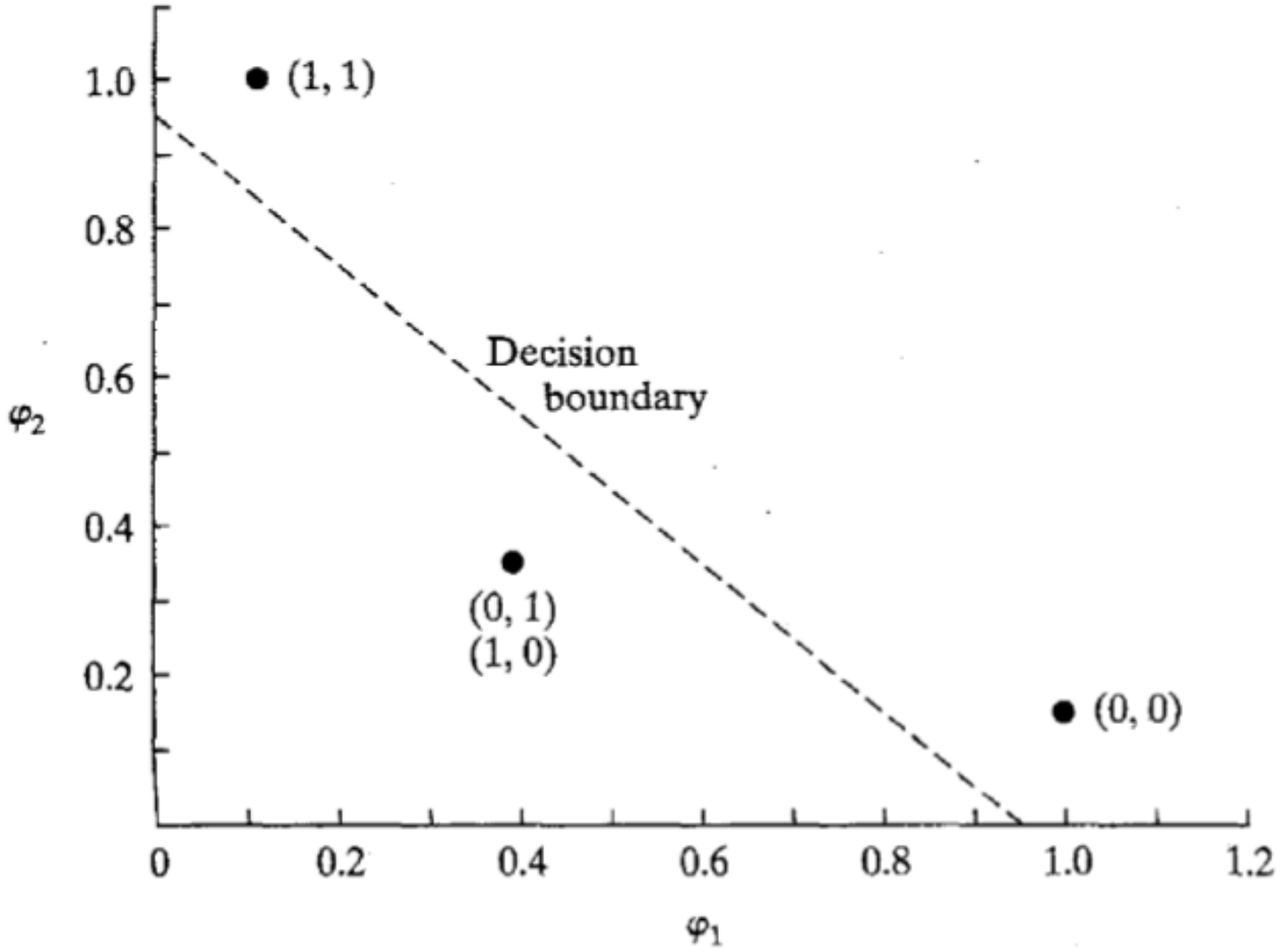
$$\varphi 1(x) = e^{-\|x-t_1\|^2}$$

$$\varphi 2(x) = e^{-\|x-t_2\|^2}$$

onde t_1=(1,1) e t_2=(0,0)

x	φ1	φ2
(0,0)	1	0.1353
(0,1)	0.3678	0.3678
(1,0)	0.3678	0.3678
(1,1)	0.1353	1

Diagrama de decisão do problema XOR



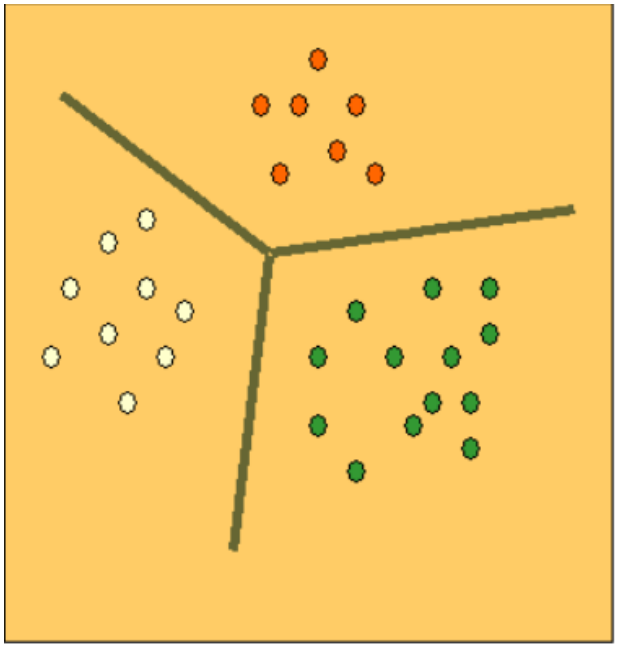
Rede RBF

- Nas redes de Função de Base Radial (RBF), a função de ativação de cada neurônio da camada escondida é função da distância entre seus vetores de peso e de entrada.
- É uma evolução da MLP
- Redes de duas camadas:
  - Primeira camada: Utiliza funções de ativação não lineares (funções de base radial).
  - Segunda camada: Utiliza funções de ativação lineares.

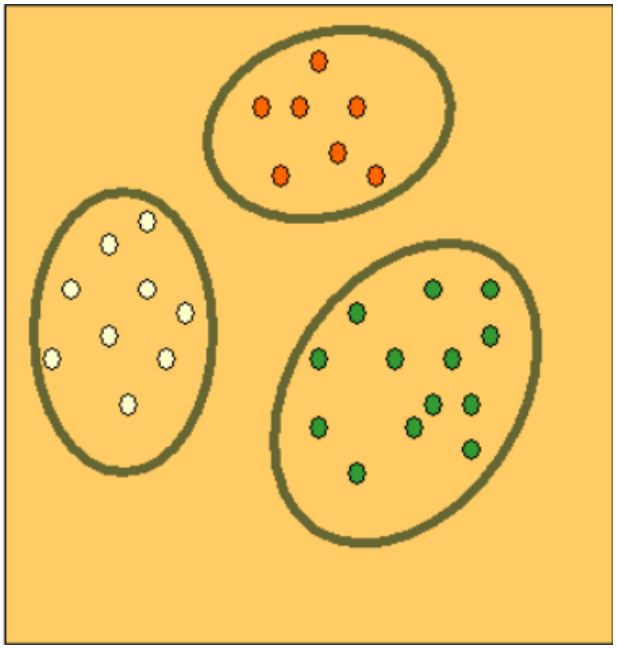
Rede RBF

- A diferença principal entre e RBF é que a primeira utiliza hiperplanos para particionar espaço de entradas enquanto a segunda utiliza hiperelipsóides para particionar o espaço de entradas (na camada escondida).

MLP



RBF



MLP x RBF

- Uma rede RBF (na maioria dos casos) tem uma única camada escondida.
- Os nós da MLP, localizados nas camadas escondidas e de saída, compartilham um modelo neuronal comum. Já na rede RBF, os nós da camada escondida são calculados diferentemente e têm um propósito diferente dos de saída.
- A camada escondida de uma rede RBF é não-linear e a saída é linear.

MLP x RBF

- O argumento da função de ativação de cada unidade escondida numa rede RBF calcula a distância (euclidiana ou não) entre o vetor de entrada e o centro da unidade. Na MLP é calculado produto interno do vetor de entrada e do vetor de pesos sinápticos da unidade.
- Redes RBF normalmente usam não-linearidades localizadas exponencialmente decrescentes (ex.: funções gaussianas) gerando aproximações locais.

MLP x RBF

- Uma rede MLP frequentemente tem muitas camadas de pesos e um complexo padrão de conectividade. Além disso, uma variedade de diferentes funções de ativação podem ser utilizadas na mesma rede. Uma rede RBF, no entanto, geralmente tem uma arquitetura simples.
- A performance de generalização de uma rede MLP é em geral mais robusta.

MLP x RBF

- Todos os parâmetros em uma rede MLP são usualmente determinados ao mesmo tempo, como parte de uma única estratégia global de treinamento, envolvendo treinamento supervisionado de alto custo computacional pela necessidade de retropropagação do erro. Já uma rede RBF é tipicamente treinada em dois estágios: as funções de base radial sendo determinadas primeiramente por meio de técnicas não-supervisionadas; e a segunda camada determinada por métodos lineares supervisionados de convergência rápida.

RBF

- Cada neurônio da camada oculta calcula uma função base radial
  - centro: protótipo de um *cluster*
  - largura: área de influência do protótipo
- Entrada total

$$u = \|x_i - t_i\| \qquad \text{(camada oculta)}$$

$$u = \sum w_i \varphi_i \|x_i - t_i\| \qquad \text{(camada de saída)}$$

- Medida de distância normalmente é a Euclidiana

RBF

- Estados de ativação:
  - 1 (+1) = ativo.
  - 0 (-1) = inativo.
- Função da primeira camada é transformar conjunto de dados não-linearmente separáveis em linearmente separáveis.
- Função de ativação das unidades escondidas:
  - Não linear.
  - Valor aumenta ou diminui com relação à distância a um ponto central.

Funções de Ativação

Lâmina *spline* fina	$\phi(\zeta) = \frac{\zeta}{\sigma^2} \log\left(\frac{\zeta}{\sigma}\right)$
Multi-Quadrática	$\phi(\zeta) = \sqrt{\zeta^2 + \sigma^2}$
Multi-quadrática inversa	$\phi(\zeta) = \frac{1}{\sqrt{\zeta^2 + \sigma^2}}$
Gaussiana	$\phi(\zeta) = \exp\left(-\frac{\zeta^2}{2\sigma^2}\right)$

Funções de Ativação

- Escolha da função depende de:
  - Nível de conhecimento sobre os dados: as funções devem cobrir pontos uniformemente distribuídos do espaço de entradas.
  - Conhecimento da estrutura geral das entradas: levar a estrutura em consideração na escolha das funções.
- Treinamento é feito em dois estágios
  - Definição da camada oculta (não supervisionado)
  - Treinamento da camada de saída (supervisionado)

Definição da camada oculta

- Determinar os parâmetros das funções de base radial:
  - Número de bases,
  - Centros das bases,
  - Larguras das bases.
- Definições de centros:
  - Existem várias abordagens: seleção aleatória dos centros ou *clustering* (K-means, SOM, algoritmos genéticos,...).

