

# Do Neurônio ao ChatGPT: Desvendando Redes Neurais e Transformers

---

Leonardo Morales Belluzzi - 0020482413017

ITE015 - Tópicos especiais em Informatica (Escolha IV)

# AVISOS IMPORTANTES



“Dúvidas podem ser enviadas e serão respondidas... se possível, dado o *limitado conjunto de dados* que treinaram este apresentador.”

# OBJETIVOS



COMPREENDER FUNDAMENTOS DE  
INTELIGÊNCIA ARTIFICIAL E REDES NEURAIS E  
TRANSFORMERS

# Inteligência Artificial

Programas com habilidade de agir como humanos.



1950

## Aprendizado de máquina

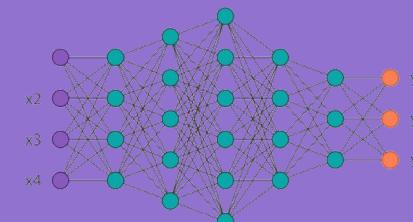
Algoritmos com habilidade de aprender sem programação expressa.



1980

## Aprendizado profundo

Redes neurais artificiais que aprendem através de um grande volume de dados.



2010

## IA Generativa

Criação de novos conteúdos, como texto, imagens, música, áudio e vídeos.



2020



# Leonardo Belluzzi

- Coordenador de Arquitetura
- Engenheiro de Sistemas Especialista
- + 10 Anos de experiência no Mercado

# Sumário

## 1. Breve História da Inteligência Artificial

Panorama da evolução da IA, desde seus conceitos iniciais até os avanços recentes.

## 2. Fundamentos do Aprendizado de Máquina (Machine Learning)

Definição, tipos de aprendizado e aplicações práticas.

## 3. Aprendizado Profundo (Deep Learning)

Funcionamento e diferenciais em relação ao aprendizado de máquina tradicional.

## 4. Métricas de Avaliação

Principais indicadores para medir desempenho e precisão de modelos.

## 5. Arquitetura Transformers

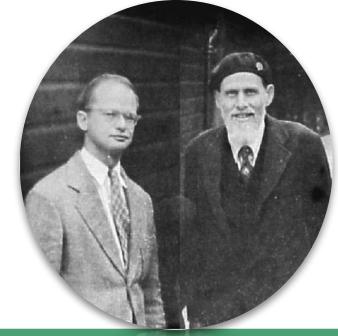
Introdução, funcionamento e impacto dessa arquitetura nos avanços da IA.

# Breve História da Inteligência Artificial



A ideia de “copia, mas não faz igual” remete ao conceito de biomimética ou bioinspiração

# A ORIGEM DO CONCEITO DE INTELIGÊNCIA ARTIFICIAL E A PRIMEIRA REDE NEURAL



Bulletin of Mathematical Biology Vol. 52, No. 1/2, pp. 99–115, 1990.  
Printed in Great Britain.

0092-8240/90\$3.00 + 0.00  
Pergamon Press plc  
Society for Mathematical Biology

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY\*

■ WARREN S. MCCULLOCH AND WALTER PITTS  
University of Illinois, College of Medicine,  
Department of Psychiatry at the Illinois Neuropsychiatric Institute,  
University of Chicago, Chicago, U.S.A.

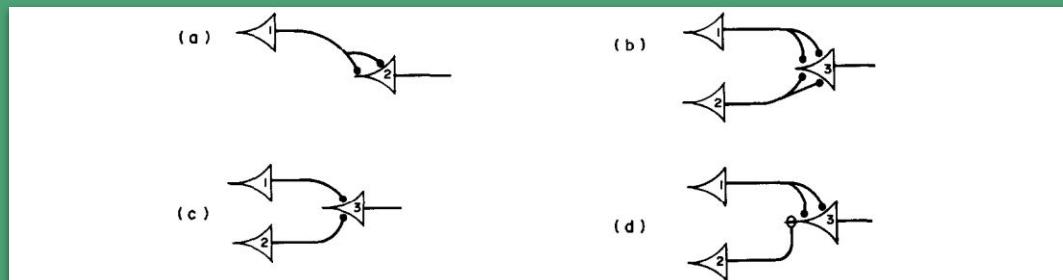
Because of the "all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic. It is found that the behavior of every net can be described in these terms, with the addition of more complicated logical means for nets containing feedback loops. The results apply to all kinds of nervous activity, other than field net behavior in the fashion it describes. It is shown that many particular choices among possible neurophysiological assumptions are equivalent, in the sense that for every net behaving under one assumption, there exists another net which behaves under the other and gives the same results, although perhaps not in the same time. Various applications of the calculus are discussed.

