Inteligência Artificial

Redes Neurais Artificiais

Universidade Estadual do Paraná - Unespar

02 de Julho de 2024

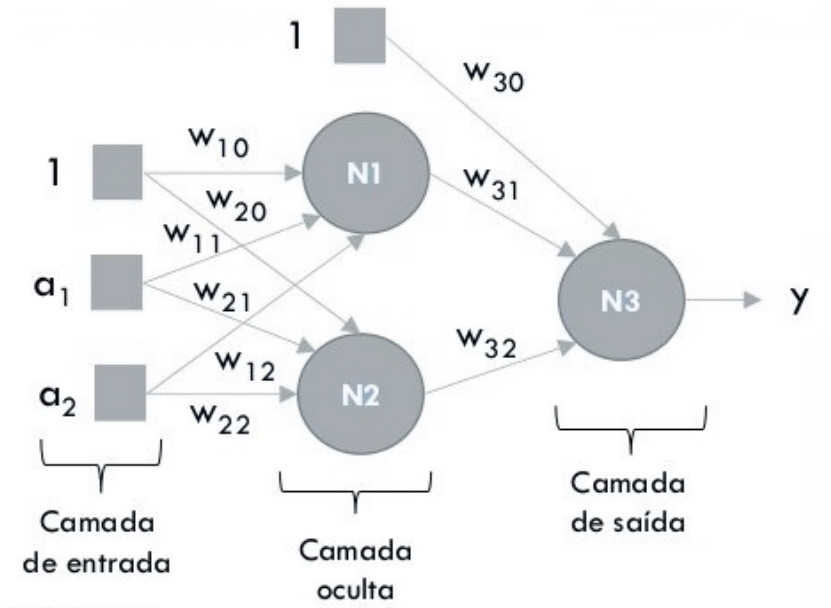
Multi-Layer Perceptron

- A arquitetura de MLP é totalmente conectada;
- Cada neurônio se conecta a todos os outros da camada seguinte;
- Neurônio similar a perceptron;
- Acíclica;
- *Feedfoward*;

Multi-Layer Perceptron

- Os neurônios são organizados em camadas;
  - Cada camada é conectada na seguinte;
  - Os neurônios de uma mesma camada não conectadas entre si;
  - Não existe conexão direta entre a camada de entrada e a camada de saída;
- Como determinar o número de camadas ocultas?
- Ou número de neurônios;
- Definido pela dimensionalidade do espaço de observação (atributos);
- O número de neurônios na saída é determinado pela dimensionalidade da resposta desejada;
- Os neurônios nas camadas ocultas e a quantidade de camadas ocultas geralmente são determinados por experimentação.

Multi-Layer Perceptron



Multi-Layer Perceptron

- Uma MLP pode ser usada para **classificação** ou **regressão**;
- Na regressão se usa o valor da unidade de saída;
- Classificação:
  - Binária:
    - Valores acima de um limiar são interpretados com uma classe e abaixo do limiar como outra lasse.
  - Multi-classe:
    - Como fazer?

Multi-Layer Perceptron

- Uma MLP pode ser usada para **classificação** ou **regressão**;
- Na regressão se usa o valor da unidade de saída;
- Classificação:
  - Binária:
    - Valores acima de um limiar são interpretados com uma classe e abaixo do limiar como outra lasse.
  - Multi-classe:
    - Usar k unidades de saída, uma para cada classe;
    - A classificação é dada para o neurônio com o melhor valor de saída;
    - Softmax;
    - Distribuição de probabilidades.

Multi-Layer Perceptron

- Como treinar uma rede MLP?
  - Perceptron é baseado na correção de erro;
  - Como estimar o erro na camadas ocultas?
- Backpropagation (Rumelhart, Hinnton e William 1986).

Backpropagation

- Retropropagar o erro na camada de saída para as camadas ocultas;
- Três principais etapas:
  - Feed-Forward: Propaga o sinal de entrada até a camada de saída;
  - Feed-Backward: Calcula o erro na camada de saída e propaga pela MLP no sentido inverso;
  - Atualização dos pesos: Calcula os novos pesos considerando os erros propagados na etapa anterior.

Backpropagation

- O backpropagation ajusta os pesos pela minimização da função de custo;
- Gradiente descendente;
  - Otimização iterativo;
  - Utiliza a informação dada pela derivada da função a ser minimizada;
  - Sentido oposto da gradiente;

$$x^{(t+1)} = x^{(t)} - \eta \frac{\partial f(x)}{\partial x}$$

Multi-Layer Perceptron - Feed-Forward

- Calcular os campos induzidos (v_i) e os valores de ativação (z_i) dos neurônios da camada oculta;

$$v_i(t) = \sum_j w_{ij}(t)\boldsymbol{a_j(t)}$$
$$\boldsymbol{z_i} = f(v_i(t))$$

- Calcular os campos induzidos (v_i) e os valores de ativação (y_i) dos neurônios da camada de saída;

$$v_i(t) = \sum_j w_{ij}(t)\boldsymbol{z_j(t)}$$
$$\boldsymbol{y_i} = f(v_i(t))$$

Multi-Layer Perceptron - Feed-Backward

- Calcular a informação de erro nos neurônios de saida (Δ):

$$\Delta_j = \Big(d_j(t) - y_j(t)\Big) f'(v_j(t))$$

- Propagar a informação de erro para os neurônios ocultos (δ):

$$\delta_j = f'(v_j(t)) \sum_i \Delta_i(t) w_{ij}$$

Exemplo de Função e sua Derivada

Considere a função tangente hiperbólica, definida como:

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

A derivada da função tangente hiperbólica é dada por:

$$\frac{d}{dx}\tanh(x) = 1 - \tanh^2(x)$$

Multi-Layer Perceptron

- Atualização dos pesos na camada oculta:

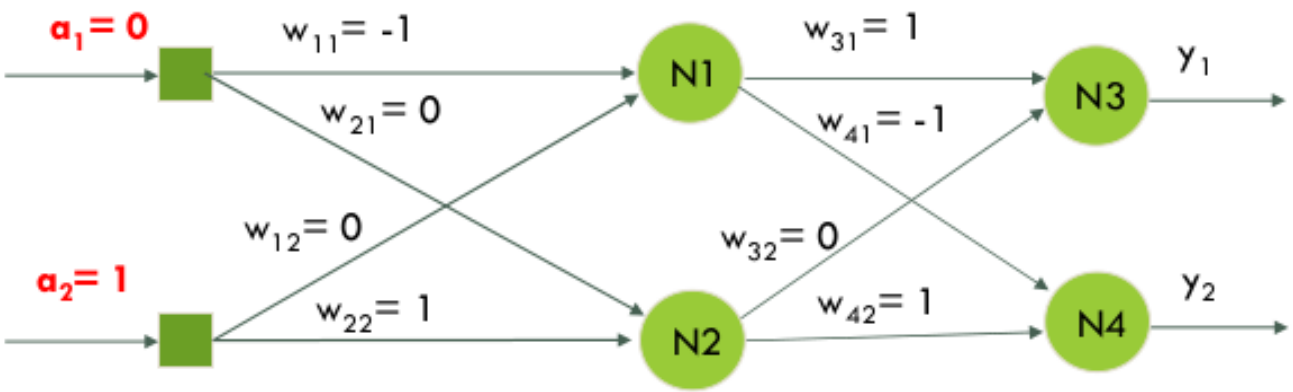
$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_i(t) a_j(t)$$

- Atualização dos pesos na camada de saída:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \Delta_i(t) z_j(t)$$