Definição da camada oculta

- Número de funções base:
  - Geralmente definido por tentativa e erro.
  - Sejam m o número de funções base, n o tamanho de Z_1 e c a quantidade de classes: $c < m << n$ (m=n leva a overfitting e m=c não funciona se alguma classe tiver mais de uma região associada).
  - Deve ser determinado pela complexidade dos dados.
  - Número de funções base radial = número de *clusters*.

Definição dos centros

- K-*means clustering*:
  - Centros são colocados no meio de agrupamentos de vetores de entrada (*clusters*).
  - Utiliza aprendizado não-supervisionado.
  - Divide os vetores de entrada em K conjuntos disjuntos S_j: cada conjunto S_j tem N_j vetores.
  - Objetivo: minimizar distâncias entre vetores de S_j e seu centro.

Definição das larguras

- Heurísticas para definir larguras σ das funções radiais

  1)Atribuir a σj um valor constante (geralmente 1).

  2)Todas as larguras iguais à média sobre todas as distâncias Euclidianas entre o centro de cada unidade N_i e o centro da unidade N_j mais próxima.

$$\sigma = \frac{1}{m} \sum_{i=1}^{m} \|t_i - t_j\|$$

  onde t_i é o centro mais próximo de t_j.

Definição das larguras

- Heurísticas para definir larguras σ das funções radiais

  3) Atribuir a cada unidade uma largura diferente baseada na distância do seu centro ao centro da unidade mais próxima $\sigma_i = \alpha \|t_i - t_j\|$ onde t_i é o centro mais próximo de t_j e $1.0 < \alpha < 1.5$

  4) Atribuir a cada σ_j a distância média de seu centro aos N vetores de entrada mais próximos.

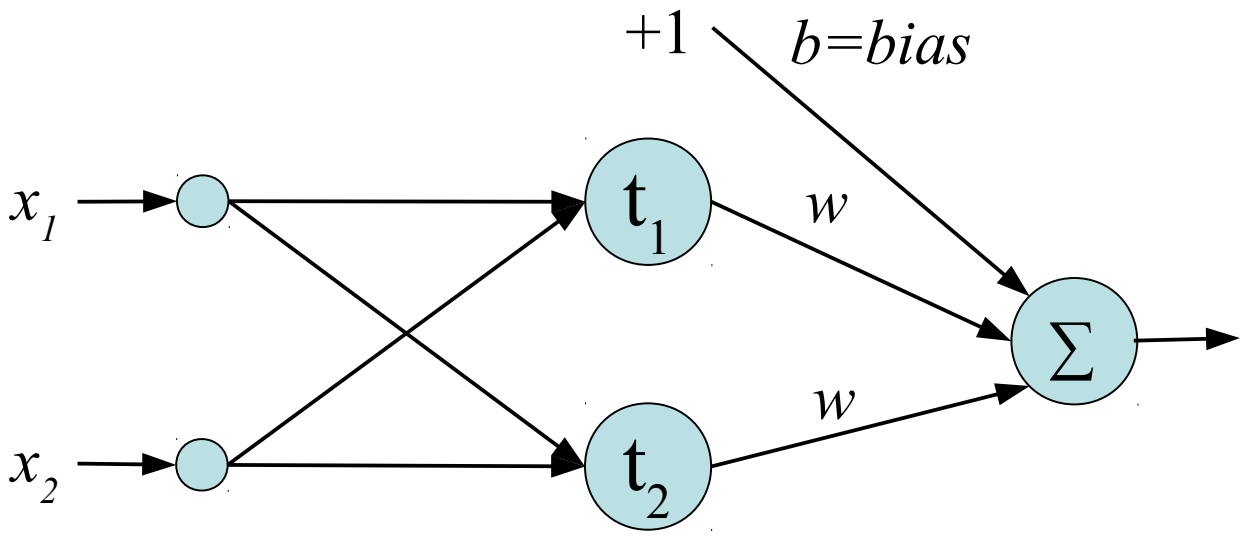
Treinamento da camada de saída

- Determinar pesos da camada de saída:
  - Recebe vetores linearmente separáveis,
  - Supervisionado,
  - Classificação/regressão dos vetores de entrada.
- Métodos para ajustar pesos:
  - Decomposição em valores singulares,
  - Regra *delta*.

XOR usando uma rede RBF

- Os centros t_1 e t_2 são: $t_1 = [1, 1]^T$ e $t_2 = [0, 0]^T$
- Par de funções gaussianas: $G(\|x - t_i\|) = e^{-\|x - t_i\|^2}$, i=1,2
- Para a unidade de saída, assume-se:
  - usa compartilhamento de pesos, devido à simetria do problema, ou seja, ambas as unidades escondidas tem o mesmo peso *w;*
  - e a unidade de saída inclui um bias *b*.

XOR usando uma rede RBF



entradas

Camada oculta
(funções de base radial)

Camada de saída

XOR usando uma rede RBF

- A relação entrada-saída da rede é definida por: $y(x)=\sum_{i=1}^{2} wG(\|x-t_i\|)+b$

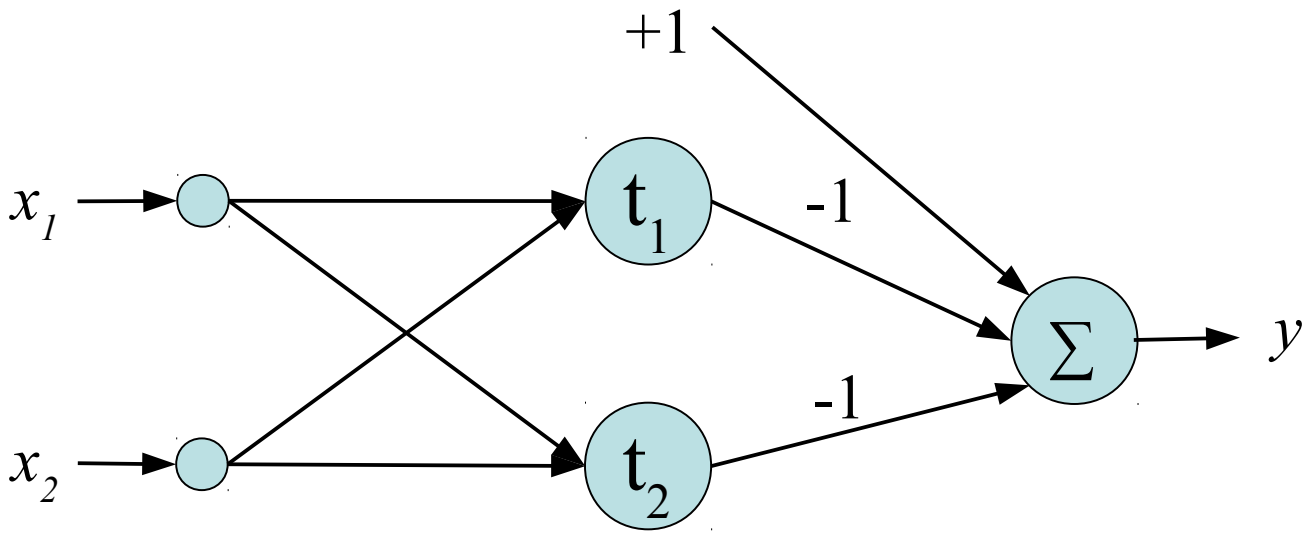
- Para ajustar os dados de treinamento é necessário que $\boldsymbol{y(x_j) = d_j, j=1,2,3,4}$ onde $\boldsymbol{x_j}$ é um vetor de entrada e $\boldsymbol{d_j}$ é o valor desejado na saída.

j	x_j	d_j
1	(0,0)	0
2	(0,1)	1
3	(1,0)	1
4	(1,1)	0

XOR usando uma rede RBF

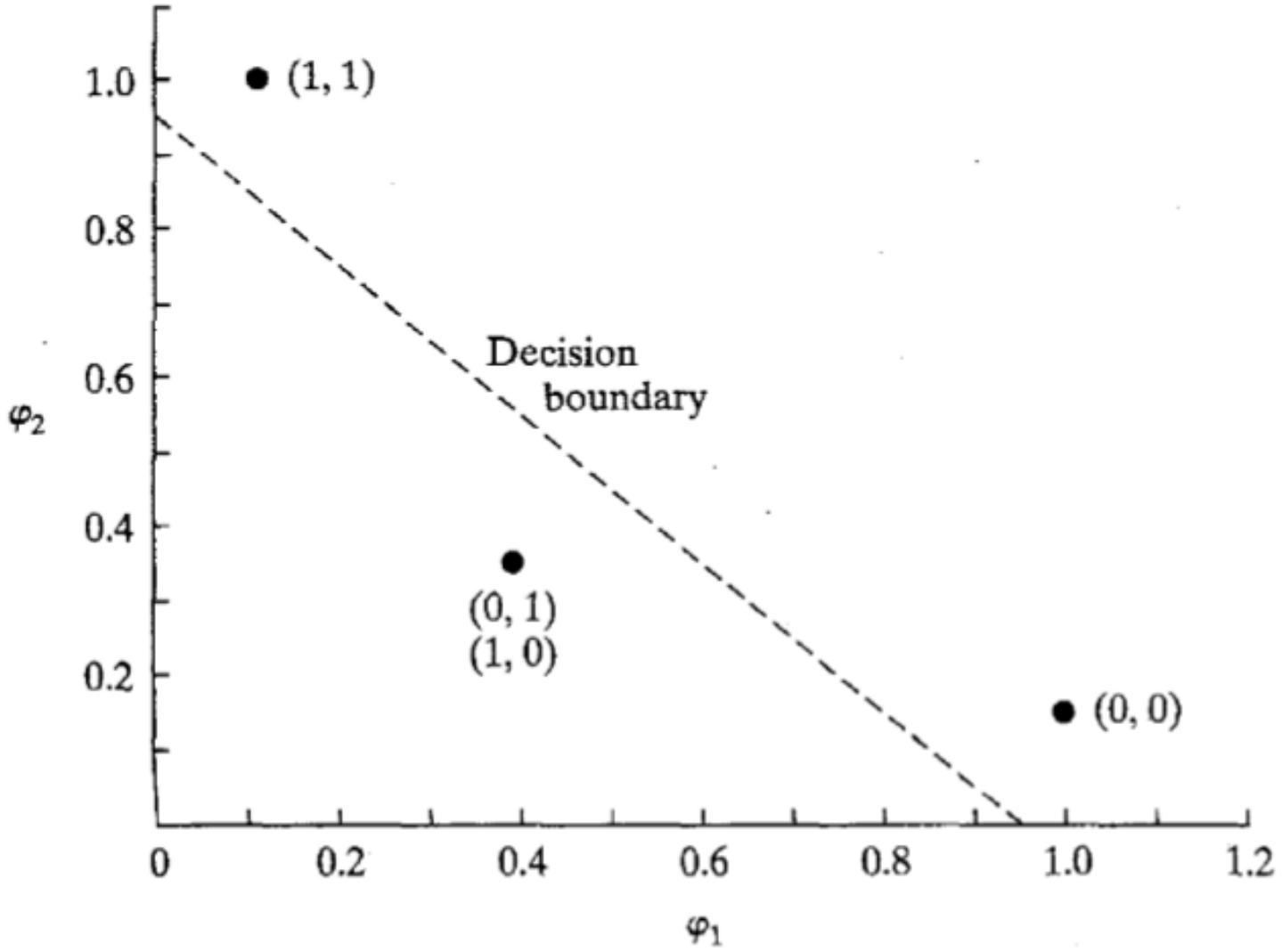
- Sendo t_1=(1,1), t_2=(0,0), w=–1 e b=+1, tem-se

$$y = -e^{-\|x - t_1\|^2} - e^{-\|x - t_2\|^2} + 1$$



- Se $y > 0$ então é da "classe" 1, senão é 0 (zero).

