UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MARLON HENRIQUE SCALABRIN

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PROFUNDAS NA DETECÇÃO DE PLACAS DE PARE

FRANCISCO BELTRÃO

2019

MARLON HENRIQUE SCALABRIN

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PROFUNDAS NA DETECÇÃO DE PLACAS DE PARE

Trabalho de Conclusão de Curso de Especialização apresentado ao Curso de Especialização em Métodos Matemáticos Aplicados da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do título de Especialista em Métodos Matemáticos Aplicados.

Orientador: Prof. Dr. Jeconias Rocha Guimarães

FRANCISCO BELTRÃO

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO 001

Trabalho de Conclusão de Curso de Especialização

APLICAÇÃO DE REDES NEURAIS ARTIFICIAIS PROFUNDAS NA DETECÇÃO DE PLACAS DE PARE

por

MARLON HENRIQUE SCALABRIN

Trabalho de Conclusão de Curso de Especialização apresentado às 15h30 do dia 26 de outubro de 2019, como requisito parcial para obtenção do grau de especialista em Métodos Matemáticos Aplicados, da Universidade Tecnológica Federal do Paraná, Campus Francisco Beltrão. O(a) candidato(a) foi arguido(a) pela Banca Avaliadora composta pelos professores que abaixo assinam este Termo. Após deliberação, a Banca Avaliadora considerou o trabalho Aprovado.

_____
Prof. Dr. Jeconias Rocha Guimarães
Professor(a) Orientador(a)

_____
Prof. Dr. Vilmar Steffen
Membro da Banca

_____
Prof. Msc. Dionisio Raony de S. Ribeiro
Membro da Banca

_____
Prof. Dr. Vilmar Steffen
Responsável pela Coordenação do CEMMA
Curso de Especialização em Métodos Matemáticos Aplicados

A FOLHA DE APROVAÇÃO ORIGINAL (ASSINADA) ENCONTRA-SE NA COORDENAÇÃO DO CURSO DE ESPECIALIZAÇÃO EM MÉTODOS MATEMÁTICOS APLICADOS.

Dedico este trabalho a Deus, que me deu forças para vencer todas as dificuldades. À minha família que esteve presente em todos os momentos importantes da minha vida. Em especial, dedico à minha esposa, Ediane, que me incentivou e me acompanhou neste desafio.

AGRADECIMENTOS

A Deus. Aos meus pais. Aos meus irmãos. Aos meus familiares. Ao professor Jeconias. Aos meus colegas. E o mais importante, a minha pitanguinha (Ediane). Obrigado.

Todas as coisas são números.
(Pitágoras, 569 A.C. - 475 A.C.)

Os números dominam o mundo.
(Platão, 428 A.C. - 347 A.C.)

SCALABRIN, Marlon Henrique. **Aplicação de Redes Neurais Artificiais Profundas na Detecção de Placas De Pare**. 2019. 53p. Monografia (Especialização em Métodos Matemáticos Aplicados) - Universidade Tecnológica Federal do Paraná. Francisco Beltrão, 2019.

RESUMO

Sistemas inteligentes estão cada vez mais presentes em nosso dia a dia, nos auxiliando em diversas tarefas. Dentre vários, destacam-se aqueles de identificação visual usando redes neurais artificiais. Neste sentido, aplicamos técnicas de inteligência artificial no desenvolvimento de uma solução em visão computacional para detecção de objetos em imagens, com o foco em placas de pare. O formalismo de redes neurais artificiais foi empregado para desenvolver um classificador capaz de integrar características extraídas de imagens, realizando a classificação e localização de objetos nelas inseridos. Utilizou-se uma Rede Neural Artificial Convolucional Multicamada com técnicas de aprendizagem profunda. A rede neural projetada segue padrões definidos no projeto YOLO (*You Only Look Once*), aproveitando-se de suas capacidades de aprendizagem rápida e amplo reconhecimento de padrões em imagens, alcançando uma boa taxa de sucesso. Os experimentos permitiram a detecção de placas de pare de forma invariável ao espaço em imagens complexas.

Palavras-chave: Redes Neurais Artificiais. Aprendizagem Profunda. YOLO.

SCALABRIN, Marlon Henrique. **Application of Deep Artificial Neural Networks for Detecting Stop Signs**. 2019. 53p. Monograph (Specialization in Applied Mathematical Methods) - Federal Technological University of Paraná. Francisco Beltrão, 2019.

ABSTRACT

Intelligent systems are increasingly present in our daily lives. Among many, stands out the ones based on artificial neural networks. In this way, we apply artificial intelligence techniques in the development of a computer vision solution for object detection in images, with focus on stop signs. Artificial neural network was used to develop a classifier capable of integrating features extracted from images, performing the classification and localization of objects in them. A multilayered convolutional artificial neural network with deep learning techniques was used. The projected neural network follows procedure set in the YOLO (You Only Look Once) project, taking advantage of its fast learning capabilities and extensive image pattern recognition, achieving a good success rate. Experiments allowed detection of stop signs invariably to space in complex images.

Keywords: Artificial Neural Networks. Deep learning. YOLO.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

ELU	*Exponential Linear Unit*
MLP	*Multilayer Perceptron*
PMC	Perceptrons multicamada
ReLU	*Rectified Linear Unit*
RNA	Redes Neurais Artificiais
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

A principal motivação para detectar placas de trânsito advém do fato que veículos autônomos estão tornando-se mais presentes e a detecção de placas é um processo importante neste contexto, podendo, também, ser uma forma de auxiliar motoristas de veículos convencionais.

Uma das maneiras de abordar o problema do reconhecimento automático de objetos em imagens é decompor a tarefa em subproblemas que detectam partes dos objetos na imagem e então combinam as soluções parciais para obter a solução do problema original. De um modo geral, a detecção automática de objetos em imagens exige a utilização de métodos para detecção de formas geométricas, qualificação de texturas, agrupamento de regiões e processamento digital de sinais para extrair informações relevantes. Estas características podem ser processadas e classificadas utilizando-se uma Rede Neural Artificial (RNA).

Neste trabalho foi empregada uma rede neural artificial convolucional profunda com 53 camadas. As redes neurais artificiais convolucionais profundas simplificam esse processo de forma que a própria rede neural se torna capaz de aprender (idealmente) todos os padrões necessários para classificação.

As redes neurais necessitam passar por um processo de aprendizado para determinação dos pesos das interconexões entre os neurônios, para que possam assim realizar sua função. O algoritmo de retro-propagação de erro é um dos métodos mais utilizados para treinamento de RNAs (HAYKIN, 2001).

O estudo está organizado da seguinte forma: no Capítulo 2 são apresentados os Objetivos; nos Capítulos 3 e 4 faz-se uma fundamentação teórica sobre Aprendizagem de Máquina e Redes Neurais; no Capítulo 5 é descrita a implementação no modelo YOLO; no Capítulo 6 é apresentada a metodologia; e nos Capítulos 7 e 8 são apresentados resultados e conclusões.

1.1 OBJETIVOS

1.1.1 Objetivos Específicos

Reconhecer placas de trânsito do tipo “pare” em imagens fazendo o uso de redes neurais profundas.

1.1.2 Objetivos Específicos

- Reunir e classificar imagens de placas de pare no padrão nacional;
- Implementar uma estrutura de rede neural inspirada no YOLOv3;
- Treinar uma rede neural multicamada com as imagens classificadas;
- Verificar o classificador com imagens aleatoriamente selecionadas.

2 APRENDIZAGEM DE MÁQUINA

A ideia chave do processo de aprendizagem de máquina é que computadores realizem processamento de forma a aprender com dados conhecidos e generalizar para dados novos.

Segundo Mitchell (1997) o aprendizado de máquina pode ser separado em:

- Aprendizado supervisionado, no qual o treinamento é realizado com dados previamente classificados;
- Aprendizado não-supervisionado, no qual ocorre a clusterização de dados criando agrupamentos de dados ainda não classificados; e
- Aprendizado semi-supervisionado, no qual é realizado o uso de ambos dados, classificados e não classificados.

O processo de aprendizagem supervisionado é capaz de realizar classificação quando as saídas esperadas são valores discretos, bem como regressão quando as saídas esperadas são valores contínuos no domínio dos reais.

Com o avanço da tecnologia os processos de aprendizagem de máquina estão se tornando cada vez mais complexos permitindo o desenvolvimento de técnicas mais robustas, como a aprendizagem profunda.

Aprendizagem Profunda (do inglês *Deep-Learning*) refere-se ao subconjunto de técnicas relacionadas a aprendizagem de máquina baseadas em redes neurais artificiais que necessitam de um alto poder computacional. Estas técnicas estão ganhando popularidade com o avanço das tecnologias para processamento gráfico (GPU – do inglês *Graphics Processing Unit*) (SCHMIDHUBER, 2015).

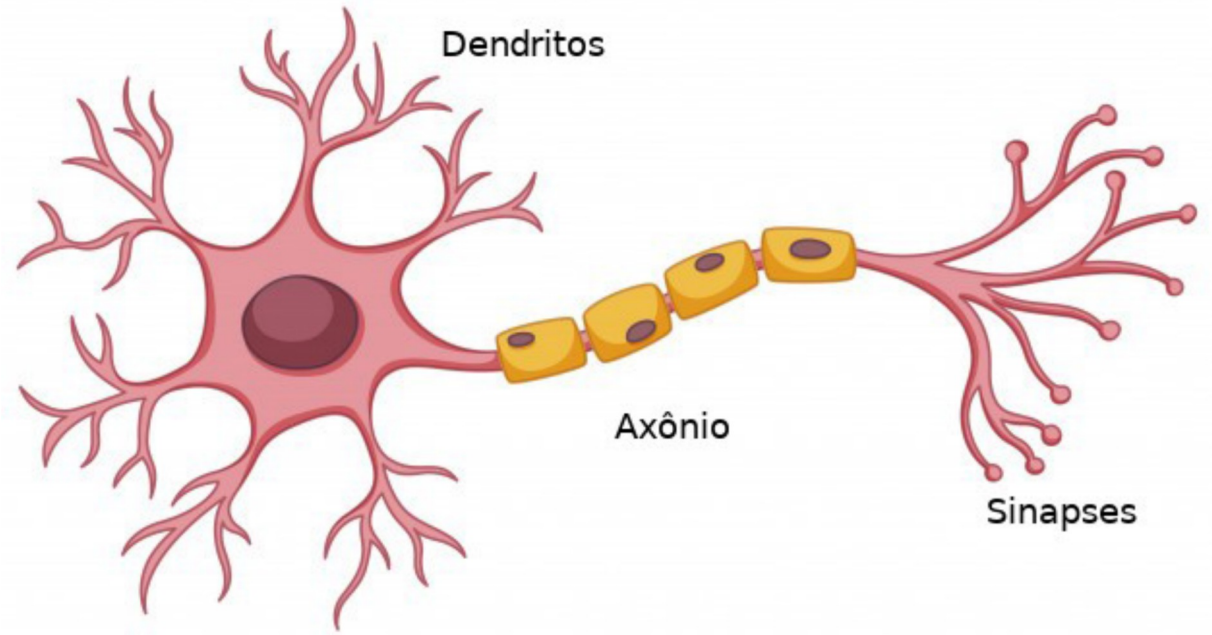
3 REDES NEURAIS ARTIFICIAIS

As Redes Neurais Artificiais (RNAs) são sistemas de neurônios que consistem de elementos de cálculo simples. Estes neurônios estão interligados entre si através de conexões ponderadas, denominadas pesos sinápticos.

As Redes Neurais Artificiais são modelos matemáticos inspirados no cérebro humano onde a unidade básica de processamento são os neurônios. O primeiro modelo matemático para um neurônio artificial foi proposto por McCulloch e Pitts (1943), no qual foram definidos teoremas matemáticos que foram inspiração para a criação do modelo Perceptron (ROSENBLATT, 1958).

O neurônio artificial inspira-se na anatomia do neurônio cortical típico humano apresentado na Figura 1. O neurônio cortical é composto basicamente por 3 estruturas (HAYKIN, 2001): os dendritos, responsáveis por coletar sinais químicos; o axônio, responsável por gerar possibilidades de respostas com base nos sinais dos dendritos; e as sinapses, responsáveis por transmitir impulsos nervosos para os demais neurônios.