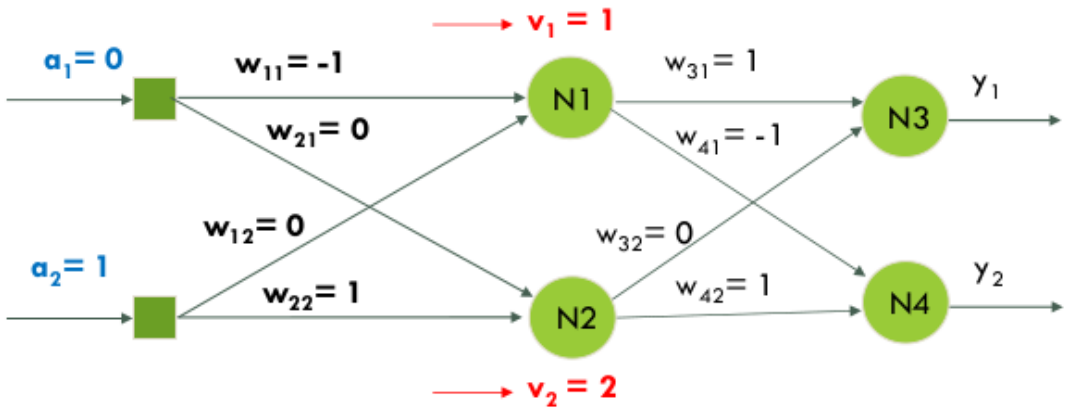
Multi-Layer Perceptron - Exemplo

- Taxa de aprendizagem n: 0,1;
- Valor de entrada 0 e 1;
- Saída desejada 1 e 0;
- Bias: 1;
- Função de ativação:
  - Função de identidade (g(x) = x);
  - g'(x) = ?



Multi-Layer Perceptron - Exemplo

- Calculo do campo induzido (Neurônios Ocultos);



Campos induzidos dos neurônios ocultos:

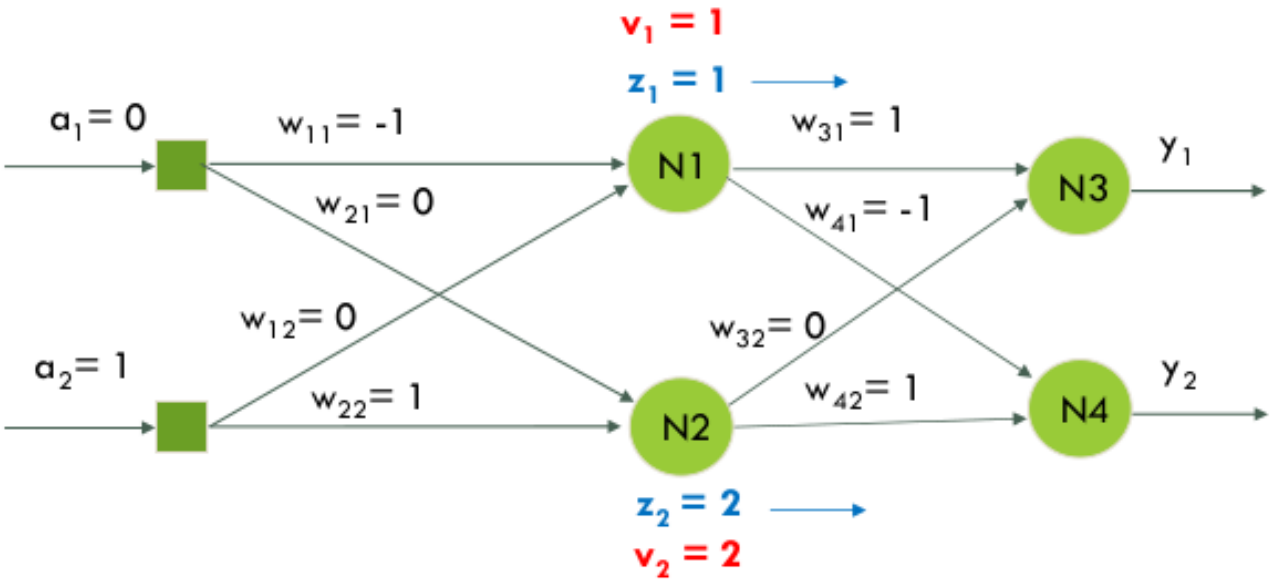
$v_1 = -1*0 + 0*1 + 1*1 = 1$

$v_2 = 0*0 + 1*1 + 1*1 = 2$

$$v_i(t) = \sum_j w_{ij}(t) a_j(t)$$

Multi-Layer Perceptron - Exemplo

- Calculo da saída (Neurônios Ocultos);



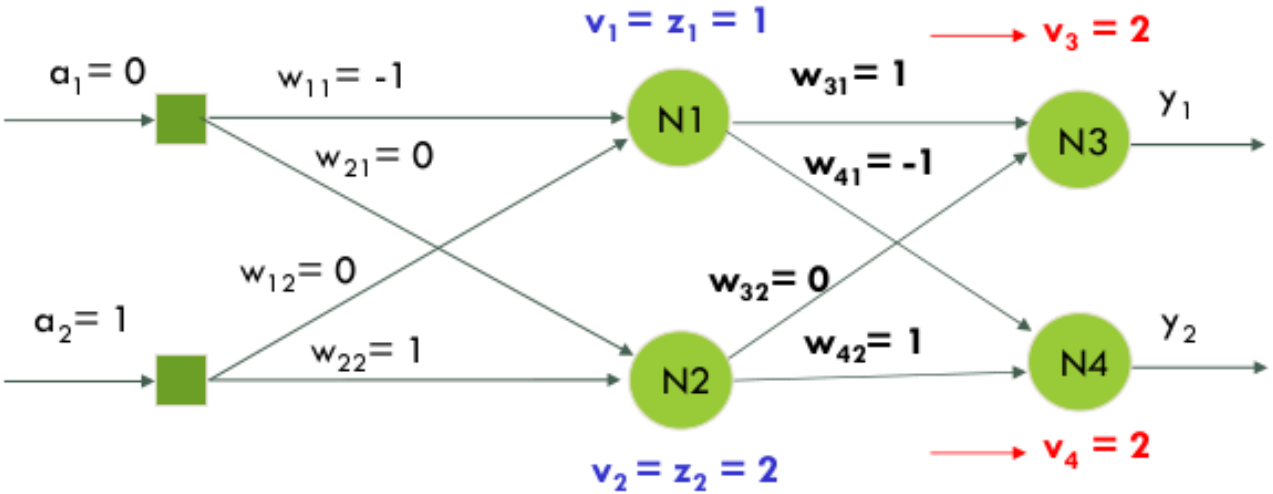
Valores de ativação dos neurônios ocultos:

$$z_1 = g(v_1) = 1$$
$$z_2 = g(v_2) = 2$$

$$z_i = f(v_i(t))$$

Multi-Layer Perceptron - Exemplo

- Calculo do campo induzido (Neurônios Saída);



Campos induzidos dos neurônios de saída:

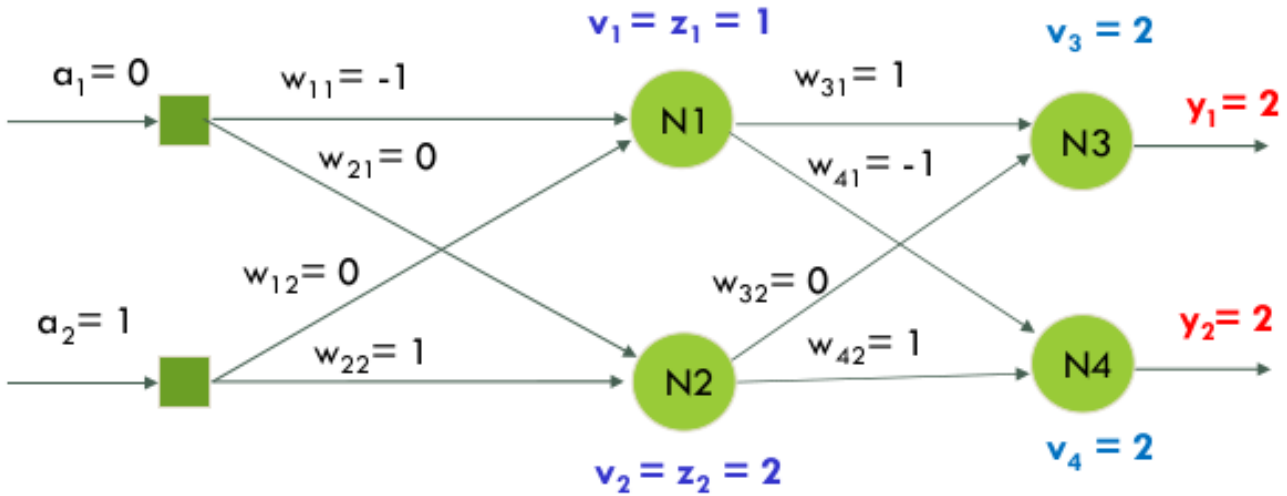
$v_3 = 1*1 + 0*2 + 1*1 = 2$

$v_4 = -1*1 + 1*2 + 1*1 = 2$

$$v_i(t) = \sum_j w_{ij}(t) z_j(t)$$

Multi-Layer Perceptron - Exemplo

- Calculo da ativação dos neurônios de saída;



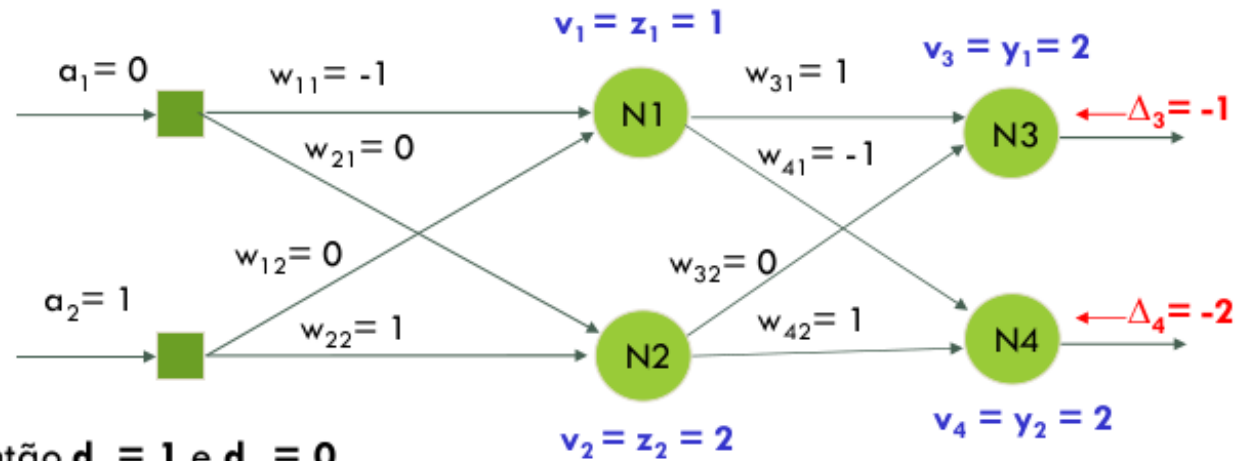
Ativação dos neurônios de saída:

$$y_1 = g(v_3) = 2$$
$$y_2 = g(v_4) = 2$$

$$y_i = f(v_i(t))$$

Multi-Layer Perceptron - Exemplo

- Calculo da informação de erro na camada de saída;



d =[1, 0], então $\mathbf{d_1 = 1}$ e $\mathbf{d_2 = 0}$

Logo, o erro na camada de saída é dado por:

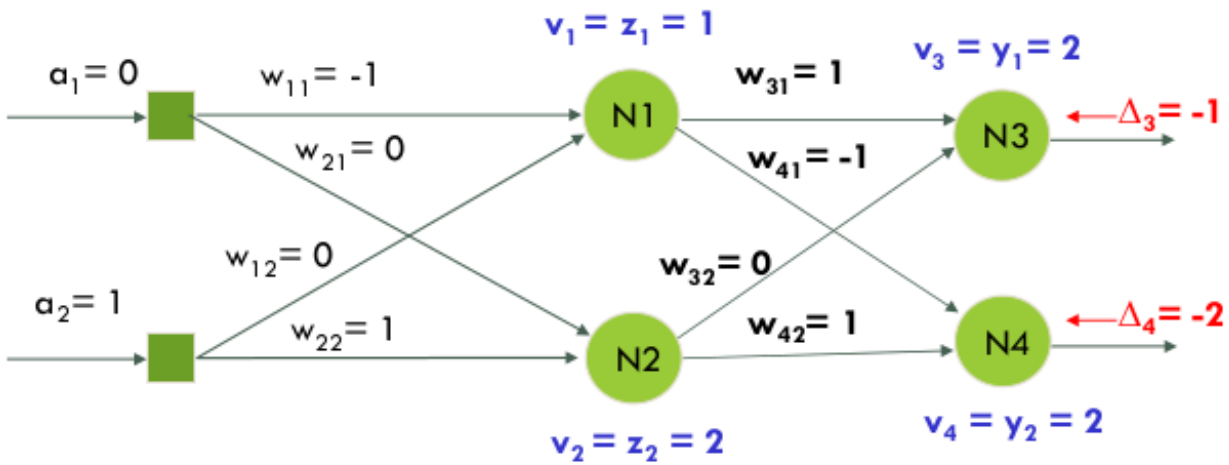
$\Delta_3 = (d_1 - y_1)*g'(v_3) = (1 - 2)*1 = -1$

$\Delta_4 = (d_2 - y_2)*g'(v_4) = (0 - 2)*1 = -2$

$$\Delta_j = \big(d_i(t) - y_i(t)\big) f'(v_j(t))$$

Multi-Layer Perceptron - Exemplo

- Propagação da informação de erro para a camada oculta;



Erro na camada oculta:

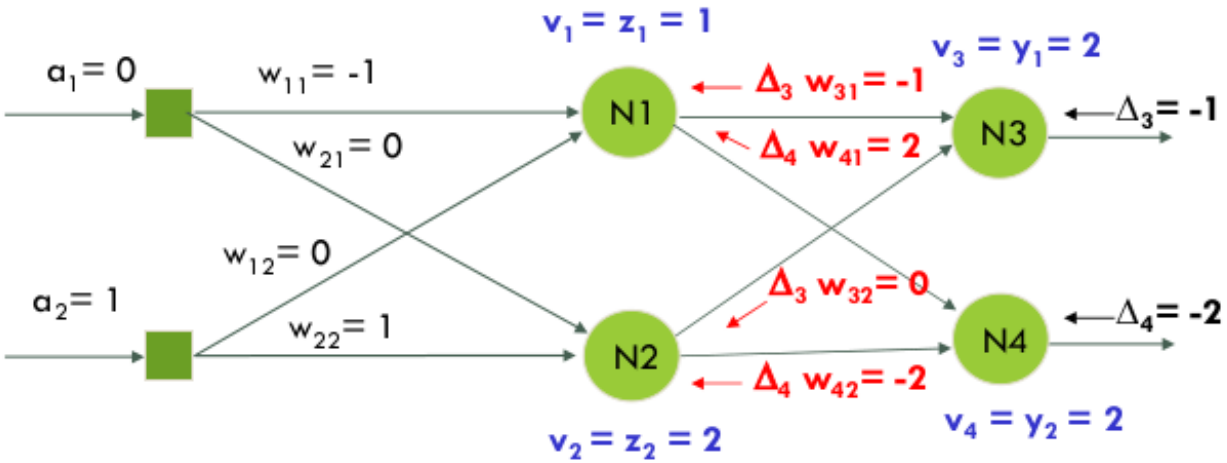
$\delta_1 = g'(v_1)*(\Delta_3*w_{31} + \Delta_4*w_{41})$
$\delta_2 = g'(v_2)*(\Delta_3*w_{32} + \Delta_4*w_{42})$