Diagrama de decisão do problema XOR



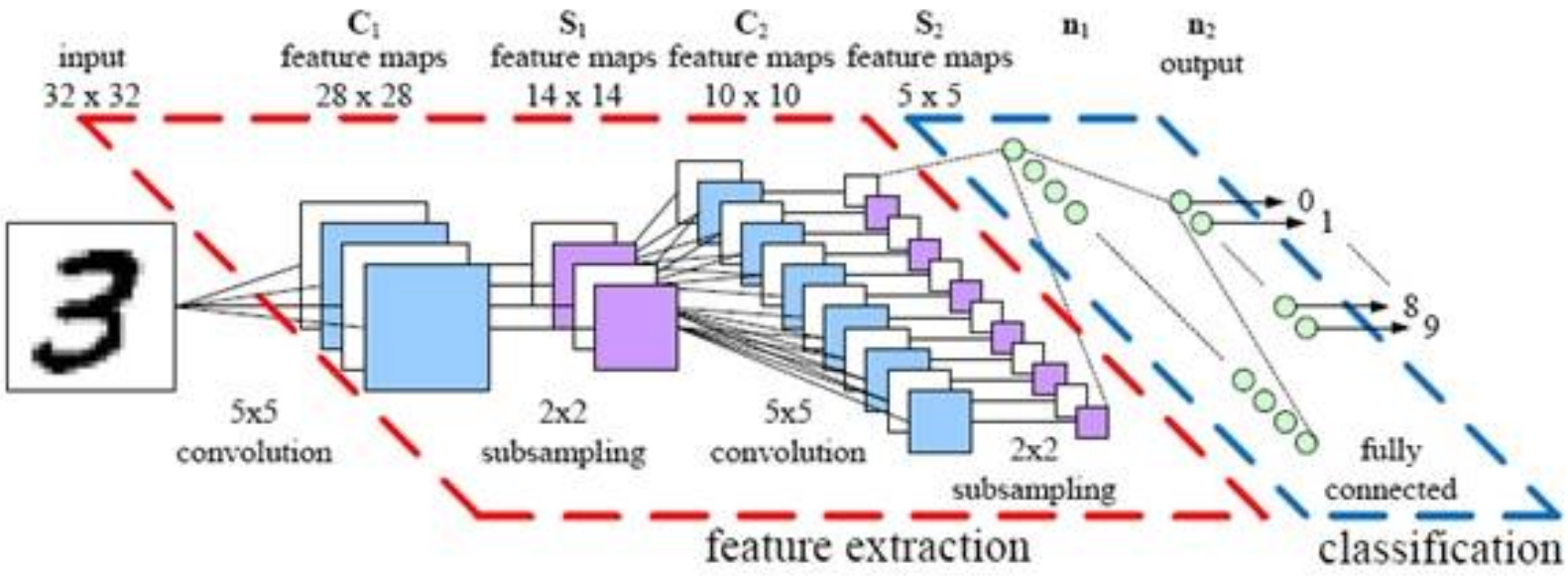
Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Rede CNN

(*Convolutional Neural Network*)

Rede CNN (LENET-5)



PEEMEN, Maurice; MESMAN, Bart; CORPORAAL, Henk. **Speed sign detection and recognition by convolutional neural networks**. In: Proceedings of the 8th International Automotive Congress. 2011. p. 162-170.

CNN

- Inspiradas no modelo biológico da visão
- Usa conceito de Redes Multi-Camadas
- Idealizada no ínicio do anos 90 [Lecun], e vasta aplicação após 2006 devido a "popularização" de GPU's
- Treinamento requer alto custo computacional e uma base de dados grande (podendo necessitar de aumento de dados artificiais)

Camadas

- Convolucional : Definem os filtros
  (Aprendizado / BackPropagation)
- Ativação: Neurônios
  (Relu / Sigmoid / TangH)
- ReLU (*Rectified Linear Units*)
  (max(0,x), tanh(x), sigmoid,...)
- Pooling : Reduzem as escalas
  (Max, Media, etc..)
- Fully-Connected (FC): Camada que determina as classes (Classificador)

Convolução

0,0	0,1	0,2	0,3	0,4	0,5
1,0	1,1	1,2	1,3		
2,0	2,1	2,2	2,3		
3,0	3,1	3,2	3,3		
4,0					
5,0					

Original image

+

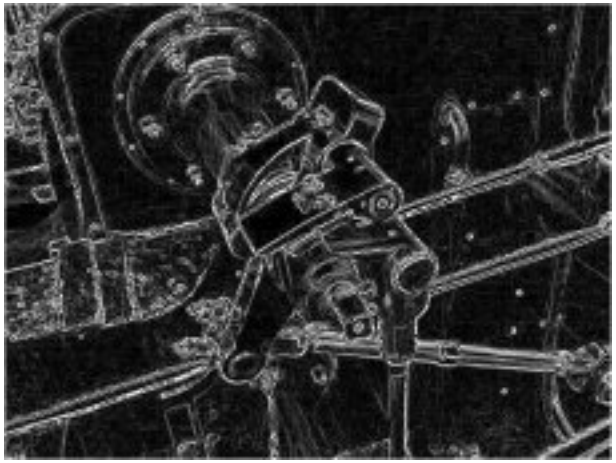
-1	0	+1
-2	0	+2
-1	0	+1

x filter

=

0,0	0,1	0,2	0,3	0,4	0,5
1,0	1,1	1,2	1,3		
2,0	2,1	2,2	2,3		
3,0	3,1	3,2	3,3		
4,0					
5,0					

Edge detected image



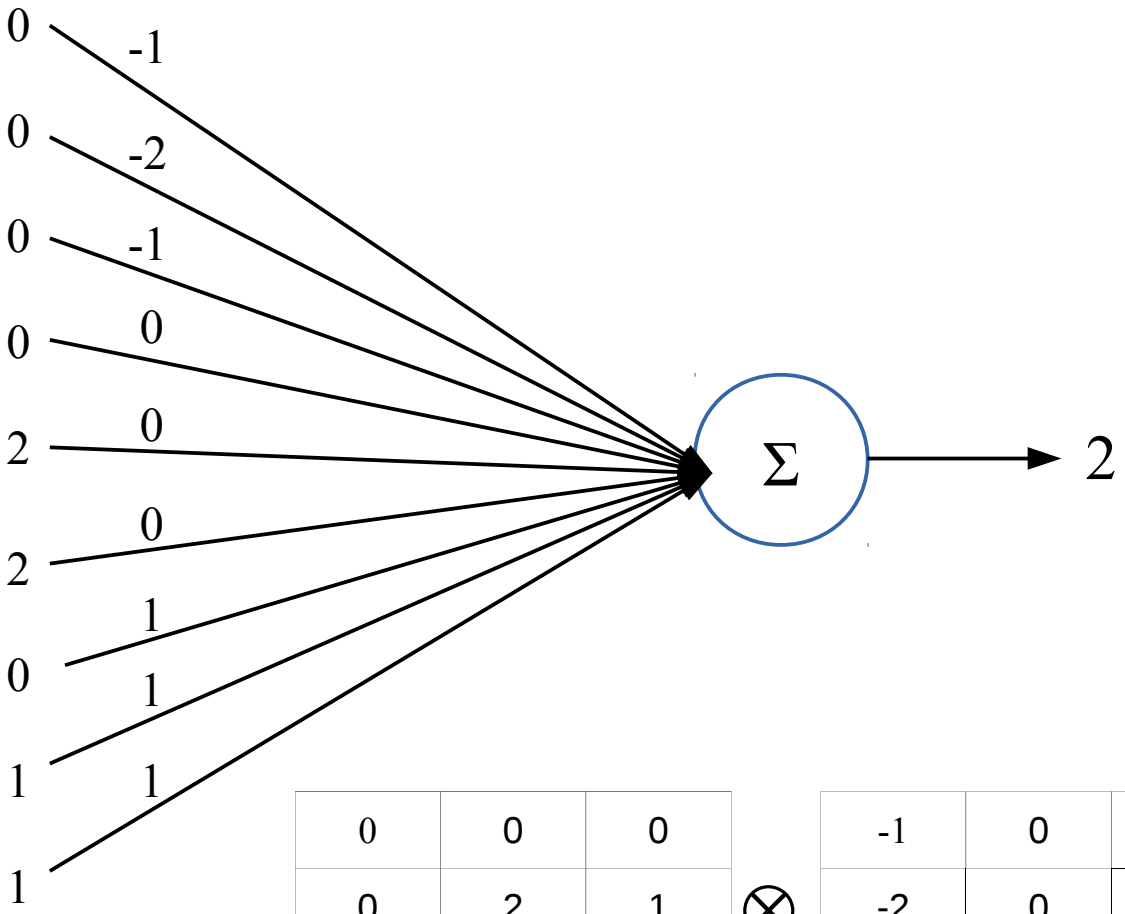
Convolução

0	0	0	0	0	0	0
0	2	1	1	3	3	0
0	2	1	1	2	2	0
0	3	1	1	3	2	0
0	3	1	1	3	2	0
0	2	1	1	2	2	0
0	0	0	0	0	0	0