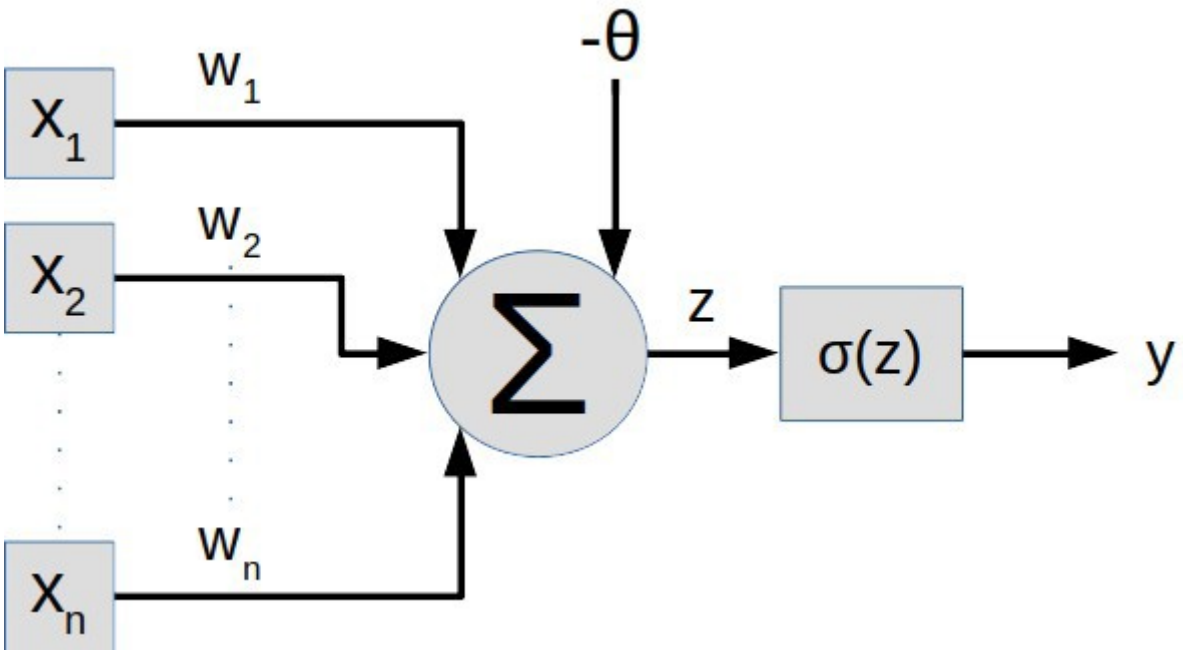
Figura 1 - Estrutura básica de um neurônio cortical.



Fonte: Adaptado de Freepik.com (2019).

O neurônio artificial Figura 2, analogamente apresenta respectivamente as entradas para coleta de sinais, a função de ativação gerando as possibilidades de respostas, e a saída pode se conectar a outros neurônios representando as sinapses (SILVA et al., 2010).

Figura 2 - Esquema básico de um neurônio artificial.



Fonte: Autoria Própria (2019).

Para Haykin (2001), a característica mais importante de uma rede neural é a sua capacidade de aprender e melhorar seu desempenho. O processo de treinamento é iterativo que busca minimizar o erro, otimizando os pesos sinápticos de acordo com os padrões de entrada apresentados, repetindo o processo para cada elemento do conjunto de treinamento. Sendo assim, a rede neural aprende extraindo regras básicas do conjunto de treinamento, encontrando uma solução geral para um problema.

3.1 MODELO PERCEPTRON

O Perceptron é um modelo probabilístico para armazenamento de informação desenvolvido por Frank Rosenblatt (1958), que permite a organização no cérebro em um componente básico, o neurônio artificial perceptron (Figura 2).

A simplicidade de um único neurônio perceptron permite uma compreensão

clara de como funciona uma rede neural em termos matemáticos. O neurônio é composto por diversas entradas e uma única saída, que produz um resultado binário, atribuindo um grau de importância a cada uma de suas entradas expressos por números reais.

Essa importância das entradas é denominada peso sináptico. Para que a saída seja um valor binário é então aplicado uma função de ativação. Esta função de ativação é a expressão matemática que rege o comportamento da saída do neurônio artificial, sendo a função degrau a mais simples aplicada.

Assim, a ativação de um neurônio é obtida por uma função de ativação (exemplificada pela função Degrau - Equação 2) aplicada sobre um produto interno entre as entradas e os pesos sinápticos (Equação 1) (ZUBEN, 2009).

$$z = \sum_{j=0}^{n} w_j x_j \tag{1}$$

$$\sigma(z) = \begin{cases} 0, se\ z - b \leq 0 \\ 1, se\ z - b > 0 \end{cases} \tag{2}$$

Onde x_j representam as entradas deste neurônio e w_j são os pesos sinápticos de cada entrada, z é a combinação linear das entradas com os pesos sinápticos, σ é a saída do neurônio após aplicar a função de ativação e b o viés (do inglês *bias*) representando o limiar.

O processo de aprendizagem deste neurônio se dá pela minimização do erro de sua saída quando comparado com todos os dados conhecidos. De forma que os pesos sinápticos funcionam como uma memória artificial, e o viés indica o quão sensível um neurônio é para ter sua saída acionada.

O teorema fundamental da aprendizagem do modelo perceptron descreve que para qualquer conjunto de dados que é linearmente separável, o algoritmo convergirá para uma solução em um número finito de etapas (Rosenblatt, 1962; Block 1962; Nilsson, 1965).

Em síntese, uma rede neural perceptron funciona como um discriminante linear, permitindo realizar a classificação dos dados em apenas 2 subgrupos, visando separá-los por uma única linha.

Desta foram, há uma limitação deste modelo, de forma que não é possível separar completamente os dados não lineares. Sendo uma abordagem para solucionar este problema incluir mais camadas na rede neural[1].

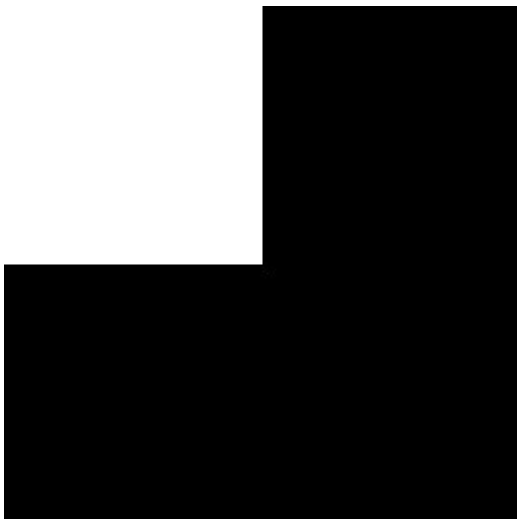
1 Mais detalhes nas seções 3.3 e 3.4.

3.2 EXEMPLO DE FUNCIONAMENTO DO NEURÔNIO ARTIFICIAL PERCEPTRON

Para exemplificar o comportamento de um Perceptron será demonstrado o processamento de um neurônio de 4 entradas para fazer a identificação de uma imagem *raster*[2] limiarizada[3] de 4 *pixels*.

Esta rede foi treinada para reconhecer o padrão apresentado na Figura 3, onde o *pixel* na posição esquerda superior (0, 0) é 1 (branco) e os demais *pixels* são 0 (preto).

Figura 3 - Imagem com o pixel na posição esquerda-superior (0, 0) é 1 (branco) e os demais pixels são 0 (preto).



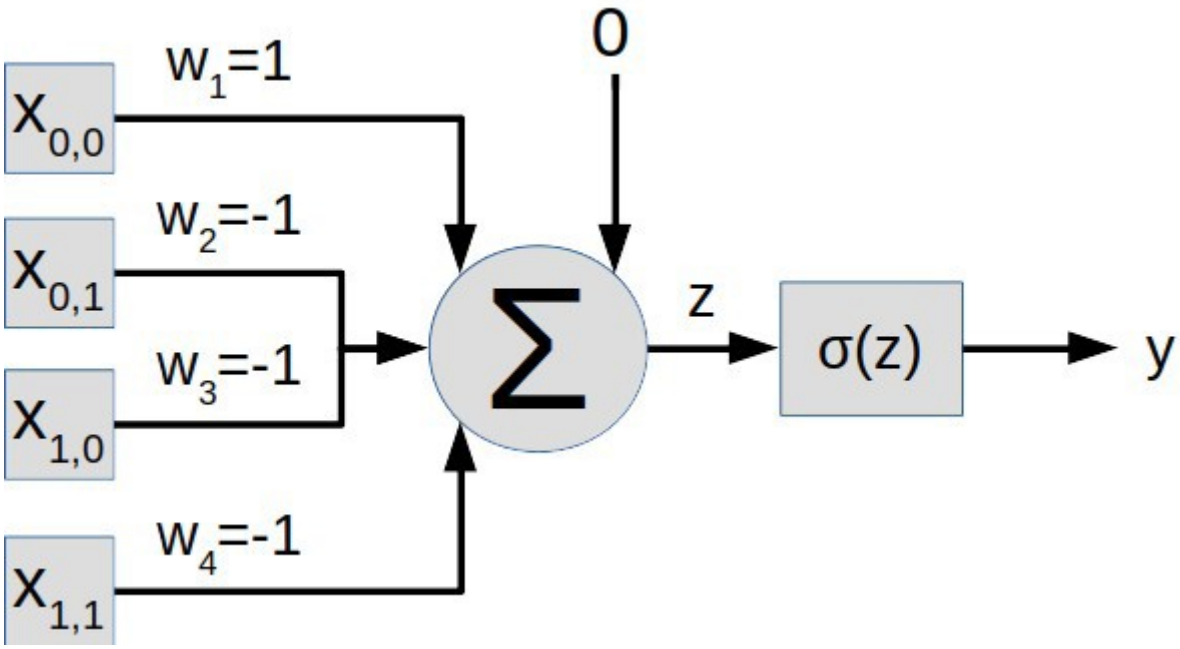
Fonte: Autoria Própria (2019).

O Perceptron treinado para reconhecer esse padrão utilizando uma função de ativação Degrau com viés 0 (zero) é apresentado na Figura 4.

2 Imagem composta por um mapa de *pixels*.

3 Apenas 2 níveis de cor: preto (0) e branco (1)

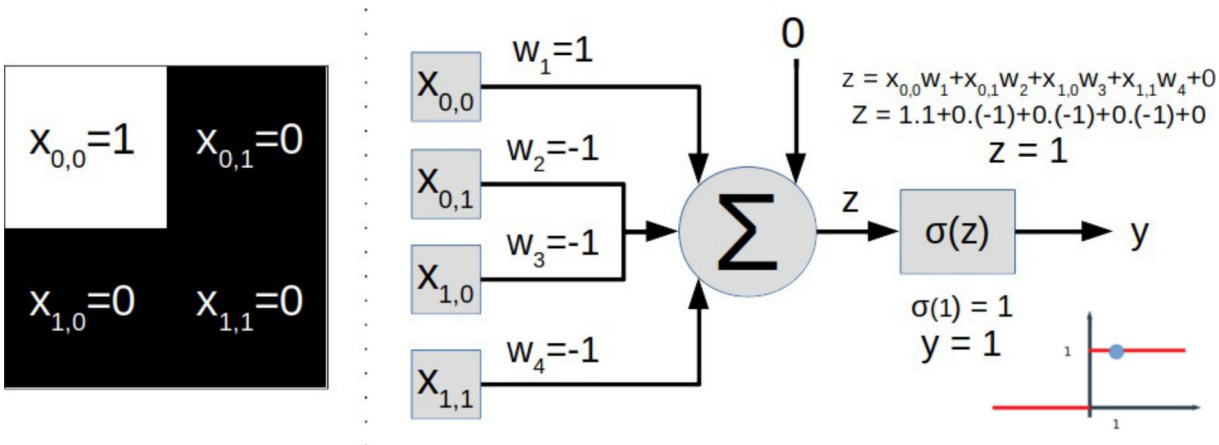
Figura 4 - Representação do Perceptron treinado para reconhecer o padrão.



Fonte: Autoria Própria (2019).

Calculando a saída do Perceptron para a imagem do padrão desejado, tem-se a saída do neurônio como $y=1$, indicando que o padrão foi reconhecido. A Figura 5 apresenta a ficha de cálculo do neurônio perceptron para reconhecimento da imagem com o padrão desejado.

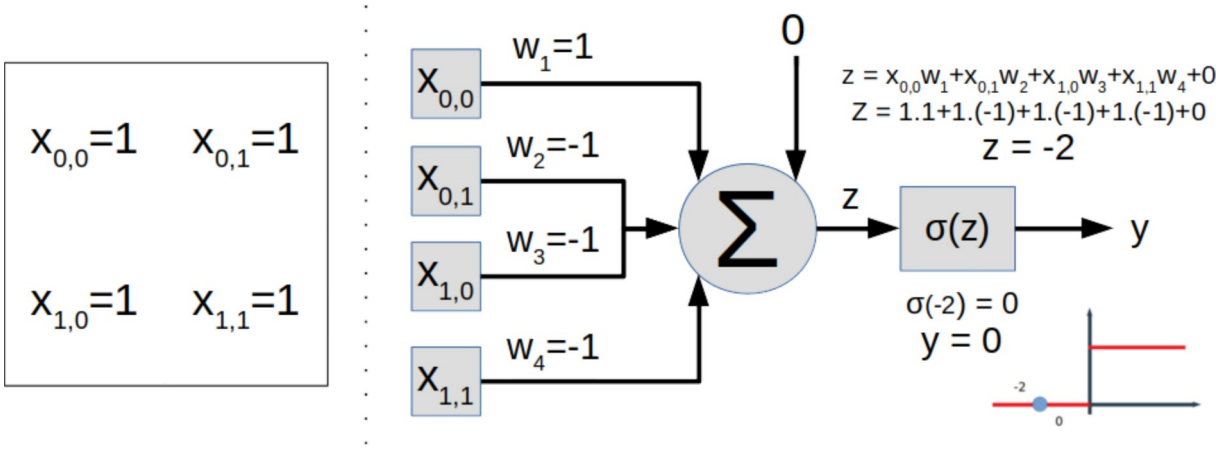
Figura 5 - Cálculo do neurônio perceptron para reconhecimento da imagem com o padrão desejado.