$\Delta_3 * w_{31} = -1*1 = -1 \quad \Delta_4 * w_{41} = -2*(-1) = 2$
$\Delta_3 * w_{32} = -1*0 = 0 \quad \Delta_4 * w_{42} = -2*1 = -2$

$$\delta_j = f'(v_j(t)) \sum_i \Delta_i(t) w_{ij}$$

Multi-Layer Perceptron - Exemplo

- Propagação da informação de erro para a camada oculta;



Erro na camada oculta:

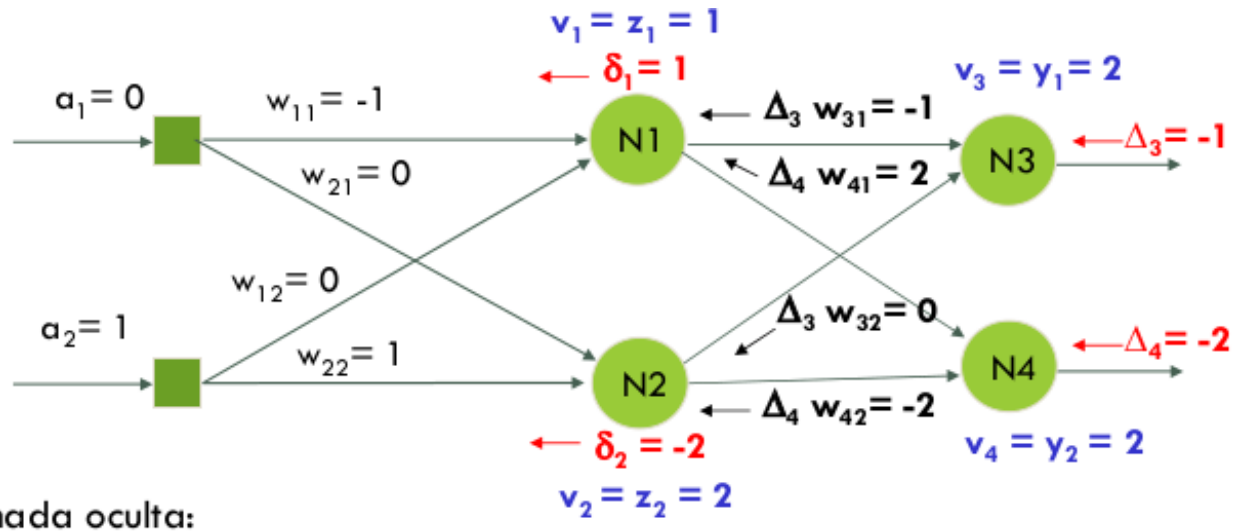
$\delta_1 = g'(v_1)*(\Delta_3*w_{31} + \Delta_4*w_{41})$
$\delta_2 = g'(v_2)*(\Delta_3*w_{32} + \Delta_4*w_{42})$

$\Delta_3 * w_{31} = -1*1 = -1 \quad \Delta_4 * w_{41} = -2*(-1) = 2$
$\Delta_3 * w_{32} = -1*0 = 0 \quad \Delta_4 * w_{42} = -2*1 = -2$

$$\delta_j = f'(v_j(t)) \sum_i \Delta_i(t) w_{ij}$$

Multi-Layer Perceptron - Exemplo

- Propagação da informação de erro para a camada oculta;



Erro na camada oculta:

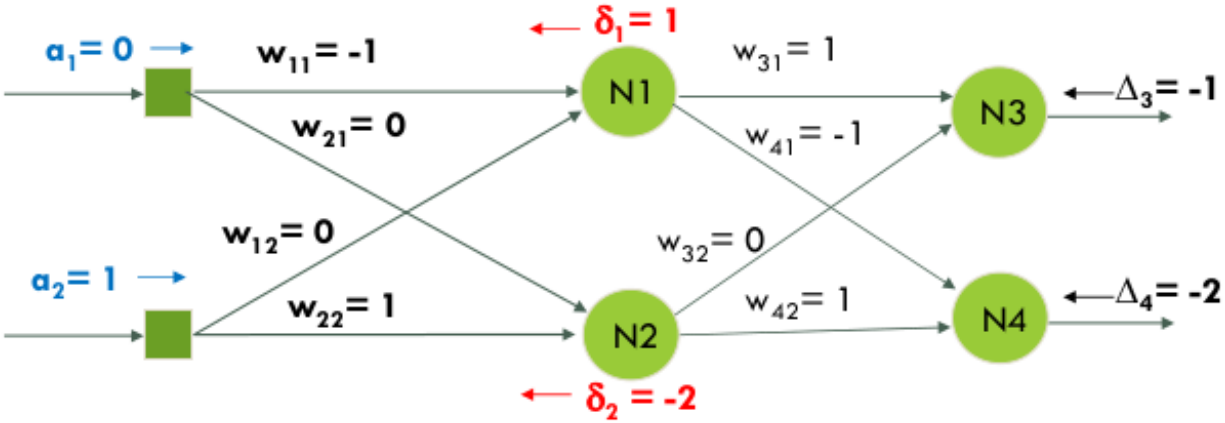
$\delta_1 = g'(v_1)*(\Delta_3*w_{31} + \Delta_4*w_{41}) = 1*((-1) + 2) = 1$

$\delta_2 = g'(v_2)*(\Delta_3*w_{32} + \Delta_4*w_{42}) = 1*(0 + (-2)) = -2$

$$\delta_j = f'(v_j(t)) \sum_i \Delta_i(t) w_{ij}$$

Multi-Layer Perceptron - Exemplo

- Atualização dos pesos da camada oculta;

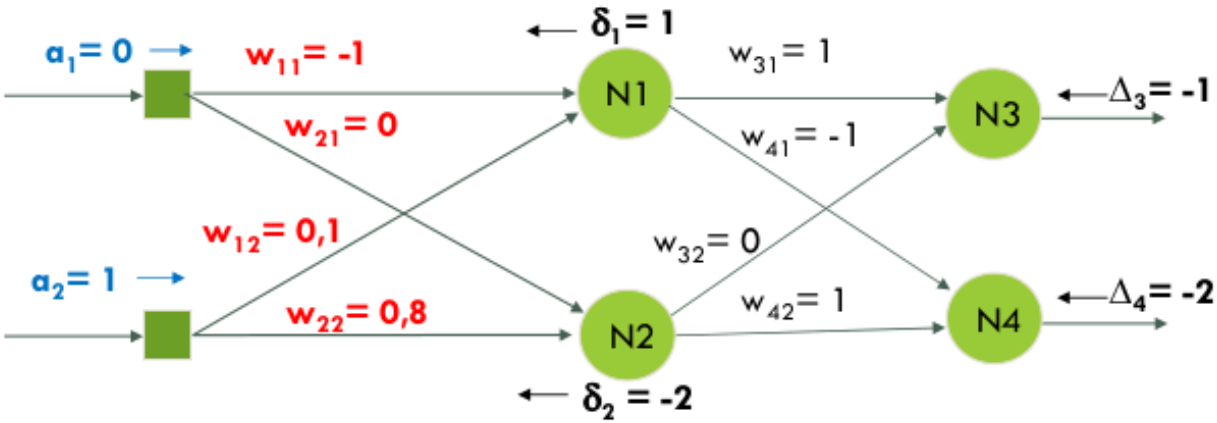


$$w_{11}(t+1) = w_{11}(t) + \eta\delta_1(t)a_1(t) = -1 + 0{,}1 * 1 * 0 = -1$$
$$w_{21}(t+1) = w_{21}(t) + \eta\delta_2(t)a_1(t) = 0 + 0{,}1 * (-2) * 0 = 0$$
$$w_{12}(t+1) = w_{12}(t) + \eta\delta_1(t)a_2(t) = 0 + 0{,}1 * 1 * 1 = 0{,}1$$
$$w_{22}(t+1) = w_{22}(t) + \eta\delta_2(t)a_2(t) = 1 + 0{,}1 * (-2) * 1 = 0{,}8$$