$$\nabla_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

	2					

Convolução



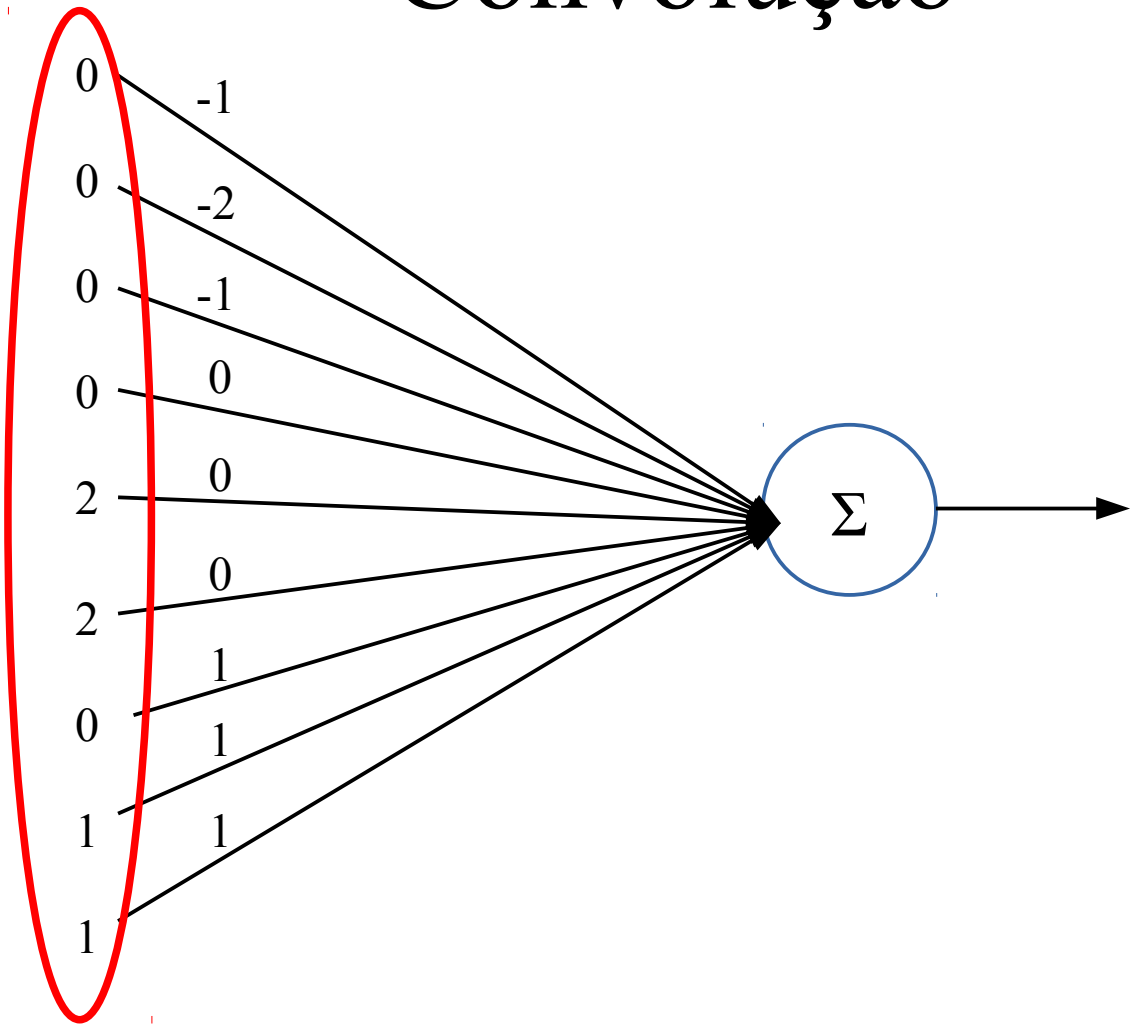
0	0	0
0	2	1
0	2	1

$\otimes$

-1	0	1
-2	0	1
-1	0	1

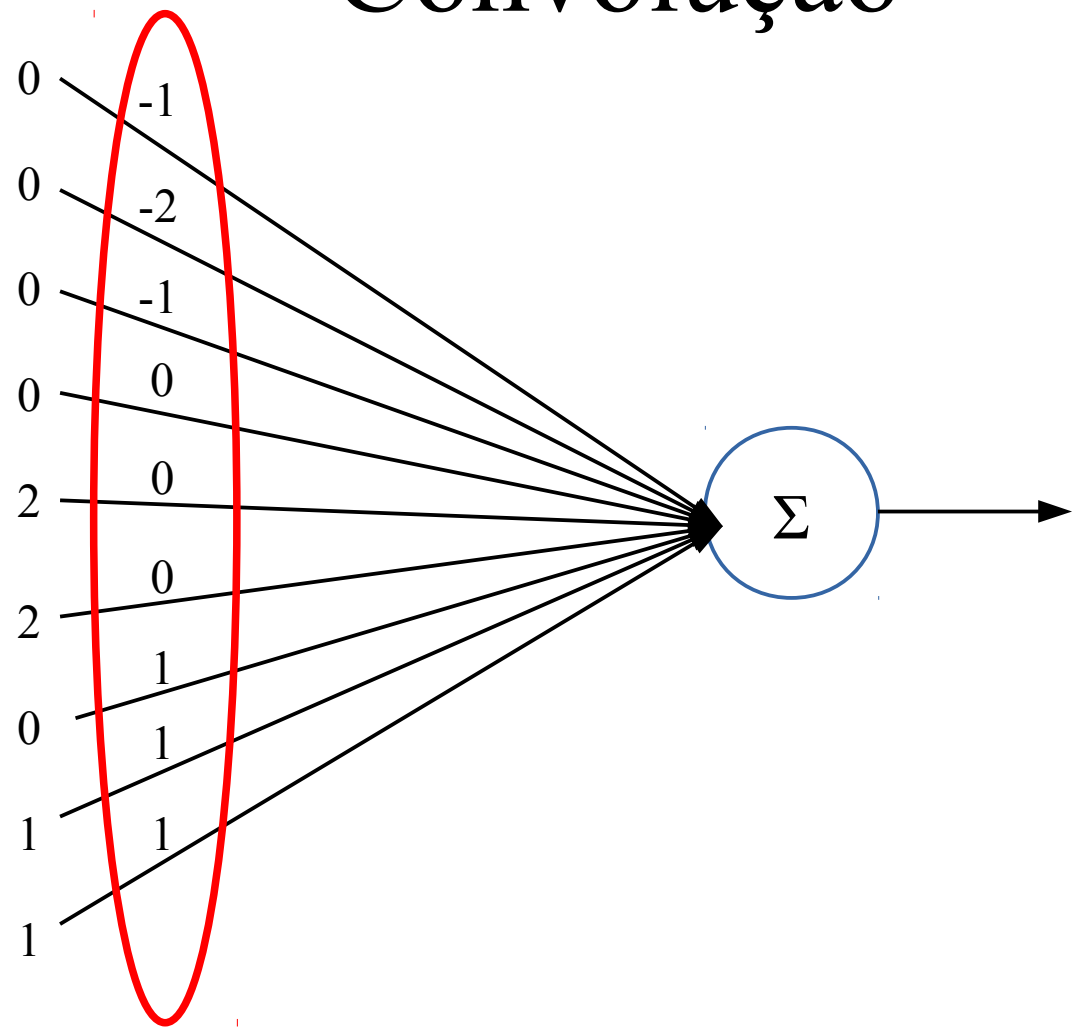
$= 2$

Convolução



Padrão de entrada (pixels)

Convolução



Pesos (kernel)

Convolução

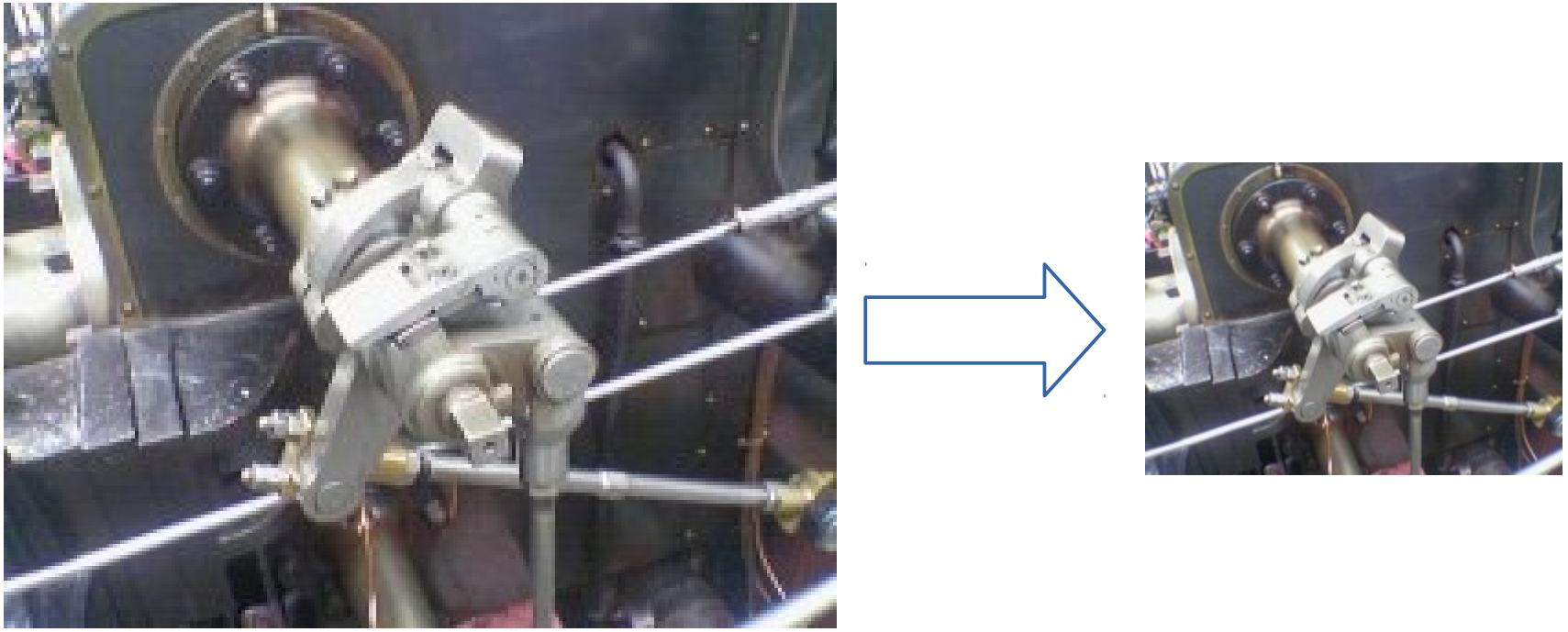
- Todos os neurônios desta camada compartilham os mesmos pesos a fim de realizar uma convolução (como vimos anteriormente);
- Todos os neurônios detectam a mesma característica (bordas, por exemplo) em todas as posições da imagem. Isso pode ser realizado em paralelo, se for utilizada uma GPU, por exemplo;
- O número de parâmetros livres é reduzido, reduzindo o processamento no aprendizado.

Convolução

- O número de convoluções a serem utilizadas é um parâmetro da rede;
- Em vez de testarmos diferentes *kernels* (tipos de convolução), a rede vai aprender quais as melhores configurações de "filtros" serão utilizados nas imagens;
- Em uma imagem colorida, o filtro é aplicado em cada um dos canais. Podendo gerar uma saída para cada canal ou realizada uma operação (média) para reduzir o processamento (tamanho da rede).