Fonte: Autoria Própria (2019).

Verificando a saída do neurônio para uma imagem totalmente branca (Figura 6), tem-se a saída do neurônio $y=0$, indicando que o padrão apresentado não satisfaz o modelo.

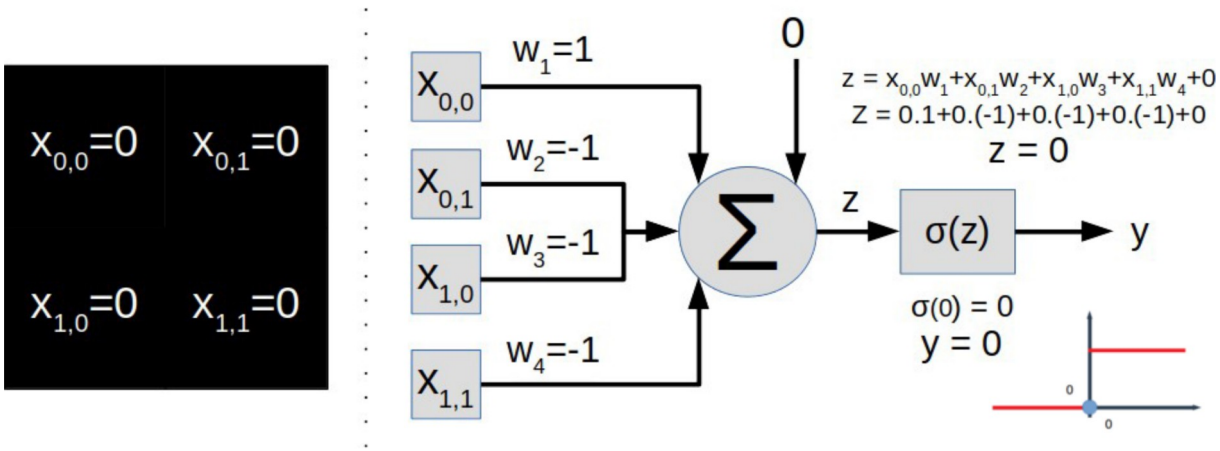
Figura 6 - Cálculo do neurônio perceptron para reconhecimento da imagem totalmente branca.



Fonte: Autoria Própria (2019).

Verificando a saída do neurônio para uma imagem totalmente preta (Figura 7), tem-se a saída do neurônio $y=0$, indicando que o padrão apresentado não satisfaz o modelo.

Figura 7 - Cálculo do neurônio perceptron para reconhecimento da imagem totalmente preta.



Fonte: Autoria Própria (2019).

3.3 REDES NEURAIS PERCEPTRON MULTICAMADA

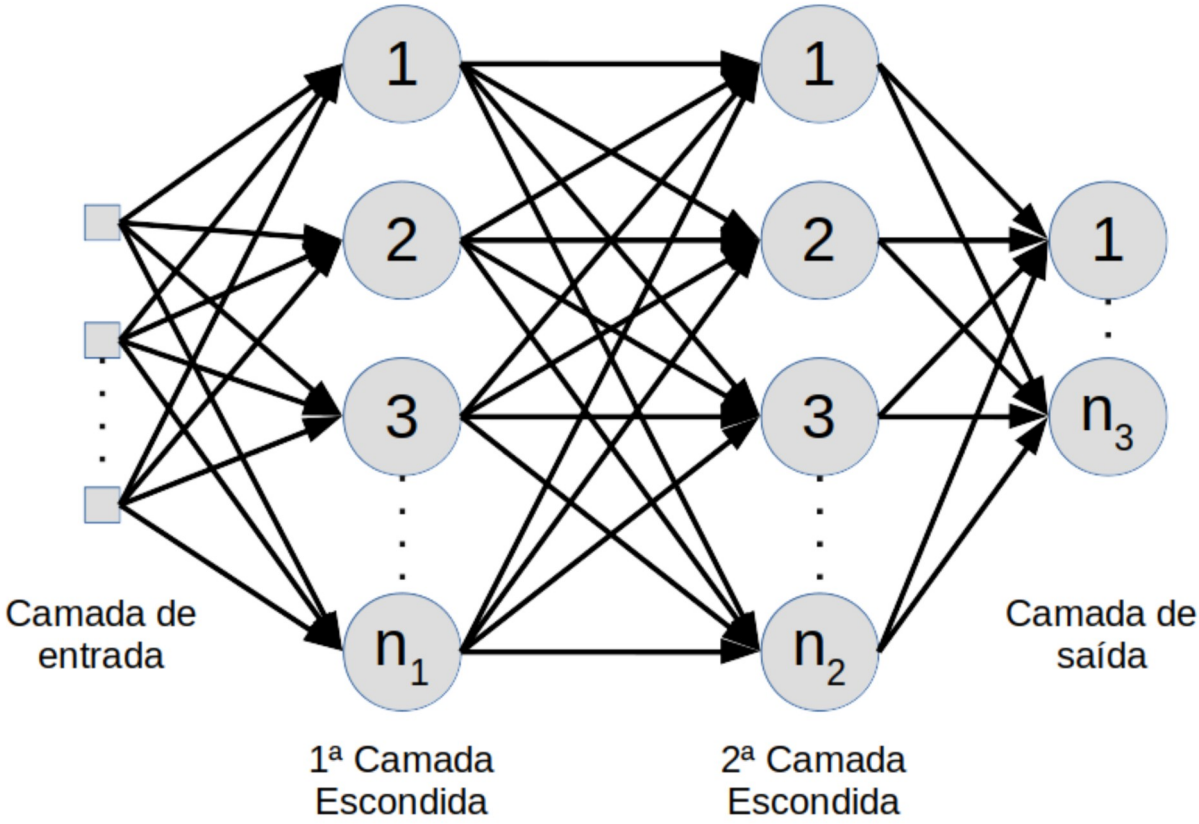
As Perceptrons multicamada (PMC, ou em inglês *Multilayer Perceptron - MLP*), propostas por Rumelhart, Hinton e Williams (1986), são redes neurais aplicadas em processos de classificação com aprendizagem supervisionada. Ela é

constituída por pelo menos 3 camadas. Cada camada composta por um ou mais neurônios perceptron. A primeira camada é composta pelos neurônios de entrada e a última camada pelos, de saída. As camadas entre elas são chamadas de camadas escondidas.

Diferentemente das camadas de entrada e saída, nas camadas escondidas não é possível prever a saída desejada. As camadas escondidas podem aprender a representação interna de dados que não são explícitos nos recursos de entrada (HAYKIN, 2001).

A Figura 8 apresenta uma representação de uma rede neural perceptron multicamada com 2 camadas escondidas.

Figura 8 - Representação de uma rede neural multicamada.



Fonte: Autoria Própria (2019).

Nesta rede neural os neurônios são dispostos em camadas totalmente conectadas (em inglês, *full-connected*), isto é, cada neurônio de uma camada é conectado a todos os neurônios da camada subsequente.

Esta distribuição permite realizar classificações cada vez mais complexas conforme aumenta o número de camadas ocultas.

Matematicamente, uma rede multicamada é representada por funções aninhadas. Por exemplo para a Figura 8 a expressão que representaria a saída 1 da rede seria algo semelhante a Equação 6 (sendo a composição das camadas de entrada, 1ª camada escondida e 2ª camada escondida, respectivamente, Equações 3, 4 e 5).

$$S_{1,j}(x) = \sigma\left(\left[\sum_{i=0}^{n_0} w_{0_{i,j}} x_i + b1\right]\right) \tag{3}$$

$$S_{2,k}(x) = \sigma\left(\left[\sum_{j=0}^{n_1} w_{1_{j,k}} S_{1,j} + b2\right]\right) \tag{4}$$

$$S_{saída,l}(x) = \sigma\left(\left[\sum_{k=0}^{n_2} w_{2_{k,l}} S_{2,k} + b3\right]\right) \tag{5}$$

$$S_{saída,l}(x) = \sigma\left(\left[\sum_{k=0}^{n_2} w_{2_{k,l}} \sigma\left(\left[\sum_{j=0}^{n_1} w_{1_{j,k}} \sigma\left(\left[\sum_{i=0}^{n_0} w_{0_{i,j}} x_i + b1\right]\right) + b2\right]\right) + b3\right]\right) \tag{6}$$

Hinton (2006) introduzem o termo Aprendizagem Profunda (do inglês *Deep-Learning*) e demonstra juntamente com Salakhutdinov (2009) como uma rede neural artificial pode ser treinada camada a camada. Permitindo sua evolução incluindo conceitos novos, mantendo as camadas pré-treinadas.

Uma variação de perceptrons multicamada são as redes convolucionais.

3.4 REDE NEURAL CONVOLUCIONAL

O termo redes neurais convolucionais foi cunhado por LeCun & Bengio (1995), porém, o modelo foi proposto inicialmente por Zhang (1988). Este modelo foi desenvolvido de modo que o processo de aprendizagem demande o mínimo de pré-processamento.

Essas redes também são conhecidas como redes neurais artificiais invariantes a deslocamento (do inglês *shift invariant*) (ZHANG, 1988) ou invariantes a espaço (do inglês *space invariant*) (ZHANG, 1990), pois possuem a capacidade de abstrair características dos dados de entrada de forma a desconsiderar sua posição.

Este modelo teve como ponto de partida as redes neurais artificiais

multicamadas Neocognitron (FUKUSHIMA, 1980), a qual teve seu padrão de conectividade inspirado na organização do córtex visual de gatos estudado por Hubel e Wiesel (1961) em relação a como o animal vê formas geométricas em diferentes orientações.

Hubel e Wiesel (1961) observaram que a captação da luz é realizada pelos neurônios mais próximos da retina. Estes neurônios encaminham os sinais com base em sua localização para neurônios específicos no para córtex visual que por sua vez redistribui para outras camadas.

Analogamente, nessa topologia, considera-se uma forte relação entre valores próximos, sendo criadas pequenas subseções de processamento da camada anterior, restringindo as conexões de entrada dos neurônios de forma a reaproveitar os pesos sinápticos em cada subseção dos dados de entrada, assim, caracterizando um produto de convoluções (GOODFELLOW et.al. 2016). Sendo estas subseções geralmente de dimensões quadradas de 3x3 ou de 5x5, por exemplo. O conjunto de pesos sinápticos desta camada é denominado *kernel* (núcleo).

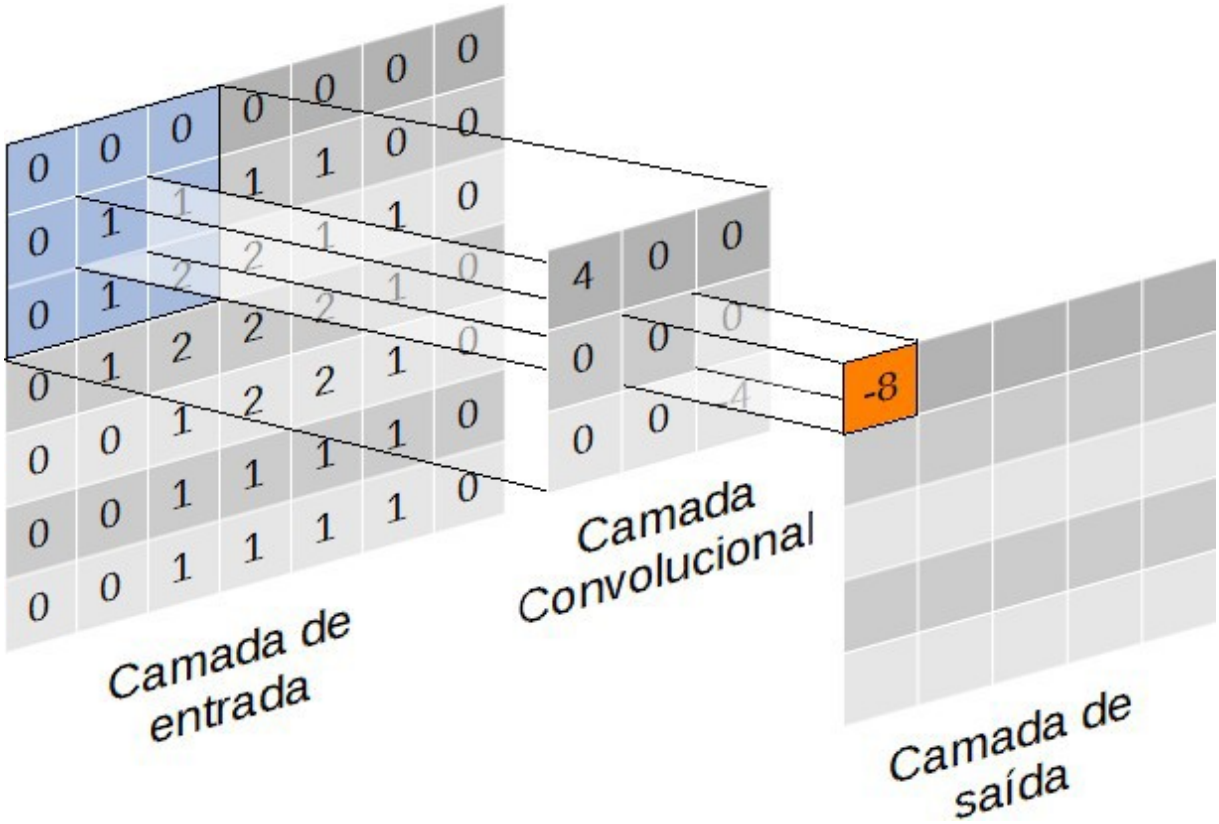
Considerando-se uma entrada de dados bidimensional, o produto de convoluções discretas desta rede neural pode ser expressa como a Equação 7.

$$S_{ij} = \sum_{a=0}^{k-1} \sum_{b=0}^{k-1} w_{ab} x_{(i-a)(j-b)} \tag{7}$$

Onde S_{ij} é a saída de cada subseção, x_{ij} é a matriz de entrada, e w_{ij} é a matriz de pesos e k é a dimensão da subseção.