$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_i(t)a_j(t)$$

Multi-Layer Perceptron - Exemplo

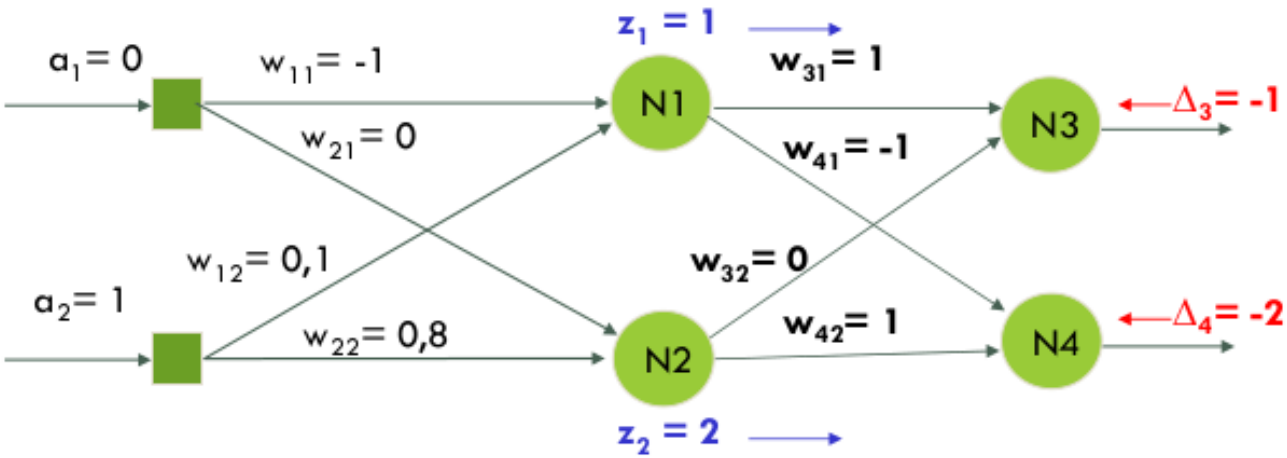
- Atualização dos pesos da camada oculta;



$$w_{ij}(t+1) = w_{ij}(t) + \eta\delta_i(t)a_j(t)$$

Multi-Layer Perceptron - Exemplo

- Atualização dos pesos da camada saída;

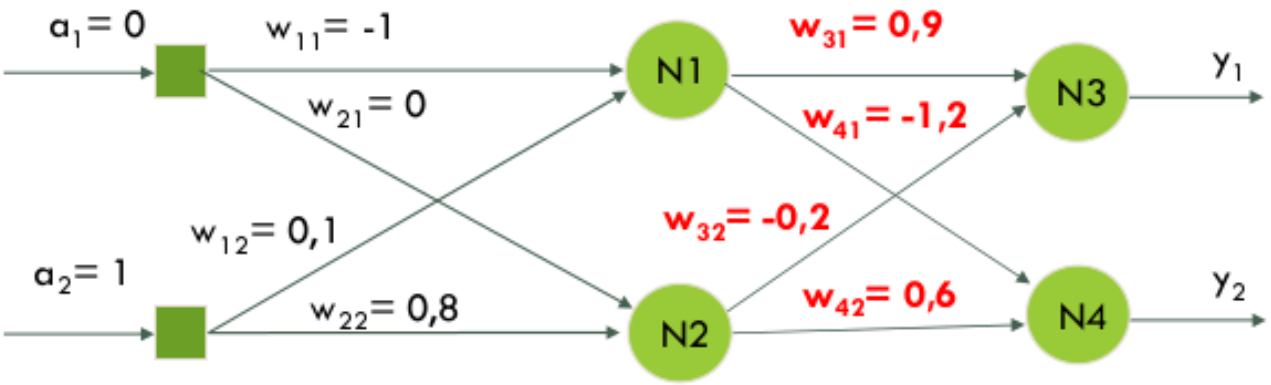


$$w_{31}(t+1) = w_{31}(t) + \eta\Delta_3(t)z_1(t) = 1 + 0,1 * (-1) * 1 = 0,9$$
$$w_{41}(t+1) = w_{41}(t) + \eta\Delta_4(t)z_1(t) = -1 + 0,1 * (-2) * 1 = -1,2$$
$$w_{32}(t+1) = w_{32}(t) + \eta\Delta_3(t)z_2(t) = 0 + 0,1 * (-1) * 2 = -0,2$$
$$w_{42}(t+1) = w_{42}(t) + \eta\Delta_4(t)z_2(t) = 1 + 0,1 * (-2) * 2 = 0,6$$

$$w_{ij}(t+1) = w_{ij}(t) + \eta\Delta_i(t)z_j(t)$$

Multi-Layer Perceptron - Exemplo

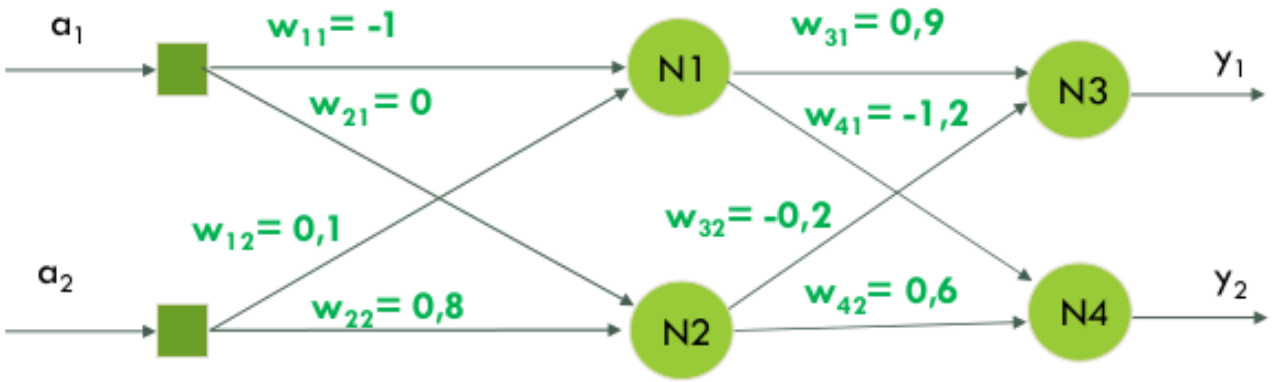
- Atualização dos pesos da camada de saída;



$$w_{ij}(t+1) = w_{ij}(t) + \eta \Delta_i(t) z_j(t)$$

Multi-Layer Perceptron - Exemplo

- Configuração da rede após o ciclo;



Multi-Layer Perceptron

- Ajuste de peso feito por exemplo apresentado à rede;
- Atualização on-line;
- Podemos optar por:
  - Época;
  - Lotes;
- Repetido diversas vezes;
- Erro acumulado;
- Número máximo de épocas;
- Estabilização de pesos;
- Ajustes.

Reconhecimento de Padrões

- Pouca dificuldade para os seres humanos;
- Até mesmo animais;
- Grande desafio para a tecnologia moderna;
- Reconhecimento de padrões (década de 60);
- Detecção de formas simples;
- Regularidades significativas em ambientes ruidosos.

Reconhecimento de Padrões

- Interesse em projetar e construir autômatos;
- Realizar tarefas com habilidades comparáveis à performance humana;
- Posição;
- Estímulos;
- Deslocamento;
- Rotação;
- Perspectiva;
- Oclusão parcial.