Pooling

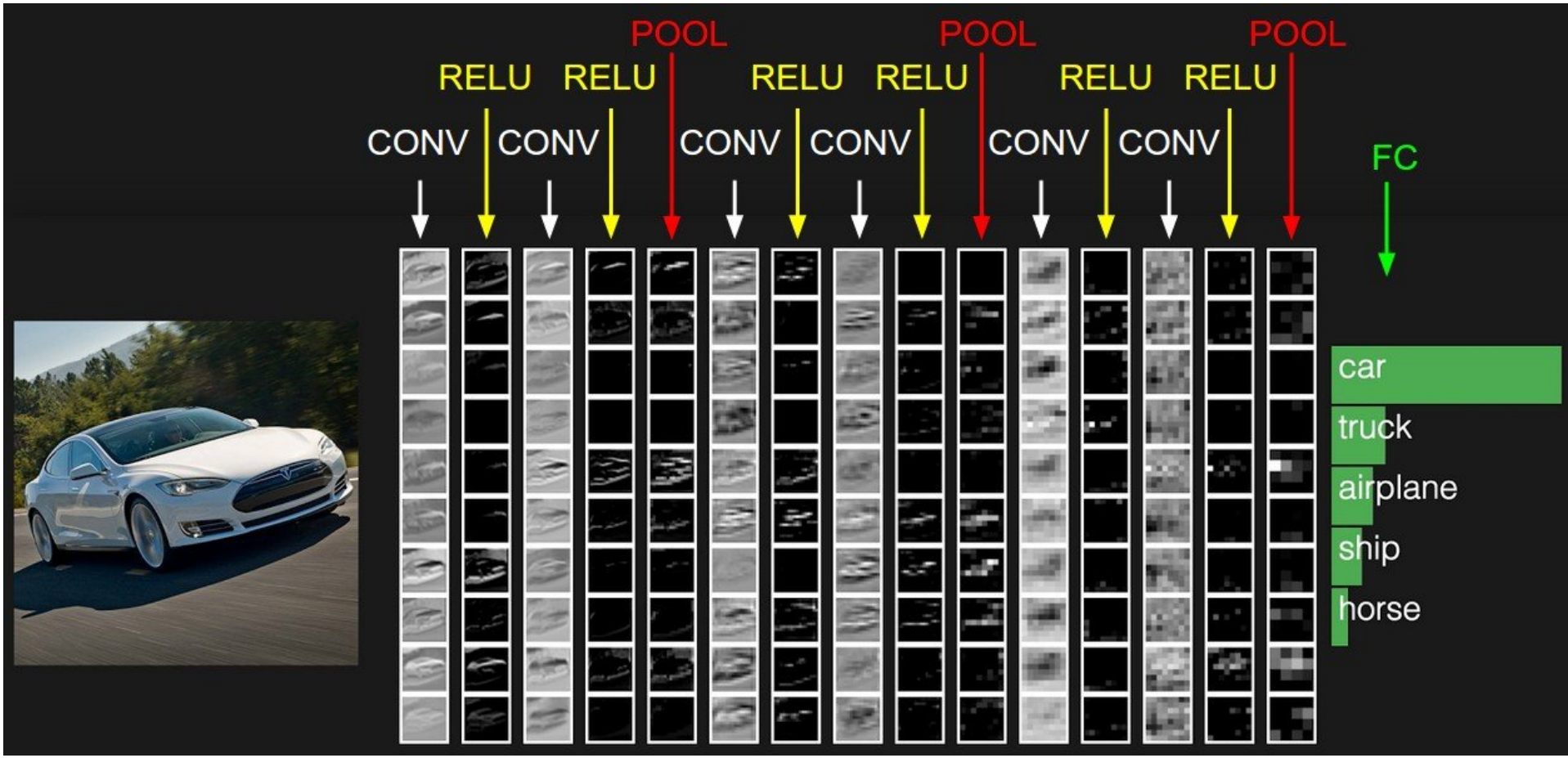
- Os pooling's são necessários para reduzir a quantidade de características por filtro (redução de escala). A Subamostragem dos pixels não altera os objetos.



ReLU (*Rectified Linear Units*)

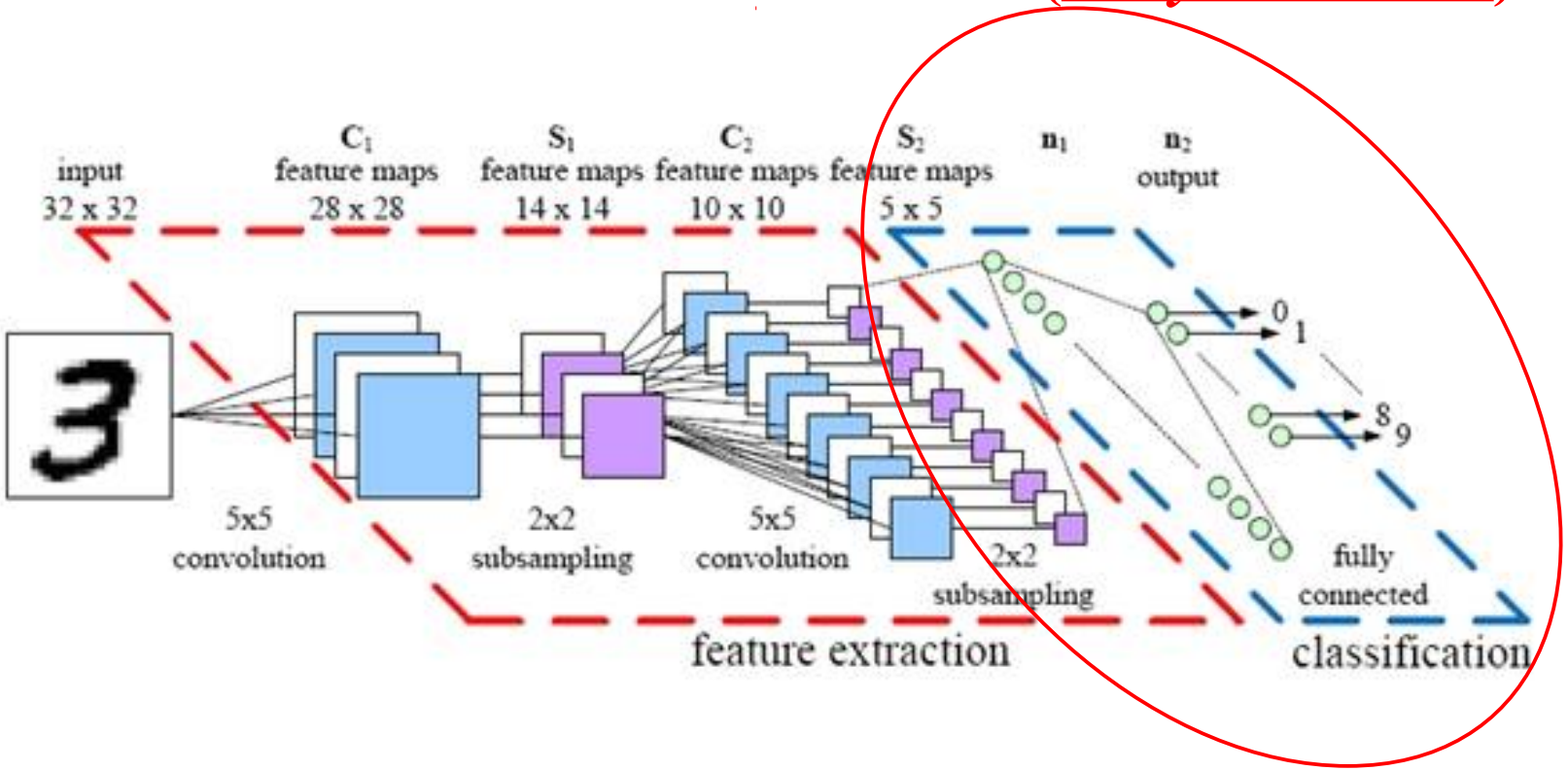
- Camada que aplica uma função de ativação sem saturação, aumentando as propriedades não lineares da função de decisão para toda a rede sem afetar os valores obtidos pela camada convolucional;
- Normalmente é utilizada a função ReLU $f(x)=max(0,x)$, mas podem ser usadas funções como $f(x)=|tanh(x)|$ ou $f(x)=(1+e^{-x})^{-1}$ (função sigmoide). A ReLU é favorita por ser muitas vezes mais rápida que as outras.

CNN


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

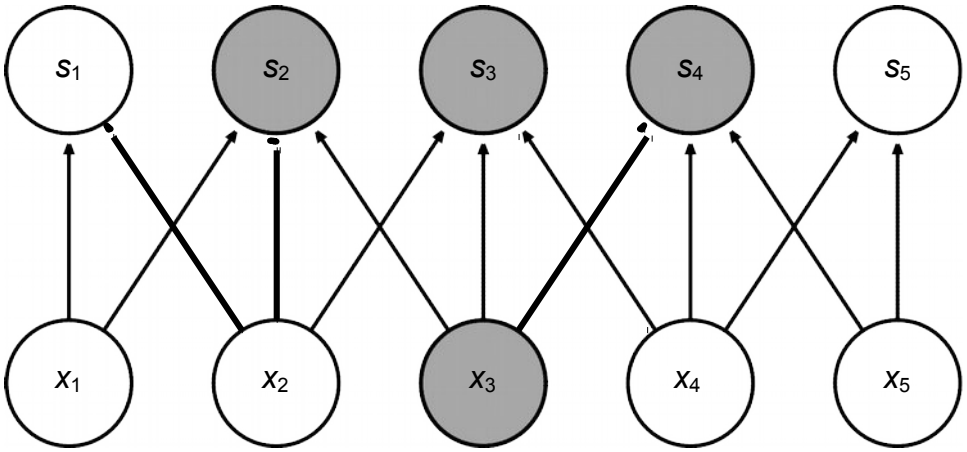
Rede CNN (LENET-5)

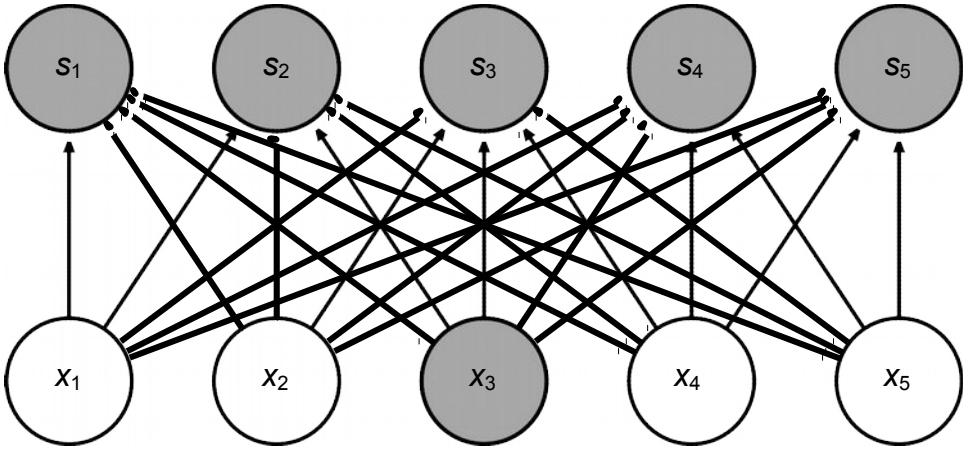
MLP densa (*Fully-Connected*)



PEEMEN, Maurice; MESMAN, Bart; CORPORAAL, Henk. **Speed sign detection and recognition by convolutional neural networks**. In: Proceedings of the 8th International Automotive Congress. 2011. p. 162-170.