A Figura 9 apresenta uma representação gráfica da operação matricial dos dados de entrada e a camada convolucional (*kernel*), onde o *kernel* é deslocado produzindo cada um dos valores de saída.

Figura 9 - Representação do processamento de uma subseção em uma rede convolucional.



Fonte: Autoria Própria (2019).

Devido a característica implícita de independência espacial, a arquitetura das redes convolucionais permite processamento paralelo, com a avaliação simultânea de várias regiões da imagem.

De um ponto de vista prático, uma rede neural convolucional funciona como uma camada de pré-processamento que aprende e aplica os filtros para extração de características sem a necessidade de intervenção humana. LISA Lab (2019) apresenta que os filtros locais criados por essa RNA permitem explorar a forte correlação local espacial presente em imagens naturais.

Em uma rede com várias camadas convolucionais ocultas cada camada se torna responsável por transformar o objeto de entrada em uma representação mais abstrata de si mesma (BENGIO, 2011), trazendo características de alto nível necessárias para o processo de classificação.

Um processo comumente utilizado com redes convolucionais é o *pooling*, uma camada utilizada após uma camada convolucional com o objetivo reduzir a dimensão da rede, reduzindo o custo computacional. Esta camada aplica um filtro de região extraindo, geralmente, o valor máximo (*Max-Pooling*) de uma subseção

criando uma invariância a pequenas distorções e evitando sobreajuste[4] (*overfitting*) (WU & GU, 2015).

Uma grande vantagem do uso de uma rede convolucional é a redução significativa no número de parâmetros (pesos sinápticos) do modelo, também apresenta independência a translação, permitindo o processamento invariante ao espaço refletindo na saída uma translação na entrada dos dados, tornando esse modelo bastante útil para processamento de sinais (OTUYAMA, 2000).

3.5 TREINAMENTO

O processo de treinamento de uma rede neural busca minimizar o erro do sistema para aumentar sua acurácia do modelo atualizando os pesos sinápticos da rede (HAYKIN, 2001).

No treinamento de retro-propagação de erro (do inglês *backpropagation*), os pesos são atualizados após o cálculo do erro entre o valor de saída da rede e o valor desejado, isto, para cada elemento do conjunto de treinamento.

Os valores de entrada são apresentados para os neurônios da primeira camada, onde cada neurônio de entrada recebe um valor de entrada, esses valores são propagados pelas camadas intermediárias através das conexões sinápticas, gerando resultados na camada de saída.

O processo é tratado como uma regressão linear ou regressão logística (VALENÇA & LUDERMIR, 1999). O processo inicia-se com a atribuição de pesos sinápticos com valores aleatórios. Então a rede é validada para cada caso de testes, sendo calculado o erro de cada caso. Na sequência, os erros são retro-propagados da saída em direção da entrada para correção dos pesos sinápticos visando minimizar o erro quadrático médio produzido. A retro-propagação do erro pode ser feita calculando-se a derivada da função de ativação, usando regra da cadeia (SILVA, 1998).

Segundo Haykin (2001) como esse processo de derivação demanda muito esforço para ser produzido e como o retro-propagação se aproveita do gradiente da

4 Sobreajuste: quando um modelo se ajusta muito bem ao conjunto de dados reduzindo ao máximo o erro no processo de treinamento. Contudo, se mostra ineficaz e comete erros significativos fazer previsões sobre dados inéditos de entrada.

função de ativação pode-se fazer uma descida de gradiente, em geral fazendo o uso da Regra Delta (Equação 8).

$$w_i' = w_i - \eta \left(S_e - S_c \right) x_i \tag{8}$$

Onde w_i' é o peso novo, w_i o peso antigo, η é a taxa de aprendizagem, S_c é a saída calculada do neurônio e S_e é a saída esperada para o neurônio.

Apesar de simples o treinamento de uma rede neural enfrenta várias dificuldades, uma das principais é o "problema da fuga de gradiente" formalizado por Hochreiter (1991) e aprofundado durante os anos 2000 (Hochreiter et.al., 2001).

Hochreiter (1991) estudou com detalhes o processo de treinamento das redes neurais, observando as atualizações proporcionais à derivada parcial da função de erro em relação ao peso atual. Foi constatado que quando o gradiente é muito pequeno, a atualização dos pesos é ínfima, efetivamente impedindo a evolução do processo de treinamento.

Assim, a função de ativação é crucial para o adequado treinamento.

Outra situação que pode ocorrer é quando funções de ativação apresentam derivadas que podem assumir valores muito maiores, causando o “problema de gradiente explosivo” (BENGIO et al., 1994).

3.6 FUNÇÕES DE ATIVAÇÃO

O processamento de cada neurônio artificial se dá pela combinação linear das entradas aplicada a uma função de ativação (HAYKIN, 2001). Diferentes funções de ativação interferem no processo de treinamento e na qualidade dos resultados apresentados por uma rede neural.

As principais funções de ativação são: Linear, Degrau, Sigmoid, Tangente Hiperbólica, ELU, ReLU, etc.

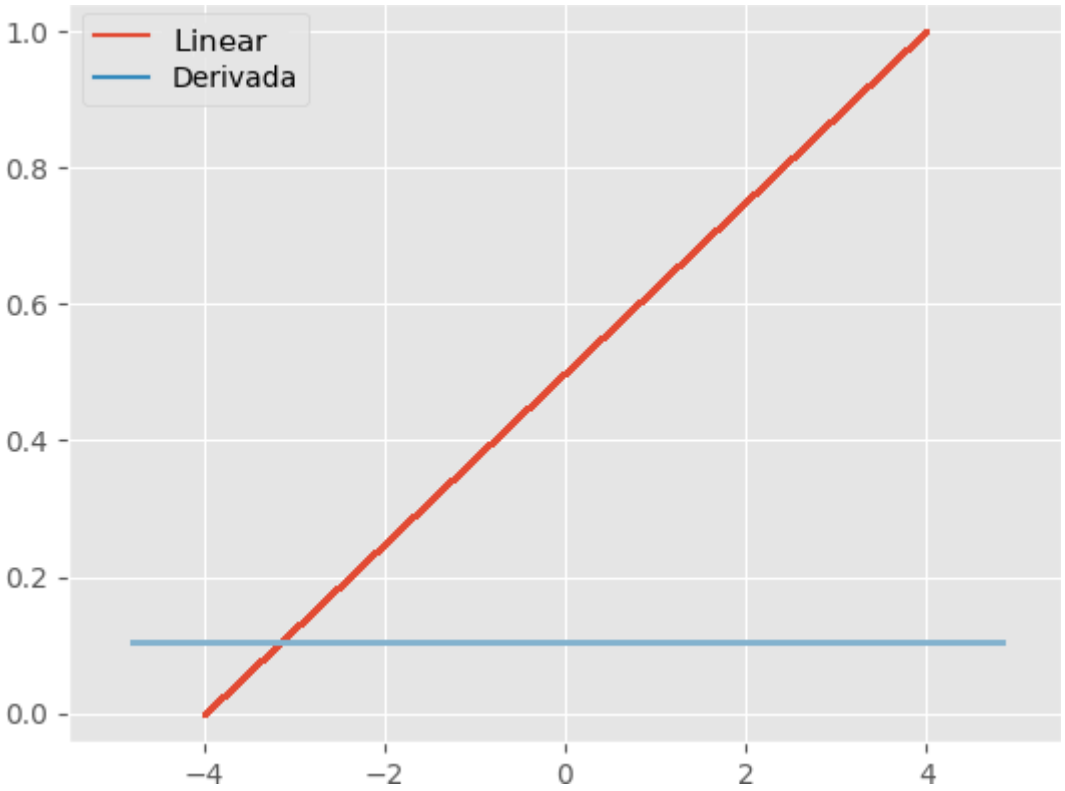
A função de ativação linear não altera a saída de um neurônio, sendo geralmente empregada em processos de regressão numérica, em especial na camada de saída (CARVALHO, 2018).

A equação da função de ativação linear, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 9, Equação 10 e Figura 10.

$$\sigma(x) = a.x + b \tag{9}$$

$$\sigma'(x) = a \tag{10}$$

Figura 10 - Função de ativação linear.



Fonte: Adaptado de FACURE (2017).

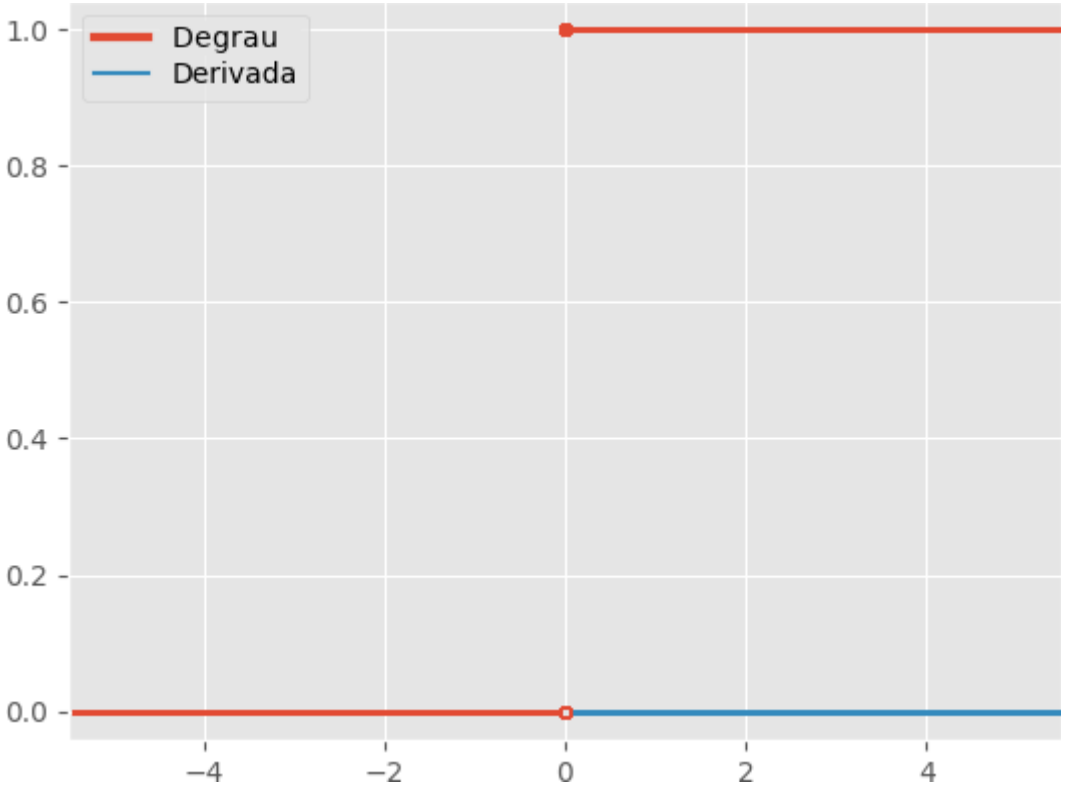
A função de ativação degrau (do inglês *step function,* também conhecida como *threshold* - limiar) define a saída como 0 ou 1 conforme um valor limite estabelecido.

A equação da função de ativação degrau, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 11, Equação 12 e Figura 11.

$$\sigma(x) = \{1, se\ x \geq 0\} \tag{11}$$

$$\sigma'(x) = 0 \tag{12}$$

Figura 11 - Função de ativação degrau.