Reconhecimento de Padrões

- Áreas de aplicações:
  - Reconhecimento automático da fala;
  - Reconhecimento da escrita;
  - Compreensão da fala;
  - Compreensão de imagens;
  - Processamento da linguagem natural;
- Defesa: Reconhecimento e orientação automático de alvos;
- Medicina: Análise de imagens, classificação de doenças;
- Veículos: Controladores de automóveis, aviões e barcos;
- Polícia e investigação: Detecção criminal a partir da fala, escrita manual, fotografias.

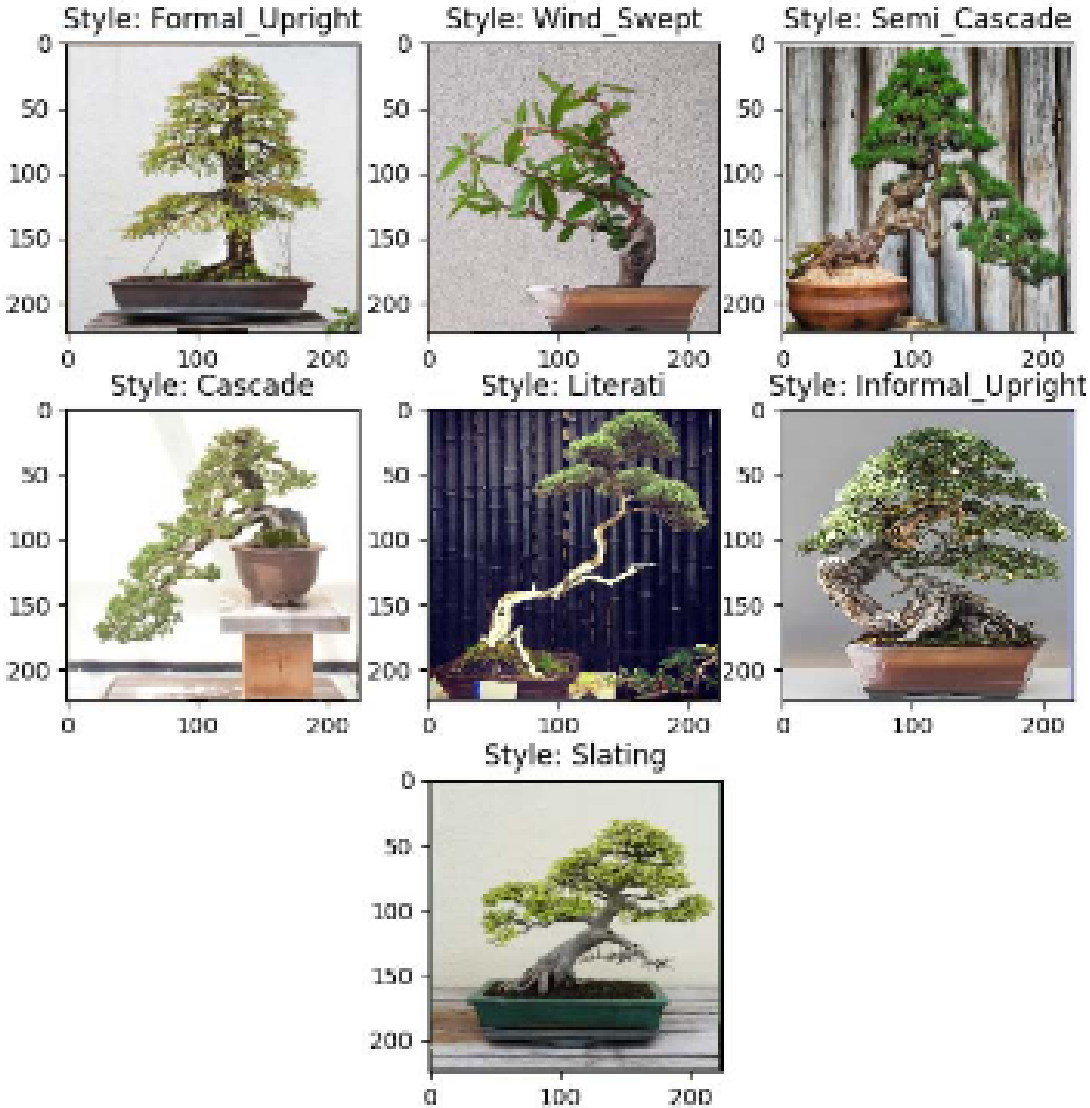
Técnicas para o Reconhecimento de Padrões

- Transfer Learning;
- Utilizando redes neurais convolucional;
- Reutilização de um modelo pré-treinado em um novo problema;
- Utilizar uma rede neural treinada em outro conjunto de dados;
- Geralmente maior;
- Por que usar?

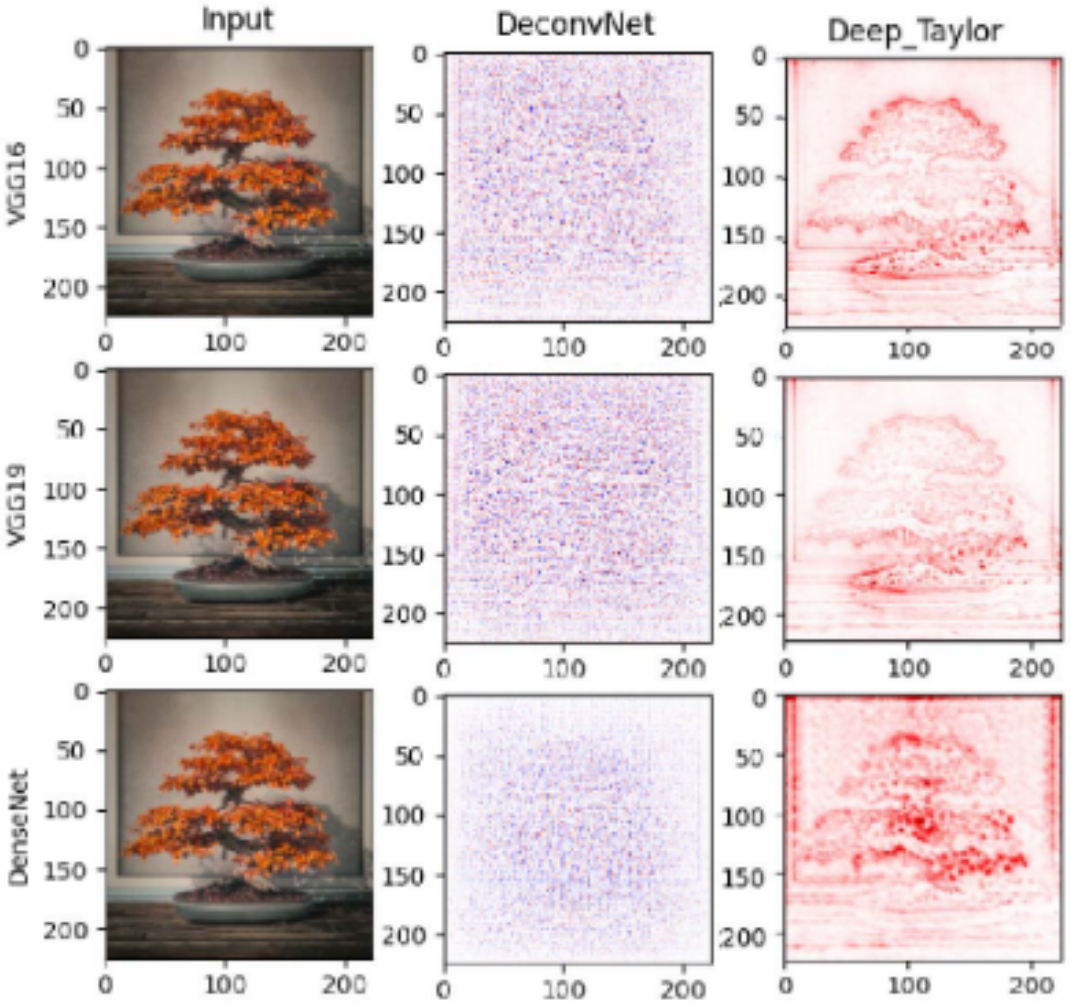
Técnicas para o Reconhecimento de Padrões

- Transfer Learning;
- Utilizando redes neurais convolucional;
- Reutilização de um modelo pré-treinado em um novo problema;
- Utilizar uma rede neural treinada em outro conjunto de dados;
- Geralmente maior;
- Por que usar?
  - Obter conjunto de dados grande o suficiente;
  - CNN são muito caras para serem treinadas do zero;
  - Demoram semanas utilizando GPU's.