Rede esparsa ou densa





Treinamento

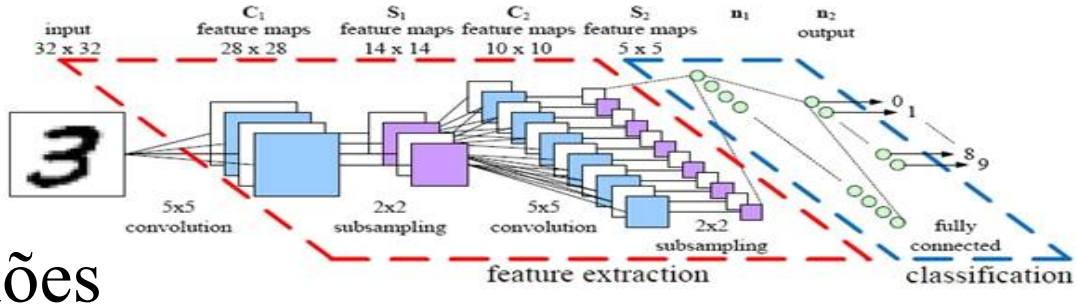
- Inicialização dos filtros e camada FC
- Fase de Aprendizado
  (Pode demorar horas ou dias dependendo da configuração da rede - uso extenso de GPUs ou nuvens)
- Validação e Backpropagation
- Armazenamento dos pesos e filtros ao longo do processo

CNNs

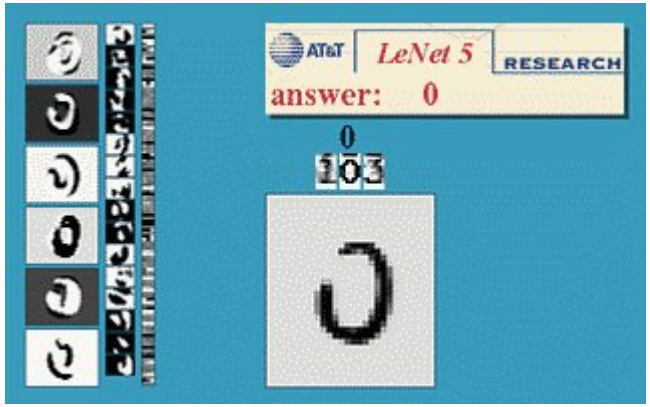
- Redes
  - LENET 5
  - AlexNet
  - GoogLeNet
- Ferramentas
  - Google Cloud Vision
  - IBM Watson Visual Recognition
  - Clarifai

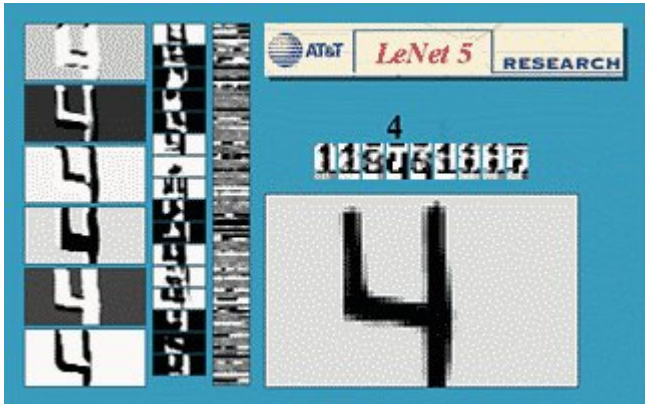
LENET 5

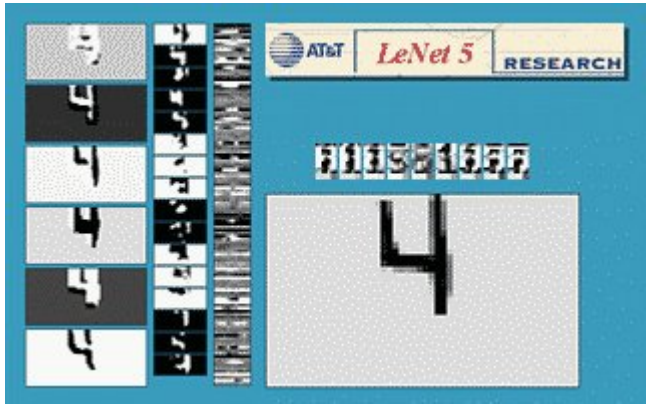
- Primeira CNN implementada e testada com sucesso (Bell Labs) / Yan Lecun – 1998
- Reconhecimento de Dígitos Manuscritos
  - MNIST DATASET (10 Classes [0-9])
  - 60 K Treinamento
  - 10 K Teste
  - 0.95% (erro)
  - ~345 K de conexões
  - ~60 K parâmetros