Fonte: Adaptado de FACURE (2017).

Estas duas funções são as mais simples, porém não apresentam boas características para participarem de um processo de treinamento por possuírem uma derivada constante (função linear) ou derivada nula (*threshold*). Com a derivada constante o gradiente se torna constante sendo um problema no processo de treinamento pois não nos permite identificar se realmente há minimização de erro (BARBOSA, 2013).

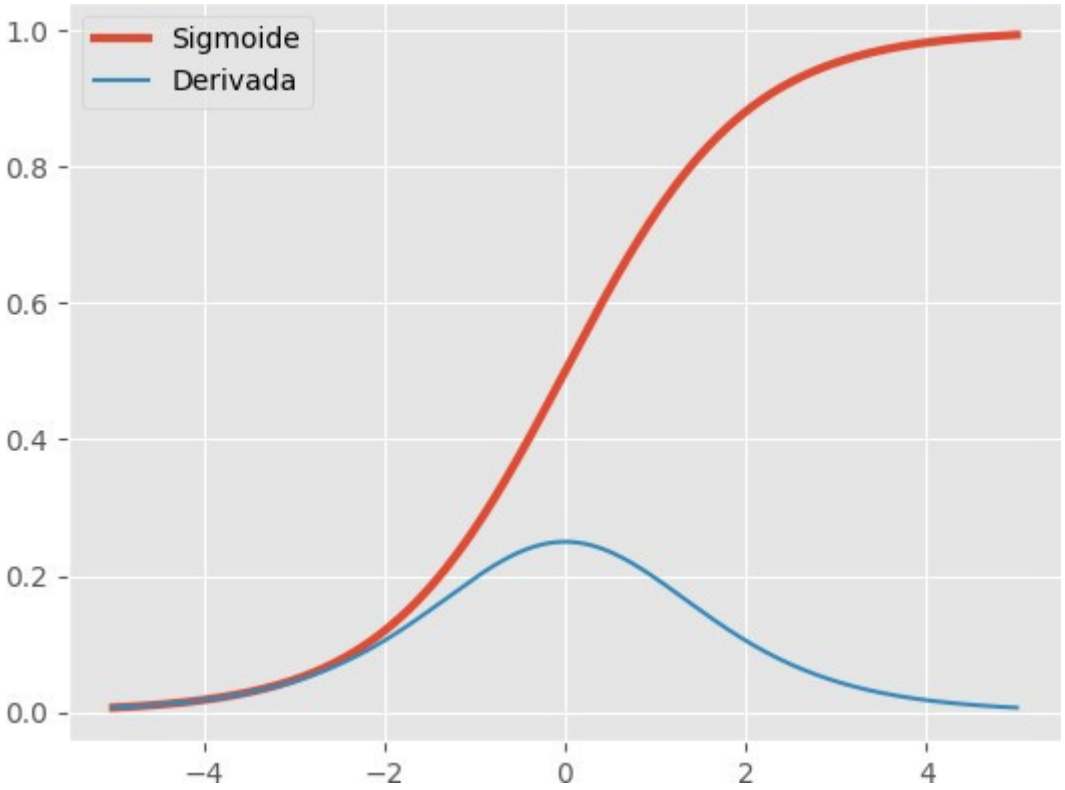
A função de ativação Sigmoid apresenta um comportamento semelhante ao da função degrau, com a vantagem de apresentar uma curva suave e continuamente diferenciável, o que facilita o processo de treinamento dando direcionamento para a adequação dos pesos sinápticos com base no erro do sistema. Como apresenta valor de saída entre 0 e 1 é bastante empregada em camadas que precisam ter como saída apenas números positivos (REIS, 2016).

A equação da função de ativação sigmoid, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 13, Equação 14 e Figura 12.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{13}$$

$$\sigma'(x) = \frac{e^{x}}{(e^{x}+1)^{2}} \tag{14}$$

Figura 12 - Função de ativação sigmoid.



Fonte: FACURE (2017).

A derivada da função sigmoid tende empurrar os valores de saída para os extremos, porém os gradientes se tornam muito pequenos nestes extremos, dificultando identificar se a rede está realmente aprendendo (Data Science Academy, 2019).

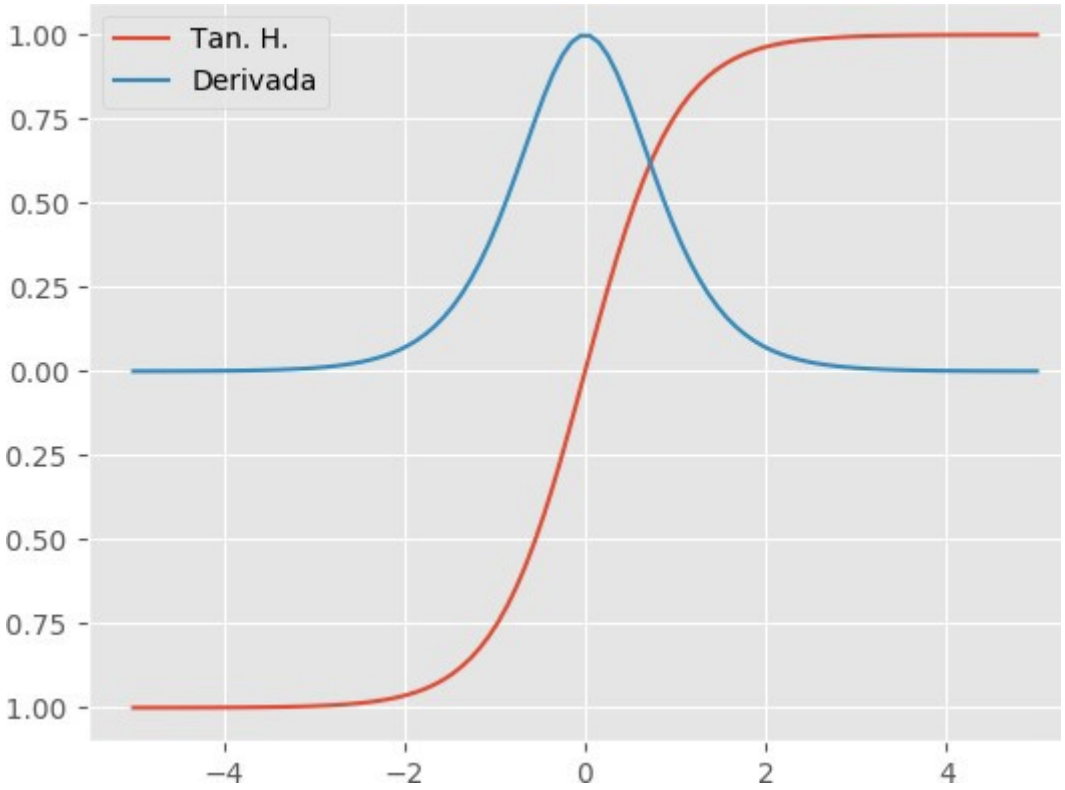
Uma função semelhante a Sigmoid é a tangente hiperbólica. A tangente hiperbólica é uma função escalonada da Sigmoid (Data Science Academy, 2019), mantendo as mesmas características, sendo muito aplicada, tendo o diferencial de possuir como saída valores entre -1 e 1.

A equação da função de ativação tangente hiperbólica, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 15, Equação 16 e Figura 13.

$$\sigma(x) = \tanh(x) \tag{15}$$

$$\sigma'(x) = sech^2(x) \tag{16}$$

Figura 13 - Função de ativação tangente hiperbólica.



Fonte: FACURE (2017).

Unidade linear retificada (do inglês *Rectified Linear Unit* – ReLU) foi proposta por Nair e Hinton (2010) é a função de ativação mais utilizada nos projetos de redes neurais modernos (Data Science Academy, 2019). Ela tem como objetivo reduzir o tempo de treinamento de redes neurais artificiais profundas sendo aplicada aos neurônios das camadas ocultas.

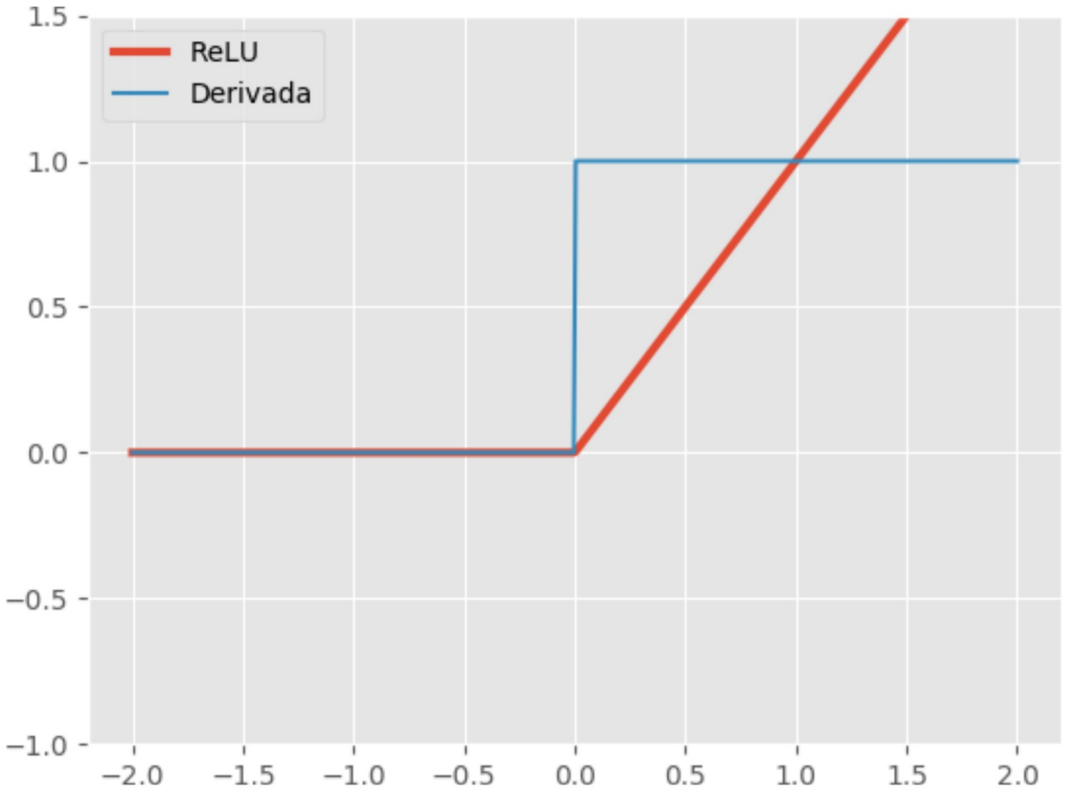
Se a entrada do neurônio for negativa, ele não é ativado o que torna o processamento mais eficiente. O gradiente 0 para x < 0 apesar de reduzir tempo de processamento pode ser um problema para o treinamento, sem dar um direcionamento para o processo de treinamento.

A equação da função de ativação ReLu, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 17, Equação 18 e Figura 14.

$$\sigma(x) = \tanh(x) \tag{17}$$

$$\sigma'(x) = \begin{cases} 0, se\ x < 0 \\ 1, se\ x \geq 0 \end{cases} \tag{18}$$

Figura 14 - Função de ativação ReLU.



Fonte: FACURE (2017).

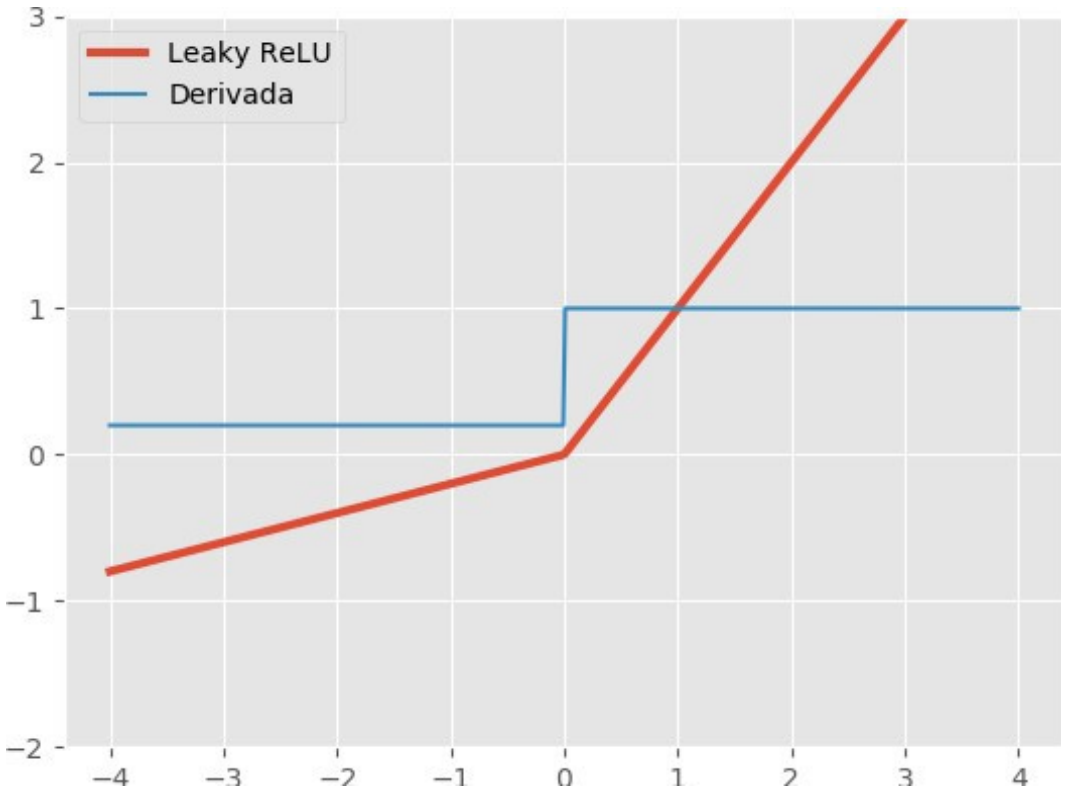
A *Leaky ReLU*, proposta por Maas et. al. (2013), é uma adaptação da ReLU resolvendo o gradiente zero. Seu nome pode ser traduzido como Unidade linear com vazamento, ou mal retificada. Nela para entrada menor que zero ocorre uma redução do gradiente, usando uma taxa baixa, por exemplo a = 0,01.