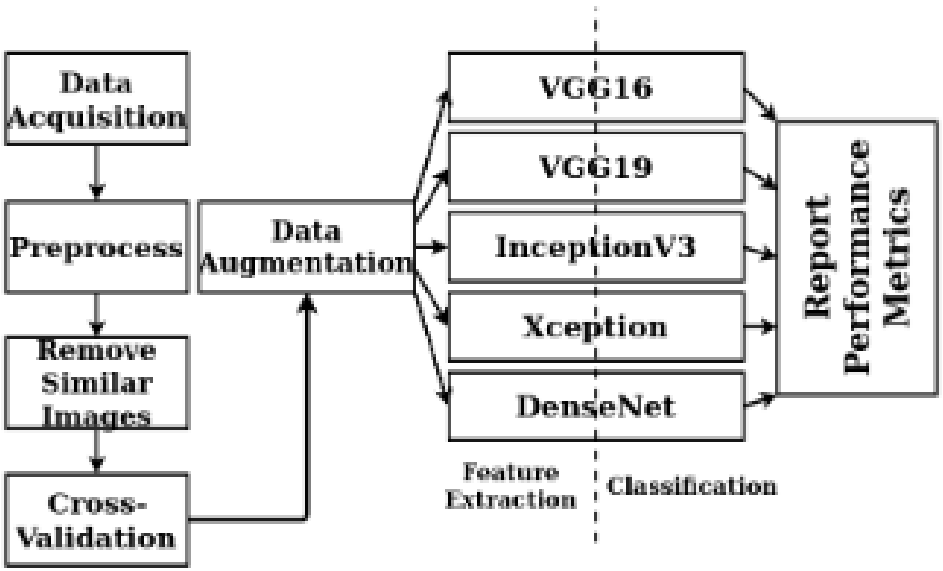
- Utilizando conjunto de dados "grandes";
- 10000 imagens;
- Não é suficiente para treinar CNN;
- Problema de overfitting;
- Keras (Pyton);
- VGG16;
- VGG19;
- DenseNet.

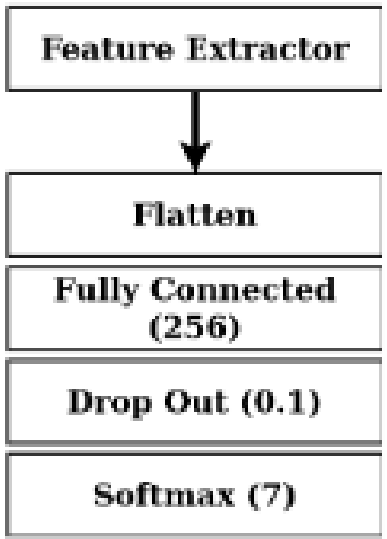

Style: Formal_Upright
Style: Wind_Swept
Style: Semi_Cascade
Style: Cascade
Style: Literati
Style: Informal_Upright
Style: Slating

Técnicas para o Reconhecimento de Padrões



Técnicas para o Reconhecimento de Padrões



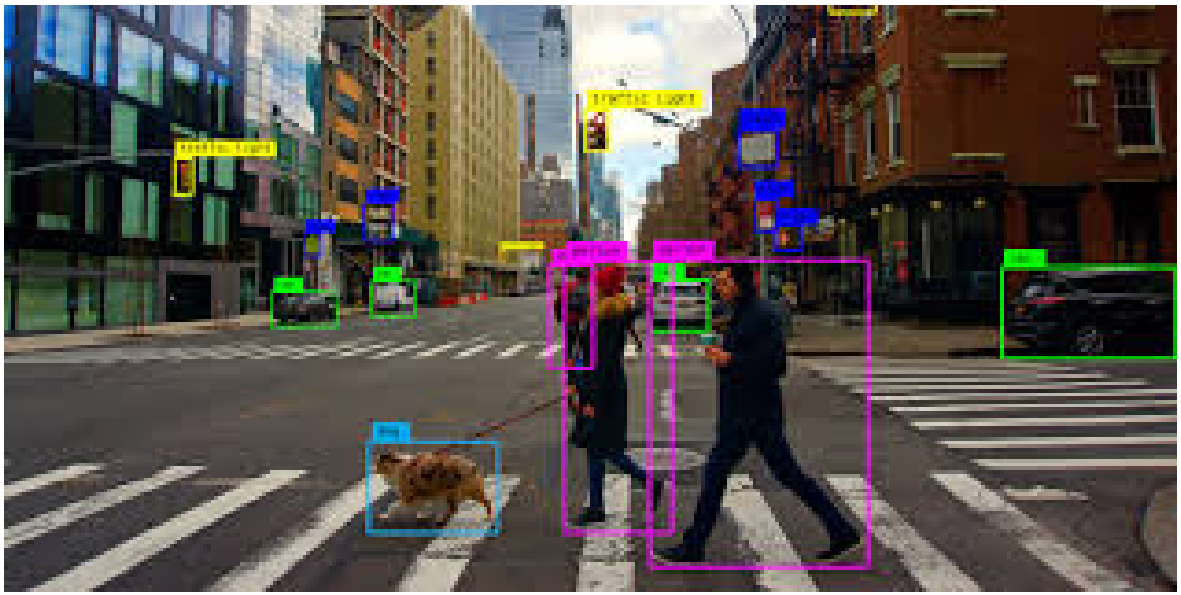

Feature Extractor
Flatten
Fully Connected (256)
Drop Out (0.1)
Softmax (7)

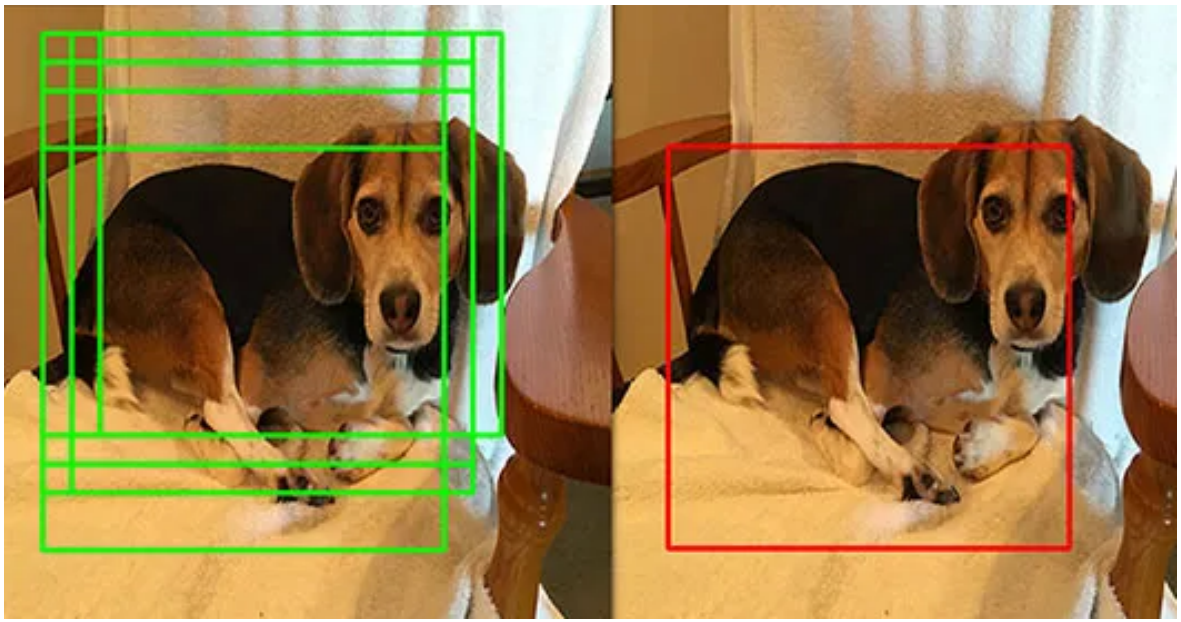
Técnicas para o Reconhecimento de Padrões

- YOLO (You only look once);
- Rápida detecção de objetos;
- Estado da arte em detecção de objetos em tempo real;
- Tem uma única passada;
- Utilizando uma rede neural convolucional;
- Extrator de caracterísitcas (Features).