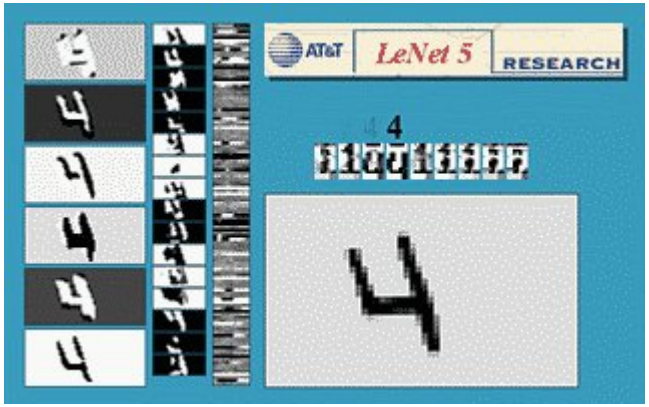


LENET 5

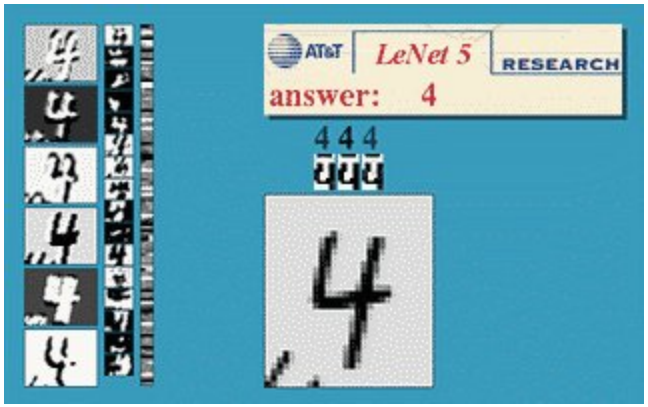

AT&T
LeNet 5
RESEARCH
answer: 0
0

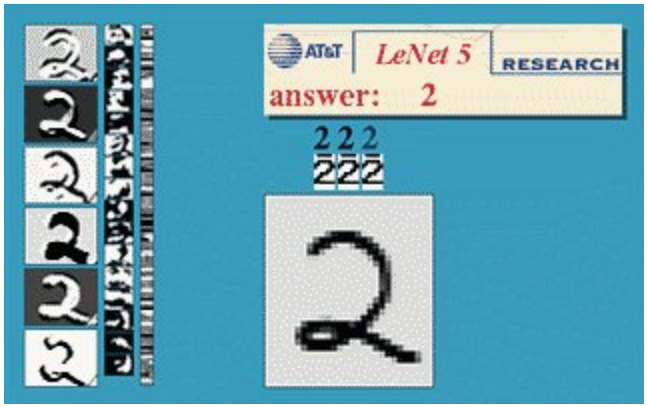

AT&T
LeNet 5
RESEARCH
4

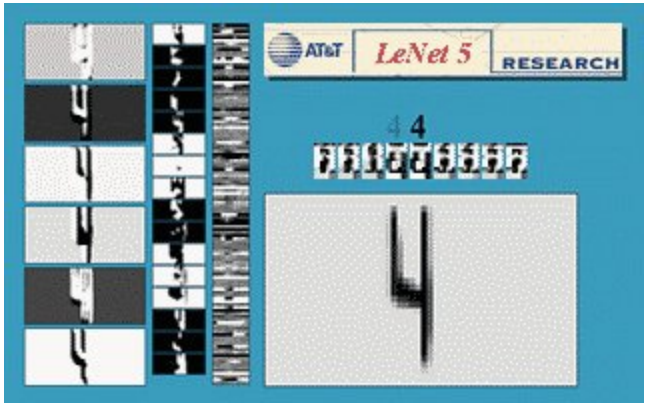

AT&T
LeNet 5
RESEARCH

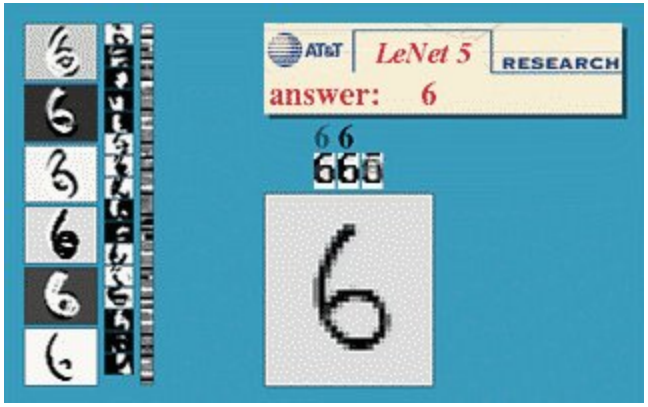

AT&T
LeNet 5
RESEARCH
4

LENET 5


AT&T
LeNet 5
RESEARCH
answer: 4

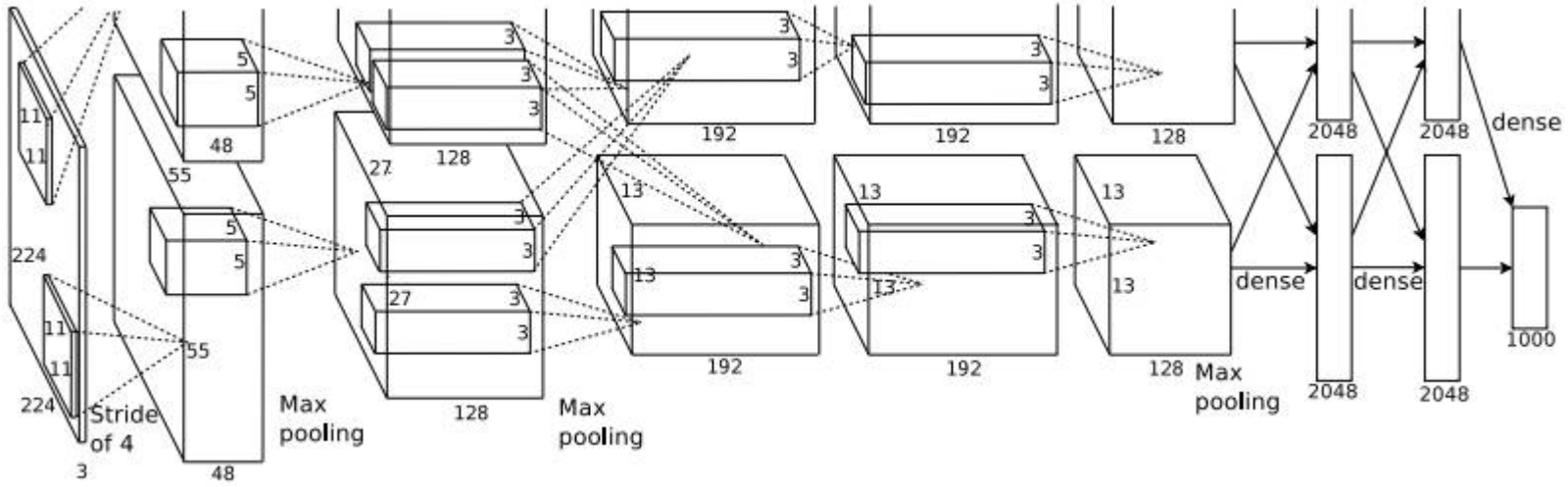

AT&T
LeNet 5
RESEARCH
answer: 2


AT&T
LeNet 5
RESEARCH


AT&T
LeNet 5
RESEARCH
answer: 6

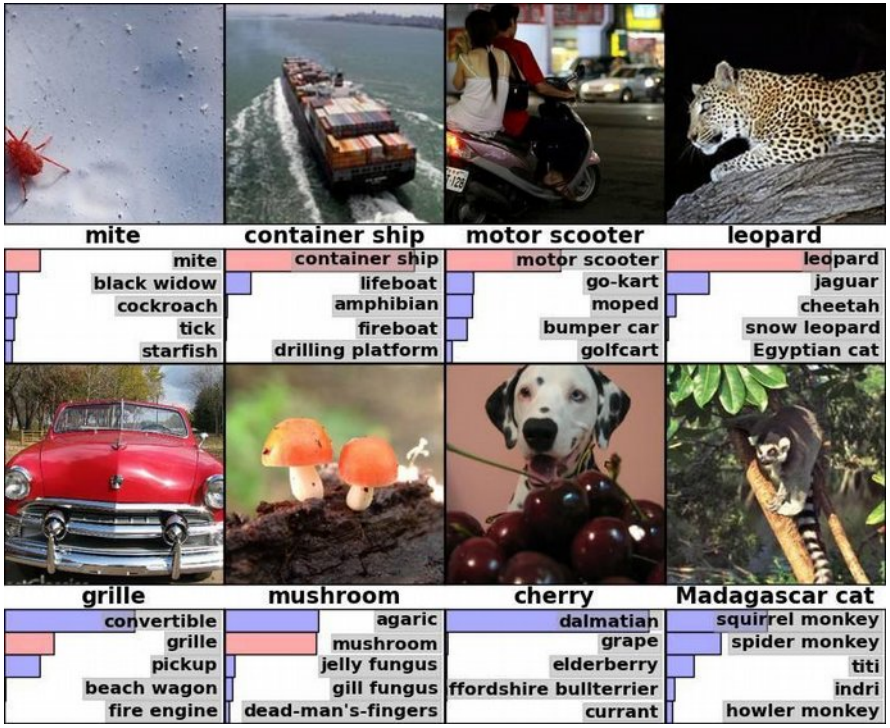
AlexNet

- Alex Krizhevsky – 2012 (Krizhevsky Net)
- Imagenet 2012 Challenge (1000 classes)
- Vencedor - erro 15.3% (2° SIFT Based - 26.2%)
- 1.2 M Treinamento
- 50 K Validação
- 150 K Teste

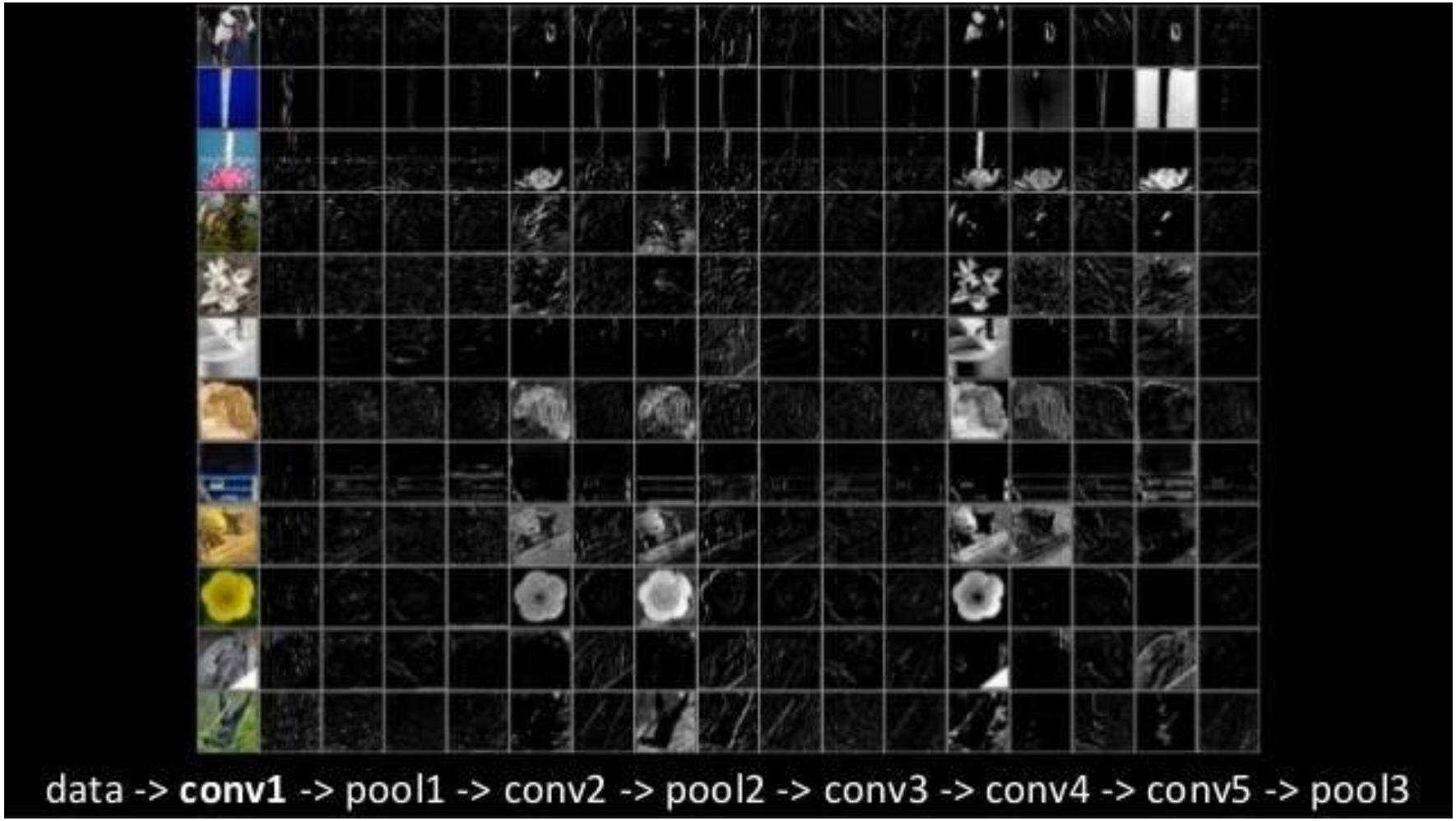


AlexNet

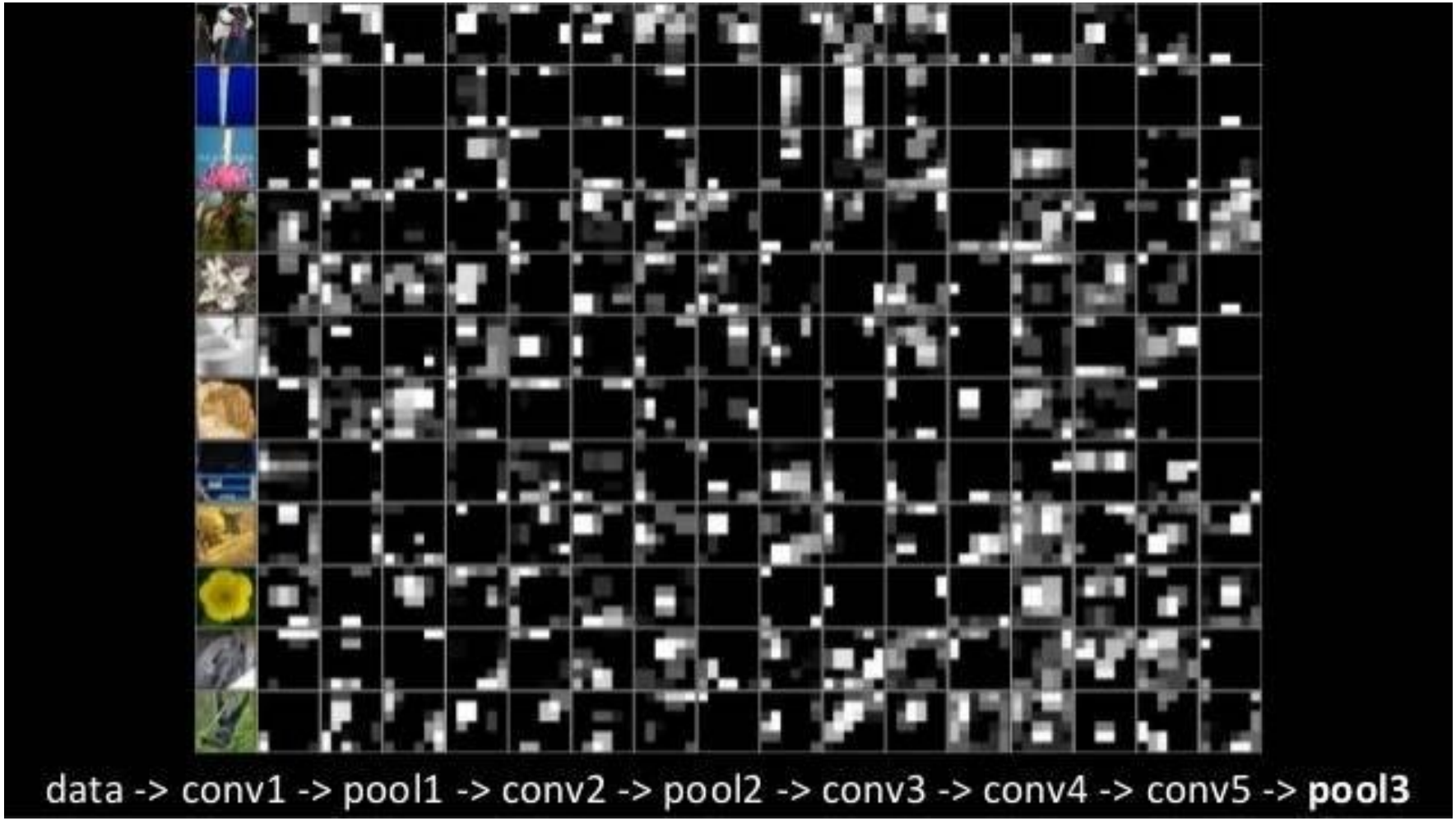
- Neurônios por Camada: ~253k, ~186k, ~64k , ~64k, ~43k, ~4k, ~4k, ~1k
- 60 M de Parâmetros
- Treinamento: 6 dias, 2 NVidia GTX 580 3GB