A equação da função de ativação *Leaky ReLu*, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 19, Equação 20 e Figura 15.

$$\sigma(x)=\begin{Bmatrix} ax, se\ x<0 \\ x, se\ x\geq 0 \end{Bmatrix} \quad (19)$$

$$\sigma'(x)=\begin{Bmatrix} a, se\ x<0 \\ 1, se\ x\geq 0 \end{Bmatrix} \quad (20)$$

Figura 15 - Função de ativação Leaky ReLU.



Fonte: FACURE (2017).

Unidade linear exponencial (*Exponential Linear Unit* - ELU), proposta por Clevert et.al. (2015) que acelera o aprendizado em redes neurais profundas e leva a maior de classificação. Sua derivada no lado negativo permite empurrar as ativações médias para perto de zero, com baixa complexidade computacional.

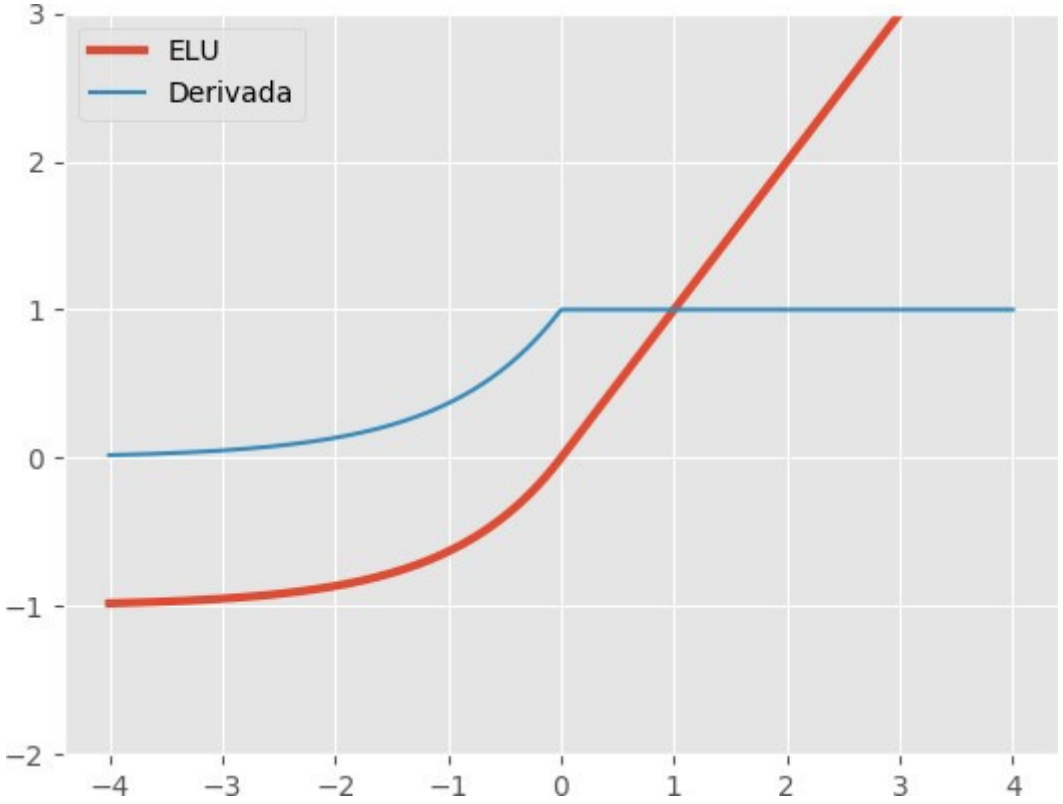
Clevert et.al. (2015) apresentam que as ELUs aceleram consideravelmente o aprendizado em comparação com uma rede ReLU com a mesma arquitetura, obtendo menos de 10% de erro de classificação.

A equação da função de ativação ELU, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 21, Equação 22 e Figura 16.

$$\sigma(x)=\begin{cases} a(e^{x}-1), se\ x<0 \\ x, se\ x\geq 0 \end{cases} \quad (21)$$

$$\sigma'(x)=\begin{cases} a e^{x}, se\ x<0 \\ 1, se\ x\geq 0 \end{cases} \quad (22)$$

Figura 16 - Função de ativação ELU.



Fonte: FACURE (2017).

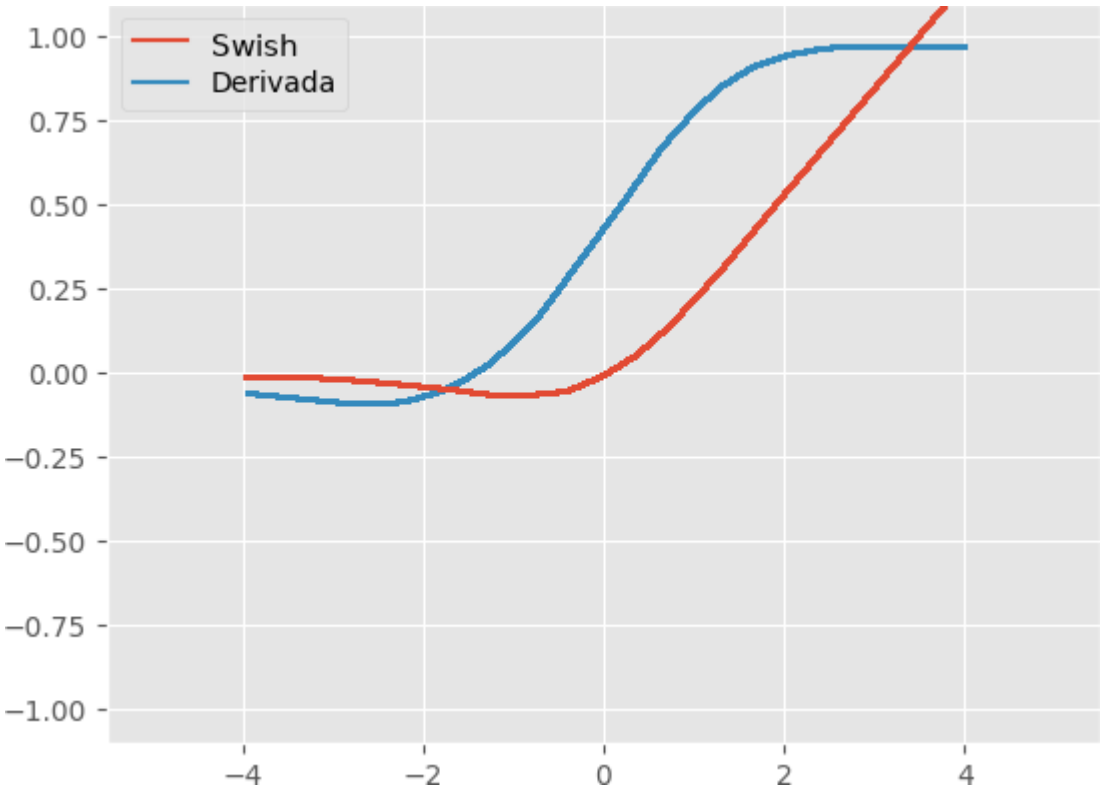
O Swish identificada por pesquisadores do Google (RAMACHANDRAN et.al., 2017) é uma nova função de ativação que apresenta desempenho melhor que o ReLU, com um nível semelhante de eficiência computacional.

A equação da função de ativação Swish, sua derivada e o gráfico de sua curva de ativação são apresentados, respectivamente, em Equação 23, Equação 24 e Figura 17.

$$\sigma(x)=\frac{x}{1+e^{-x}} \tag{23}$$

$$\sigma'(x)=\frac{e^{x}\left(x+e^{x}+1\right)}{\left(e^{x}+1\right)^{2}} \tag{24}$$

Figura 17 - Função de ativação Swish.



Fonte: Adaptado de FACURE (2017).

3.7 RESOLVENDO O OPERADOR LÓGICO E COM UMA REDE NEURAL SIMPLES

Para exemplificar o funcionamento de uma rede neural será apresentado a resolução de um problema simples, o operador E lógico com 2 variáveis de entrada.

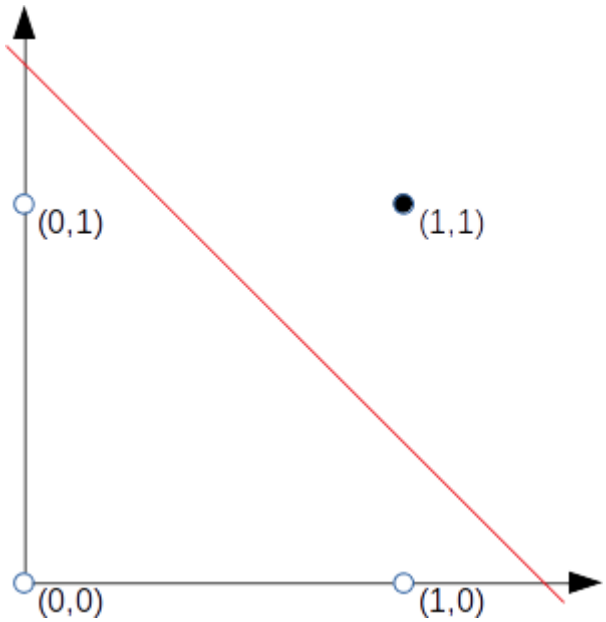
Quando se trabalha com operações lógicas de primeira ordem utiliza-se valores booleanos, as quais podem assumir apenas dois valores: verdadeiro (1) ou falso (0).

O operador E espera que ambas as entradas sejam acionadas recebendo valores verdadeiro (1) para que o resultado da operação também seja verdadeiro (1). Nas demais combinações a resposta é falso (0). Desta forma há 4 combinações

possíveis para as entradas desse operador, as quais serão utilizados no processo de treinamento.

Mapeando essas combinações no plano cartesiano observar-se que as soluções são linearmente separáveis. Isto é, uma única linha consegue separar as 2 classes de interesse, como ilustrado na Figura 18. Sendo a classe de saída verdadeira (1) com elemento destacado em preto e a classe com saída falso (0) em branco. A linha vermelha representa uma possível reta separando as duas classes.

Figura 18 - Combinações da operação E.



Fonte: Autoria própria (2019).

Para este problema simples, pode-se simplificar ao máximo e considerar nossa rede neural composta por um único neurônio perceptron com 2 entradas e uma saída. Como a saída é booleana foi aplicado um operador de limiar como função de ativação. Para efeitos didáticos será utilizado o valor 0,5 como limiar.

Sendo a equação simplificada que descreve a saída do neurônio dada pela Equação 25.

$$S_c = x_1 w_1 + x_2 w_2 - b > 0 \quad (25)$$

Supondo que no primeiro momento foram atribuídos valores aleatórios para os pesos sinápticos de nossa rede neural, sendo $w_1 = 0,8$ e $w_2 = 0,6$.

Então para realiza-se a propagação, avaliando as amostras de treinamento (Quadro 1).

Quadro 1 - Cálculo das saídas da primeira etapa da retro-propagação do exemplo da porta lógica E.

x_1	x_2	Cálculo	S_c	S_e
0	0	S_c = 0*0,8 + 0*0,6 - 0,5 > 0	0	0
0	**1**	S_c = 0*0,8 + 1*0,6 - 0,5 > 0	**1**	**0**
1	0	S_c = 1*0,8 + 0*0,6 - 0,5 > 0	**1**	**0**
1	1	S_c = 1*0,8 + 1*0,6 - 0,5 > 0	1	1

Na sequência, no Quadro 2, calcula-se o erro quadrático médio entre as saídas calculadas (S_c) e esperadas (S_e).