Técnicas para o Reconhecimento de Padrões

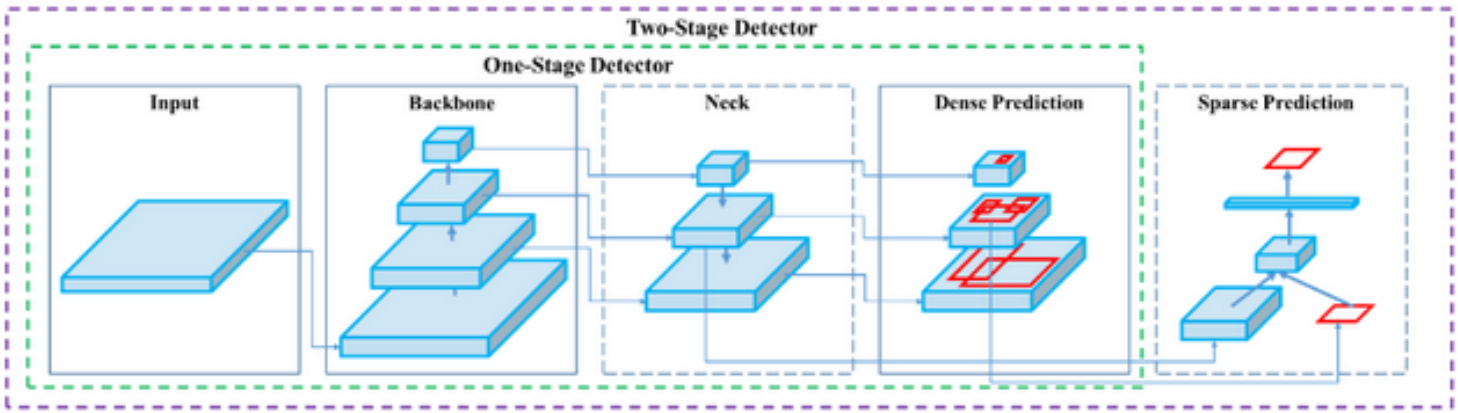
- R-CNN (Region-based Convolutional Neural Network);
- Faster R-CNN;
- YOLO;
- Velocidade;
- Divisão da imagem em grades (Grid de S x S): 13 x 13, 19 x 19...
- Predição das boundings boxes: Caso haja mais de um objeto naquela célula, pontuação de confiança;
- Combinação de previsões: Baseado nas pontuações de confiança.

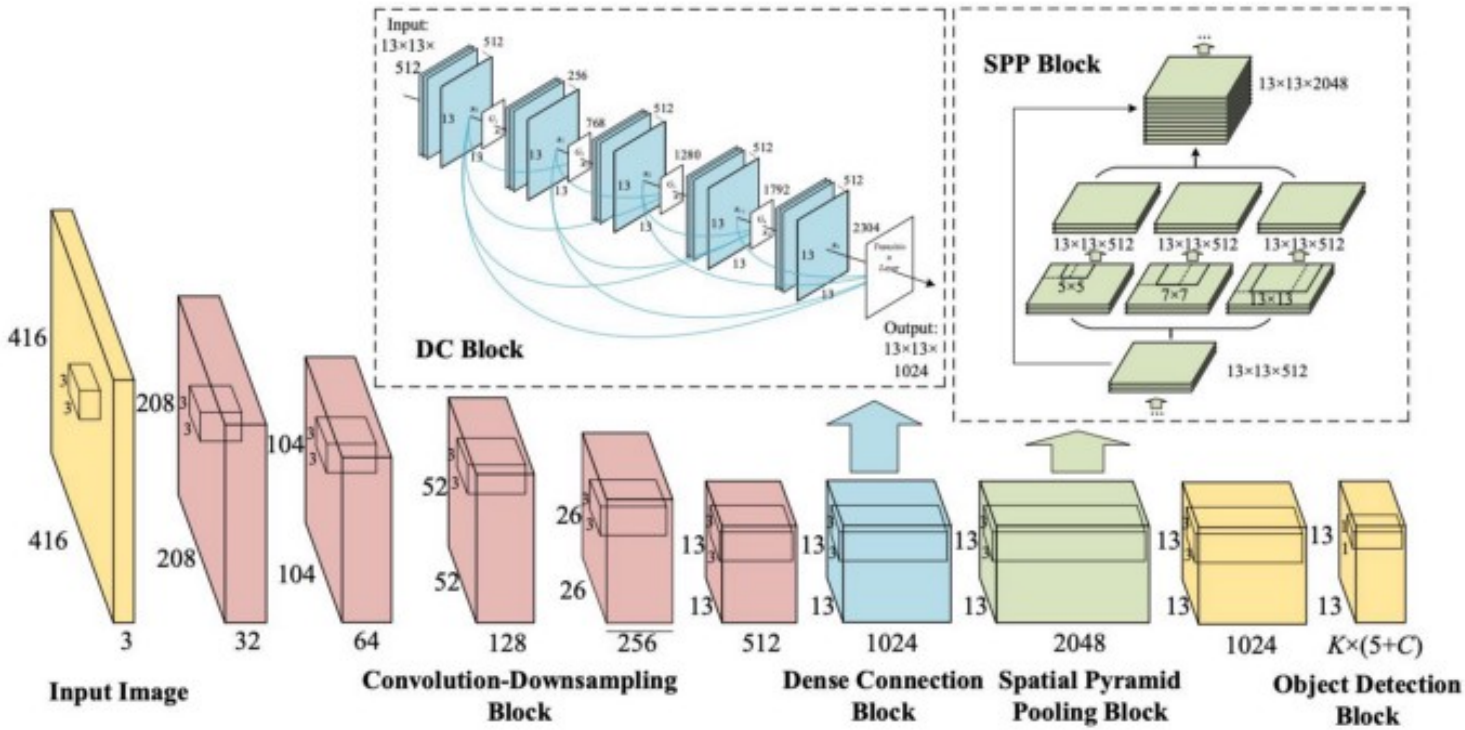




- Necessita uma rede neural profunda;
- Darknet;
- Framework desenvolvido por Joseph Redmon;
- Open source;
- Linguagem C;
- Possui suporte para GPU;
- Python.

Técnicas para o Reconhecimento de Padrões




Input:
13×13×
512
DC Block
Output:
13×13×
1024
SPP Block
13×13×2048
13×13×512
13×13×512
13×13×512
5×5
7×7
13×13
13×13×512
416
416
3
208
208
32
104
104
64
52
52
128
26
26
256
13
13
512
13
13
1024
13
13
2048
13
13
1024
13
13
K×(5+C)
Input Image
Convolution-Downsampling Block
Dense Connection Block
Spatial Pyramid Pooling Block
Object Detection Block

Obrigado!
Dúvidas?

Guilherme Henrique de Souza Nakahata

guilhermenakahata@gmail.com

https://github.com/GuilhermeNakahata/UNESPAR-2024