AlexNet



AlexNet

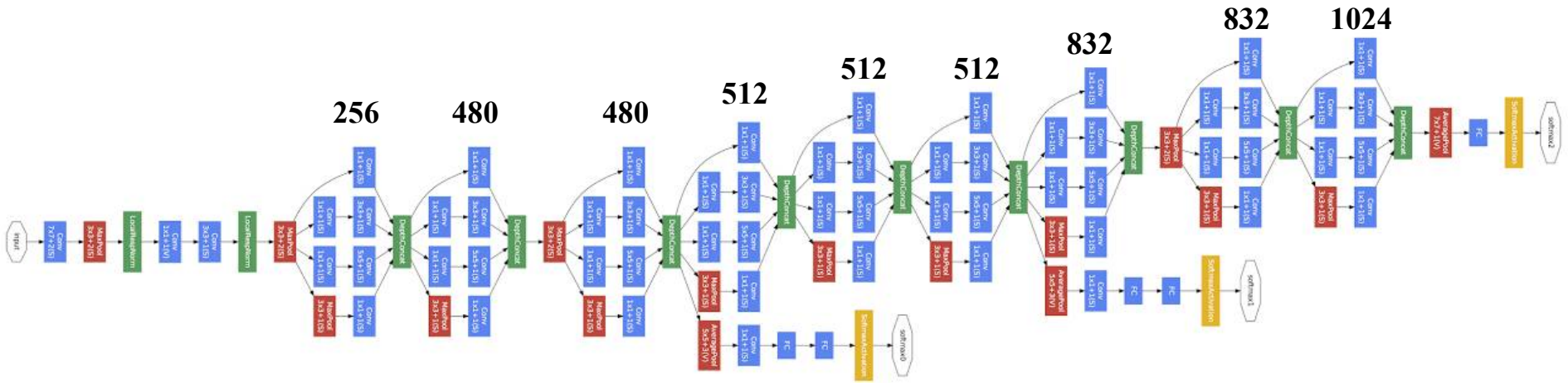


Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

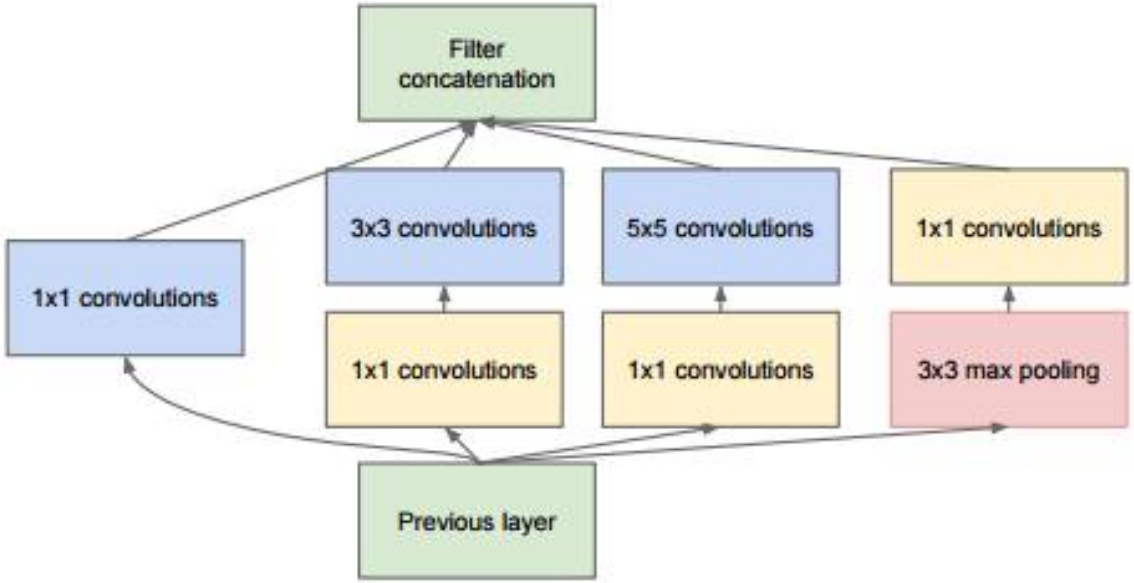
GoogLeNet

- Custo computacional “apenas” 2x mais que AlexNet
- Possui 5 camadas de convolução seguidas por pooling
- Conceito de inceptions (redes dentro de redes)
- Imagenet 2014 Challenge (1000 classes)
  - 1.2 M Treinamento, 50 K Validação, 100 K Teste
- Vencedor - erro 6.67% (2° e 3° também CNN)



GoogLeNet - Inceptions

- Redes dentro de Redes
- 256 a 1024 filtros



GoogLeNet

- Em 2012, o Google utilizou uma parte da Imagenet, para treinar sua rede. O subconjunto era composto por imagens de cães, aprendendo a extrair características com base somente em... cães!

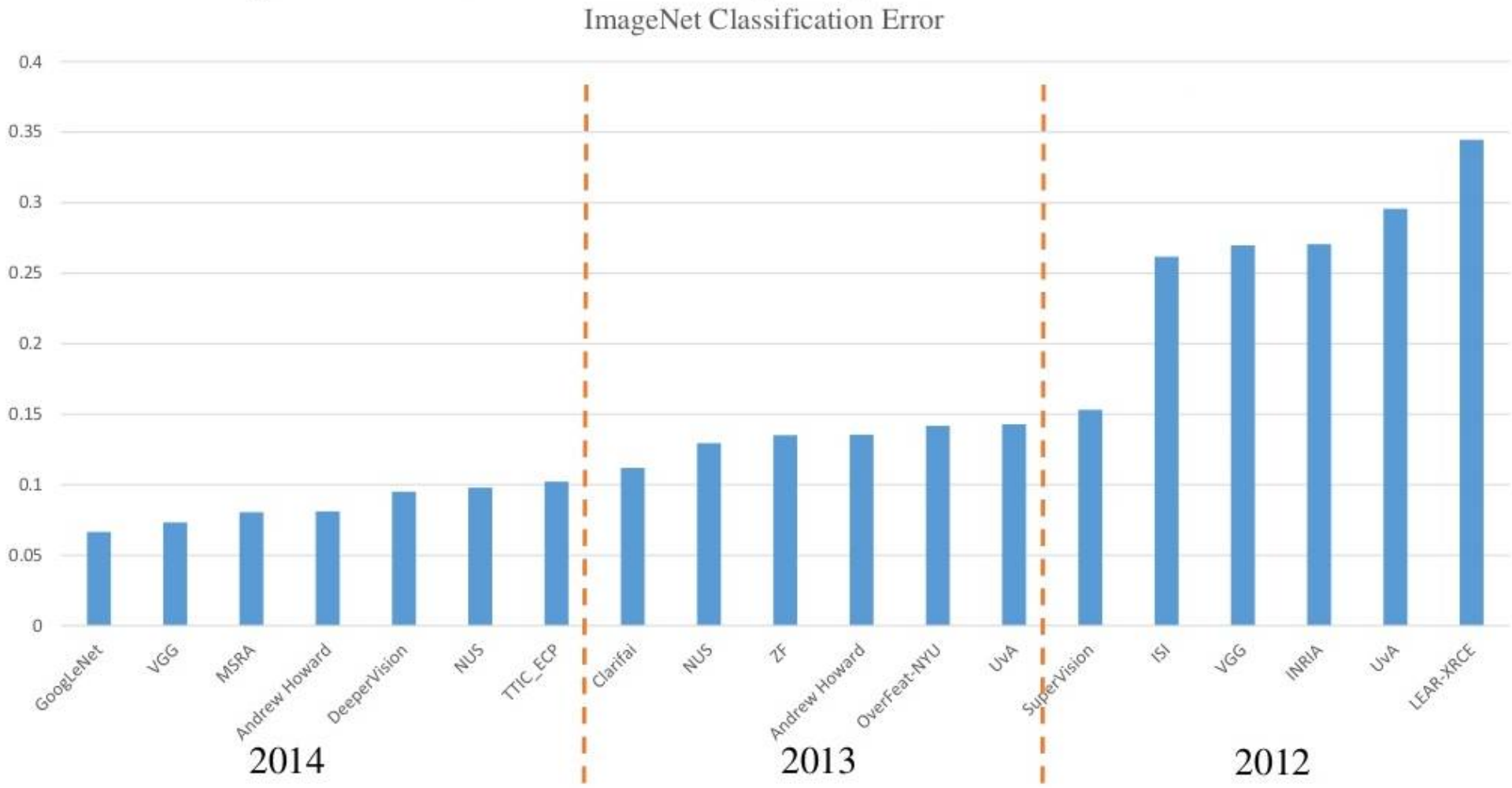



Departamento de Ciência da COMPUTAÇÃO

Resultados ImageNet

UDESC Joinville

- **1000** categories and **1.2** million training images



Mais recentes: WMW **0.02251** (2017), Trimps-Soushen **0.02991** (2016) e MSRA **0.03567** (2015)

Utilizando CNNs

- Construir uma CNN a partir do zero pode ser uma tarefa cara e demorada. Para isso existe série de APIs que visam permitir que as organizações obtenham insights sem necessidade de implementar uma CNN ou necessitar de experiência em Visão Computacional.
- Google Cloud Vision
  - É um serviço de reconhecimento visual do Google que utiliza uma API REST. Esta API é baseada na estrutura TensorFlow de código aberto. Ele detecta rostos e objetos individuais e contém um conjunto de rótulos bastante abrangente.
- IBM Watson Visual Recognition
  - É parte do Watson Developer Cloud e vem com um enorme conjunto de classes embutidas. Foi construído para treinar classes personalizadas com base nas imagens que você fornece. Ele também suporta uma série de características como a detecção de NSFW e OCR, como o Google Cloud Vision.

Utilizando CNNs

- Clarifai
  - É um serviço de reconhecimento de imagens que também utiliza uma API REST. Possui uma série de módulos que ajudam a adaptar seu algoritmo a assuntos específicos, como alimentos, viagens e casamentos.
- Embora esses serviços sejam adequadas para aplicações comuns, melhores resultados podem ser obtidos desenvolvendo uma solução personalizada para tarefas específicas. Existem diversas bibliotecas otimizadas que permite que desenvolvedores se concentrem nos modelos de treinamento: Caffe, Torch, DeepLearning4J, Keras, MxNet, TensorFlow, entre outras.

Desvantagens

- Em relação à quantidade de memória, uma CNN não é muito maior que uma MLP normal. Mas as camadas convolucionais demandam um tempo computacional elevado, usando mais de 67% do tempo.
- As CNNs são aproximadamente 3X mais lentas que uma rede totalmente conectada do mesmo tamanho (mesma quantidade de pesos a ajustar).
- Requerem muitos dados para treinamento (podendo necessitar *Data Augmentation*).