Quadro 2 - Cálculo do erro na primeira etapa da retro-propagação do exemplo da porta lógica E.

$$e=\sum \frac{1}{2}\left(S_e-S_c\right)^2 \rightarrow e=\frac{1}{2}\left((0-0)^2+(1-0)^2+(1-0)^2+(1-1)^2\right) \rightarrow$$

$$e=\frac{1}{2}\left((0-0)^2+(1-0)^2+(1-0)^2+(1-1)^2\right) \rightarrow e=1$$

Aplicando a regra delta, usando taxa de aprendizagem $\eta=0,1$ tem-se a atualização dos pesos sinápticos (Quadro 3). Sendo selecionados casos de treinamento com erro onde a entrada x_i é não zero para atualização dos respectivos pesos, destacados no Quadro 1.

Quadro 3 - Atualização dos pesos sinápticos na primeira etapa da retro-propagação do exemplo da porta lógica E.

$$w_i'=w_i-\eta\left(S_e-S_c\right)x_i$$

$$w_1'=w_1-0,1(1-0)1 \rightarrow w_1'=0,8-0,1 \rightarrow w_1'=0,7$$

$$w_2'=w_2-0,1(1-0)1 \rightarrow w_2'=0,6-0,1 \rightarrow w_2'=0,5$$

Agora com os pesos atualizados tem-se as novas saídas, conforme Quadro 4.

Quadro 4 - Cálculo das saídas da segunda etapa da retro-propagação do exemplo da porta lógica E.

x_1	x_2	Cálculo	S_c	S_e
0	0	S_c = 0*0,7 + 0*0,5 - 0,5 > 0	0	0
0	1	S_c = 0*0,7 + 1*0,5 - 0,5 > 0	0	0
1	**0**	**S_c = 1*0,7 + 0*0,5 - 0,5 > 0**	**1**	**0**
1	1	S_c = 1*0,7 + 1*0,5 - 0,5 > 0	1	1

Calculando erro quadrático médio, é obtido $e=0,5$.

Ocorre então uma nova atualização dos pesos sinápticos (Quadro 5), considerando as linhas com divergência de resultado.

Quadro 5 - Atualização dos pesos sinápticos na segunda etapa da retro-propagação do exemplo da porta lógica E.

$w_1' = w_1 - 0{,}1(1-0)1 \rightarrow w_1' = 0{,}7 - 0{,}1 \rightarrow w_1' = 0{,}6$
$w_2' = w_2 - 0{,}1(1-0)0 \rightarrow w_2' = 0{,}5$

Com os pesos sinápticos atualizados calcula-se as saídas novamente (Quadro 6).

Quadro 6 - Cálculo das saídas da terceira etapa da retro-propagação do exemplo da porta lógica E.

x_1	x_2	Cálculo	S_c	S_e
0	0	S_c = 0*0,6 + 0*0,5 - 0,5 > 0	0	0
0	1	S_c = 0*0,6 + 1*0,5 - 0,5 > 0	0	0
1	**0**	**S_c = 1*0,6 + 0*0,5 - 0,5 > 0**	**1**	**0**
1	1	S_c = 1*0,6 + 1*0,5 - 0,5 > 0	1	1

Calculando erro quadrático médio, $e = 0{,}5$.

Repetindo-se a atualização dos pesos são encontrados os valores apresentados no Quadro 7.

Quadro 7 - Atualização dos pesos sinápticos na terceira etapa da retro-propagação do exemplo da porta lógica E.

$w_1' = w_1 - 0{,}1(1-0)1 \rightarrow w_1' = 0{,}6 - 0{,}1 \rightarrow w_1' = 0{,}5$
$w_2' = w_2 - 0{,}1(1-0)0 \rightarrow w_2' = 0{,}5$

Chegando a uma solução para o problema apresentada no Quadro 8. Com erro quadrático médio $e = 0$ o treinamento da rede neural foi concluído.

Quadro 8 - Cálculo das saídas da última etapa da retro-propagação do exemplo da porta lógica E.

x_1	x_2	Cálculo	S_c	S_e
0	0	S_c = 0*0,5 + 0*0,5 - 0,5 > 0	0	0
0	1	S_c = 0*0,5 + 1*0,5 - 0,5 > 0	0	0
1	0	S_c = 1*0,5 + 0*0,5 - 0,5 > 0	0	0
1	1	S_c = 1*0,5 + 1*0,5 - 0,5 > 0	1	1

4 YOLO

YOLO (do inglês *You Only Look Once*) é um algoritmo de redes neurais convolucionais para detecção de objetos em imagens em tempo real que possui grande capacidade de generalização (REDMON et al., 2016). YOLO roda sobre o software de código livre darknet[5], escrito em C e CUDA.

Redmon e Farhadi (2018b) destacam que a versão 3 do YOLO é um dos algoritmos mais rápidos quando comparado as demais técnicas mantendo elevada acurácia.

A arquitetura do YOLOv3 é de uma rede convolucional multicamada, com 53 camadas convolucionais e *Max-Pooling.*

Na primeira camada (entrada) a imagem é subdividida em uma grade de tamanho N *x* N e o algoritmo estima a probabilidade de cada célula ser o centro de uma determinada classe.

Redmon et al. (2016) apresenta que a resposta de cada célula tem 5 componentes, suas coordenadas centrais (x e y), sua dimensão (w e h) e o nível de confiança da previsão. As coordenadas e dimensões são normalizadas para retornarem valores entre 0 e 1. Já a confiança indica a presença ou ausência de um objeto de qualquer classe. Cada célula da grade faz B essas previsões, portanto, há no total N *x* N *x* B * 5 saídas relacionadas às previsões da célula.

A rede também estima as probabilidades de cada célula conter um objeto, independentemente do número de previsões B. Isso torna as probabilidades da classe N *x* N *x* C no total. Adicionando as predições de classe ao vetor de saída, obtém-se um tensor N *x* N *x* (B * 5 + C) como saída (MENEGAZ, 2018).

YOLOv3 se utiliza de alguns recursos modernos das redes neurais como blocos residuais, pular conexões e reamostragem.

Nos Blocos Residuais o sinal de saída de uma camada é transmitido para a camada subsequente e salta também para outra camada a frente. Permitindo, assim, que informação úteis sejam reaproveitadas (SAHOO, 2018).

Pular conexões (do inglês *skip-connections*), proposto por Orhan & Pitkow (2018) possibilitou o treinamento de redes muito profundas. O seu objetivo principal

5 O repositório do software *darknet* pode ser acessado no GitHub pelo endereço: https://github.com/pjreddie/darknet.git

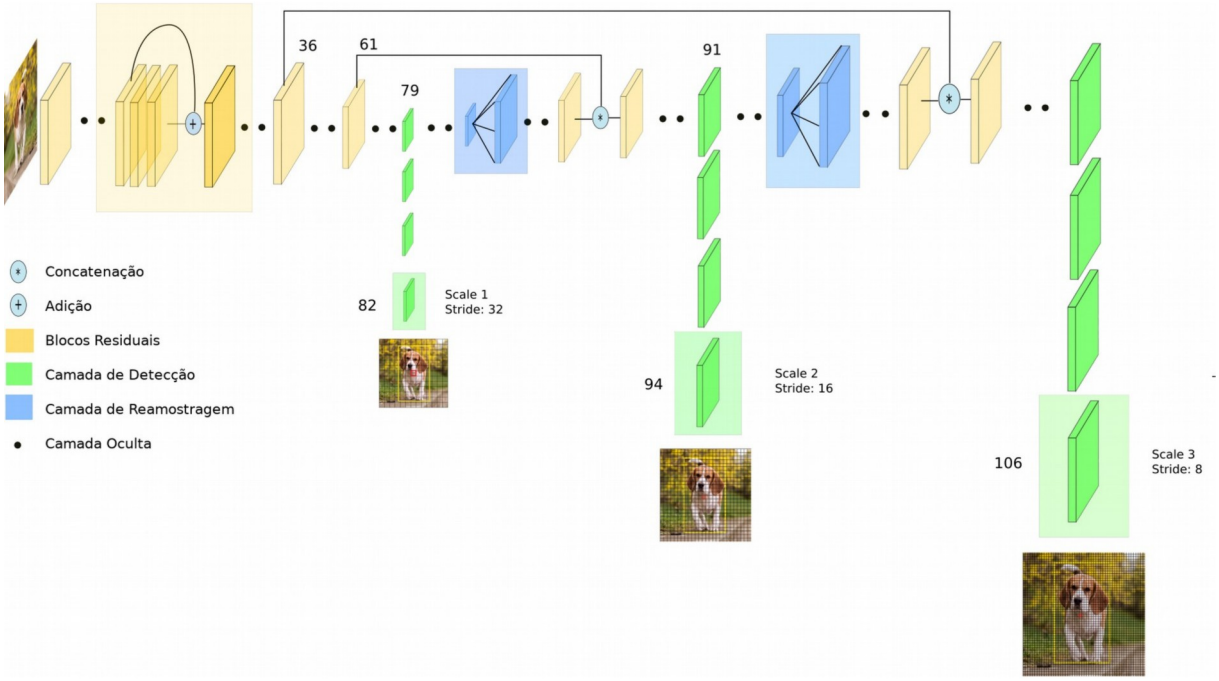
é desativar propositalmente neurônios inertes, reduzindo o custo de treinamento em neurônios que nunca são ativados.

Reamostragem (do inglês *Upsamplings*), sem como objetivo reescalar as informações de uma camada para a outra mantendo a referência posicional da informação (Hua & Gong, 2018).

Fazendo o uso dos blocos residuais e reamostragem a YOLOv3 tem a capacidade dar refinamento aos resultados com 3 diferentes escalas dando detalhes a classificações nas camadas anteriores.

A Figura 19 apresenta a arquitetura completa do YOLOv3.

Figura 19 - Diagrama das camadas da YOLOv3.



Fonte: Adaptado de Kathuria (2018).

5 METODOLOGIA

O estudo iniciou com uma pesquisa bibliográfica realizando levantamento de técnicas de redes neurais e a aplicação das mesmas no processo de visão computacional.

Na sequência realizou-se uma pesquisa exploratória e experimental realizando o treinamento e teste de uma rede neural.

Utilizou-se o algoritmo de aprendizagem profunda YOLOv3 para detectar placas de trânsito em imagens, em especial a sinalização vertical de pare normatizadas pelo CONTRAN (2007),

Primeiramente realizou-se o processo de treinamento com um conjunto de imagens de exemplo usando a rede YOLOv3, adotando o conjunto parâmetros de configuração padrão.

No treinamento a rede pode aprender a reconhecer placas de sinalização vertical através de um conjunto de imagens fornecidas, previamente rotuladas, indicando a posição correta da placa em cada uma das imagens. Esta rotulação foi realizada utilizando o software BBox-Label-Tool[6] (Figura 20).

Os arquivos gerados pelo Bbox-Label-Tool precisam então ser convertidos para o padrão esperado pelo darknet usando o *script* `convert.py`[7]. No arquivo convertido então ajustou-se as classes para cada área selecionada nas imagens.

Os arquivos foram separados em 2 grupos, o grupo de treinamento composto por 10 imagens planas com fundo preto e com fundo branco, e o grupo de validação com 10 imagens reais das placas.

Utilizou-se pesos sinápticos da rede previamente treinada pelos autores carregando o modelo darknet53[8], bem como os arquivos com as regras de treinamento, fazendo o uso do arquivo de configuração `yolov3.cfg`.

6 O repositório do *software* Bbox-Label-Tool pode ser acessado no GitHub pelo endereço: https://github.com/puzzledqs/BBox-Label-Tool.git

7 O script de conversão do padrão Bbox para o padrão esperado pelo YOLO pode ser baixado em: https://github.com/Guanghan/darknet/blob/master/scripts/convert.py

8 Os pesos sinápticos da darknet53 pode ser baixados através do endereço: https://pjreddie.com/media/files/darknet53.conv.74 . Mais arquivos com pesos sinápticos podem ser enconrtados em https://drive.google.com/drive/folders/1uxgUBemJVw9wZsdpboYbzUN4bcRhsuAI

Figura 20 - Rotulação no Bbox-Label-Tool.

Fonte: Autoria Própria (2019).

Para o treinamento foram criados os arquivos `.data` e `.names`. O arquivo `.data` informa o número de classes a serem treinadas, a localização do arquivo com o nome das classes (`.names`), a localização dos arquivos com as listas de imagens de treinamento e validação, e a pasta na qual serão gravados os arquivos dos pesos gerados no treinamento.

No arquivo `yolov3.cfg` foram informados o número de imagens utilizadas em cada etapa do treinamento, sendo utilizadas 8 imagens (`batch=8`, na linha 3), para reduzir o uso da memória foi utilizado subdivisão das imagens de treinamento em 4 grupos (`subdivisions=4`, na linha 4), também foram informados o número de classes a serem identificados (`classes=1`, nas linhas 610, 696 e 783) e o tamanho dos filtros conforme o número de classes dado pela fórmula `filters=(classes + 5)*3` (assim, foi definido `filters=18`, nas linhas 603,

689 e 776).

Estando o arquivo configurado foi executado o comando para iniciar o treinamento:

```
darknet detector train cfg/signs-obj.data cfg/yolo-pare.cfg
        darknet53.conv.74
```

Na sequência usando a rede recém treinada foi executado o algoritmo sobre o conjunto de imagens de validação.

Ao término do treinamento é gerado um arquivo com os novos pesos da rede neural. Então foi executado o comando para carregar a rede neural e fazer a classificação de algumas imagens.

```
darknet detector test cfg/obj.data cfg/yolo-pare.cfg yolo-obj1000.weights
        data/testimage.jpg
```

Foram selecionadas aleatoriamente na internet 40 imagens com a placa de pare, algumas delas com mais de uma placa totalizando 50 placas de pare. Sendo seis images com duas placas de pare, duas imagens com três placas de pare, quatro imagens com placas rotações significativas, sete imagens de difícil reconhecimento por estarem obstruídas, rasuradas ou com focos de luz. As demais imagens a havia uma única placa de pare em boas condições visuais.

Estas imagens foram classificadas pela rede treinada gerando os resultados apresentados a no Capítulo 7.

6 RESULTADOS E DISCUSSÕES

A partir da rede neural treinada, alimentada com 40 imagens contendo a placa de “pare”, sendo algumas imagens com uma ou mais placa desta categoria, totalizando 50 placas a serem identificadas pela rede.

A rede neural treinada conseguiu reconhecer em 82% das placas de pare nas imagens fornecidas, havendo um caso de falso positivo, reconhecendo o verso de uma placa de pare.

A rede neural treinada foi capaz de reconhecer múltiplos objetos sobre a mesma imagem, conforma apresentado na Figura 21, foram apresentados a rede oito imagens com múltiplas placas de pare, totalizando 18 placas e destas a rede reconheceu 14 placas, ou seja 77% das placas em imagens com múltiplas placas.

Figura 21 - Múltiplas placas de pare onde todas foram reconhecidas.

A Figura 22 apresenta um caso em que na mesma imagem uma placa foi reconhecida e a outra não.

Figura 22 - Múltiplas placas de pare onde nem todas foram reconhecidas.

As placas de pare foram reconhecidas mesmo estando rotacionada ou parcialmente obstruída, evidenciando o alto poder de abstração da rede neural. As Figura 23 e Figura 24 apresentam exemplos dessas imagens reconhecidas.

Figura 23 - Placa de pare rotacionada.

Figura 24 - Placa de pare obstruídas reconhecidas.

Foram apresentados a rede 4 imagens com rotação significativa e sete imagens de difícil reconhecimento por estarem obstruídas, rasuradas ou com focos de luz. Sendo reconhecidas 75% das placas de pare rotacionadas e 43% das placas obstruídas. As Figura 25, Figura 26 e Figura 27 apresentam exemplos destas placas que não foram reconhecidas pela RNA.

Figura 25 - As placas não reconhecidas que estavam distorcidas pela rotação excessiva.

Figura 26 - As placas não reconhecidas que estavam rasuradas.

Figura 27 - As placas não reconhecidas com focos de luz.

Havendo uma única amostra com falso positivo, sendo o verso de uma placa de pare classificado fora do padrão de treinamento (Figura 28).

Figura 28 - Caso de falso positivo, verso da placa reconhecido (à esquerda).

Observando as imagens onde haviam apenas placas em bom estado, 96% delas foram reconhecidas.

7 CONCLUSÕES

Foi alcançada uma boa taxa de sucesso o treinamento de uma rede neural convolucional para o reconhecimento de placas de pare, a partir da adaptação da rede neural YOLOv3. A estrutura da rede neural profunda, de 53 camadas YOLOv3 foi mantida intacta. Nesta rede realizou-se um novo treinamento, com o modelo de placa de trânsito, a partir da reunião de dez placas com fundo preto e branco. A partir de então a rede foi validada com quarenta imagens selecionadas aleatoriamente, compondo um total de cinquenta placas de pare a serem identificadas.

O modelo conseguiu reconhecer a placa de pare em 82% das situações propostas. Sendo que as placas em bom estado, com baixa rotação, sem obstrução ou focos de luz obtiveram maiores percentuais de reconhecimento chegando a 96%.

A rede demonstrou boa capacidade de abstração partindo de imagens planas, sendo capaz de reconhecer os objetos independente de sua escala, translação e rotação. Sendo que a qualidade dos resultados pode ser melhorada aumentando-se o conjunto de treinamento com mais exemplos reais.

Para trabalhos futuros há a possibilidade de trabalhar com o modelo YOLOv3 adaptando-o com diferentes funções de ativação, bem como realizar o treinamento para a classificação de diferentes tipos de objetos.

REFERÊNCIAS

BARBOSA, F. R. **Diagnóstico de falhas incipientes a partir das propriedades físico-químicas do óleo isolante em transformadores de potência com o método alternativo à análise de gases dissolvidos.** 2013. Tese (Doutorado). Programade Pós-Graduação em Engenharia Elétrica, Universidade Federal do Ceará, Fortaleza, 2013.

BENGIO, Y. Deep learning of representations for unsupervised and transfer learning. In: **Proceedings of the Unsupervised and Transfer Learning challenge and workshop**, 2011.

BENGIO, Y.; SIMARD, P.Y.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. In: **IEEE transactions on neural networks**, 5 2, 157-66. 1994.

BLOCK, H. The perceptron: a model for brain functioning. **Reviews of Modern Physics**. 123-135p. v.34. 1962.

CARVALHO, V. P. **Previsão de Séries Temporais no Mercado Financeiro de Ações com o uso de Rede Neural Artificial.** 2018. Dissertação (Mestrado) Programa de Pós-graduação em Engenharia Elétrica e Computação. Universidade Presbiteriana Mackenzie. 2018. Disponível em: <http://tede.mackenzie.br/jspui/bitstream/tede/ 3710/6/VALTER%20PEREIRA%20DE%20CARVALHO.pdf> Acesso em: set. 2019.

Data Science Academy. **Deep Learning Book**, 2019. Disponível em: <http://www.deeplearningbook.com.br/>. Acesso em: abr. 2019.

FACURE, M. **Funções de Ativação**: Entendendo a importância da ativação correta nas redes neurais. 2017. Disponível em: <https://matheusfacure.github.io/ 2017/07/12/activ-func/> Acesso em: ago. 2019.

Freepik.com, Imagem: Diagrama De Celulas Tronco No Fundo Branco, Disponível em <https://br.freepik.com/vetores-gratis/diagrama-de-celulas-tronco-no-fundo-branco_2480958.htm>, Acesso em: Outubro de 2019.

FUKUSHIMA, K. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in **Position. Biological Cybernetics.** 36 (4): 193–202. Springer-Verlag. 1980.

GOODFELLOW, I; BENGIO, Y.; COURVILLE, A**. Deep Learning**. MIT Press. p.326. 2016. Disponível em: <http://www.deeplearningbook.org>. Acesso em: out. 2019.

HAYKIN, S. **Redes Neurais. Princípios e prática**. 2 ed. Porto Alegre: Bookman. 2001.

HINTON, G. Learning multiple layers ofrepresentation. TRENDS in **Cognitive Sciences**. v.11. Elsevier, 2006. Disponível em: <https://www.cs.toronto.edu/~hinton/ absps/tics.pdf>. Acesso em Outubro de 2019.

HOCHREITER, S. **Untersuchungen zu dynamischen neuronalen Netzen**. 1991. Tese (PHD), Institut f. Informatik, Technische Univ. Munich, 1991.

HOCHREITER, S.; BENGIO, Y.; FRASCONI, P.; SCHMIDHUBER, J.. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: **A Field Guide to Dynamical Recurrent Neural Networks.** IEEE Press, 2001.

HUA J., GONG X. A Normalized Convolutional Neural Networkfor Guided Sparse Depth Upsampling. In: **Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence**. 2018. Disponível em: <https://pdfs.semanticscholar.org/fb1d/c10e7920fecdac843d3a3d537774dd6e62c2.pdf>. Acesso em: ago. 2019.

KATHURIA, A. **What's new in YOLO v3?.** 2018. Disponível em: <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>. Acesso em: out. 2019.

LECUN, Y; BENGIO, Y. Convolutional networks for images, speech, and time series. In Arbib, Michael A. (ed.). **The handbook of brain theory and neural networks** (2ed.). The MIT press. 1995.

LISA Lab. **Convolutional Neural Networks (LeNet)** - DeepLearning 0.1 documentation. Disponível em: <http://deeplearning.net/tutorial/lenet.html>. Acesso em jul. 2019

MCCULLOCH, W; PITTS, W. A Logical Calculus of the Ideas Immanent in **Nervous Activity**. 1943. Disponível em: <http://www.cse.chalmers.se/~coquand/ AUTOMATA/ mcp.pdf>. Acesso em: 10/08/2019

MENEGAZ, M. **Understanding YOLO**. 2018. Disponível em: <https://hackernoon.com/understanding-yolo-f5a74bbc7967>. Acesso em: set. 2019.

MITCHELL, T. M. **Machine Learning**, New York: McGraw-Hill. 1997.

NAIR, V.; HINTON, G. E., Rectified linear units improve restricted boltzmann machines. in: **Proceedings of the 2010 International Conference on Machine Learning (ICML).** 2010.

NILSSON, N. J. **Learning Machines**: Foundations of trainable pattern-classifyig systems. New York: Mcgraw-Hill. 1965.

ORHAN, E., PITKOW, X. Skip Connections Eliminate Singularities. In: **International Conference on Learning Representations**. 2018. Disponível em: < https://openreview.net/forum?id=HkwBEMWCZ>. Acesso em: set. 2019.

OTUYAMA, J.M. **Rede Neural por Convolução para Reconstrução Estéreo.** 2000. Dissertação (Mestrado) Programa de Pós-graduação em Computação. Universidade Federal de Santa Catarina. Florianópolis. 2000. Disponível em: <https://repositorio.ufsc.br/bitstream/handle/123456789/78340/171568.pdf?sequence=1&isAllowed=y> Acesso em: set. 2019.

RAMACHANDRAN, P, ZOPH, B, LE, Q. V. Swish: a Self-Gated Activation Function. In: **Neural and Evolutionary Computing**. Cornell University. 2017

REIS. B, **Redes neurais - Funções de ativação**. 2016. Disponível em: <http://www.decom.ufop.br/imobilis/redes-neurais-funcoes-de-ativacao/> Acesso em: set. 2019.

ROSENBLATT, F. **Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms**. Washington: Spartan Books. 1962.

ROSENBLATT, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in **The Brain. Psychological Review**. Cornell Aeronautical Laboratory. v. 65. 1958.

SAHOO, S.. **Residual blocks** - Building blocks of ResNet. 2018. Disponível em: <https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec>. Acesso em: set. 2019.

SALAKHUTDINOV, R,; HINTON, G. Deep Boltzmann Machines. In: **Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)**. 16–18 April 2009, Clearwater Beach, Florida, USA. 2009. Disponível em: <https://pdfs.semanticscholar.org/ddc4/5fad8d15771d9f1f8579331458785b2cdd93.pdfproceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf>. Acesso em Outubro de 2019.

SCHMIDHUBER, J. **Deep learning in neural networks:** An overview. Elsevier Neural Networks. v. 61, 2015.

SILVA, I.N.; SPATTI, D.H.; FLAUZINO, R.A., **Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas**: Um Curso Prático. Porto Alegre, RS: Artliber, 2010.

SILVA, L.N.C. **Análise e Síntese de Estratégias de Aprendizado para Redes Neurais Artificiais.** 1998. Dissertação (Mestrado em Automação) - Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 1998. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/theses/ lnunes_mest/>. Acesso em: set. 2019.

VALENÇA, M.J.S.; LUDERMIR T.B., “Selforganization neurons blocks networks”, ICCIMA, IEEE, pages 60-64, New Delhi, Índia, Sep. 1999.

VERMA, R. **Convolutional Neural Network Basics**. (blog). 2019. Disponível em: <https://rohanverma.net/blog/2018/10/14/convolutional-neural-network-basics/> Acesso em: ago. 2019.

WU, H.; GU, X, Max-Pooling Dropout for Regularization of Convolutional Neural Networks. In: **ArXiv**. 2015. Disponível em: <https://arxiv.org/pdf/1512.01400.pdf> Acesso em: set. 2019.

ZHANG, W. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. In: **Applied Optics.** 1990.

ZHANG, W. Shift-invariant pattern recognition neural network and its optical architecture. In: **Proceedings of Annual Conference of the Japan Society of Applied Physics**. 1988.

ZUBEN, F. J. Von. **Redes neurais artificiais**. 2009. Notas de Aula. Disponível em: <ftp://ftp.dca.fee.unicamp.br/pub/docs/vonzuben/ia013_2s09/notas_de_aula/topico2_2s09.pdf>. Acesso em: ago. de 2019.