**1. Introduction.** Theoretical neurophysiology rests on certain cardinal assumptions. The nervous system is a net of neurons, each having a soma and an axon. Their adjunctions, or synapses, are always between the axon of one neuron and the soma of another. At any instant a neuron has some threshold, which excitation must exceed to initiate an impulse. This, except for the fact and the time of its occurrence, is determined by the neuron, not by the excitation. From the point of excitation the impulse is propagated to all parts of the neuron. The velocity along the axon varies directly with its diameter, from  $< 1 \text{ ms}^{-1}$  in thin axons, which are usually short, to  $> 150 \text{ ms}^{-1}$  in thick axons, which are usually long. The time for axonal conduction is consequently of little importance in determining the time of arrival of impulses at points unequally remote from the same source. Excitation across synapses occurs predominantly from axonal terminations to somata. It is still a moot point whether this depends upon irreversibility of individual synapses or merely upon prevalent anatomical configurations. To suppose the latter requires no hypothesis *ad hoc* and explains known exceptions, but any assumption as to cause is compatible with the calculus to come. No case is known in which excitation through a single synapse has elicited a nervous impulse in any neuron, whereas any neuron may be excited by impulses arriving at a sufficient number of neighboring synapses within the period of latent addition, which lasts  $< 0.25 \text{ ms}$ . Observed temporal summation of impulses at greater intervals

\* Reprinted from the *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115–133 (1943).

## A logical calculus of the ideas immanent in nervous activity (Warren McCulloch e Walter Pitts)

A origem da inteligência artificial remonta a 1943, quando Warren McCulloch e Walter Pitts desenvolveram o primeiro modelo computacional de redes neurais.



# A ORIGEM DO CONCEITO DE INTELIGÊNCIA ARTIFICIAL E A PRIMEIRA REDE NEURAL



O objetivo deste trabalho é explorar a formulação inicial de um modelo lógico e matemático que busca capturar a essência da atividade nervosa

Propondo neurônios artificiais capazes de processar informações e gerar respostas a partir de estímulos. O estudo enfatiza como as operações lógicas podem simular funções cerebrais

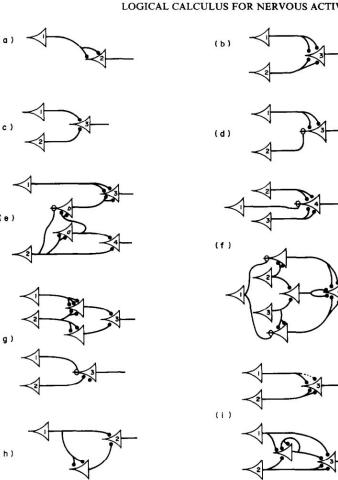
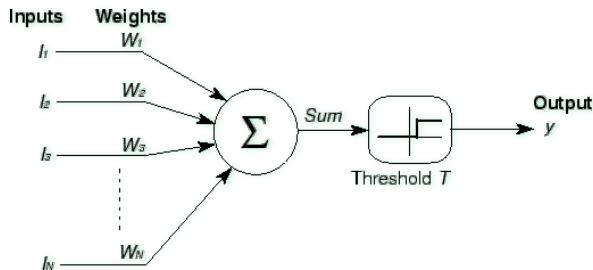
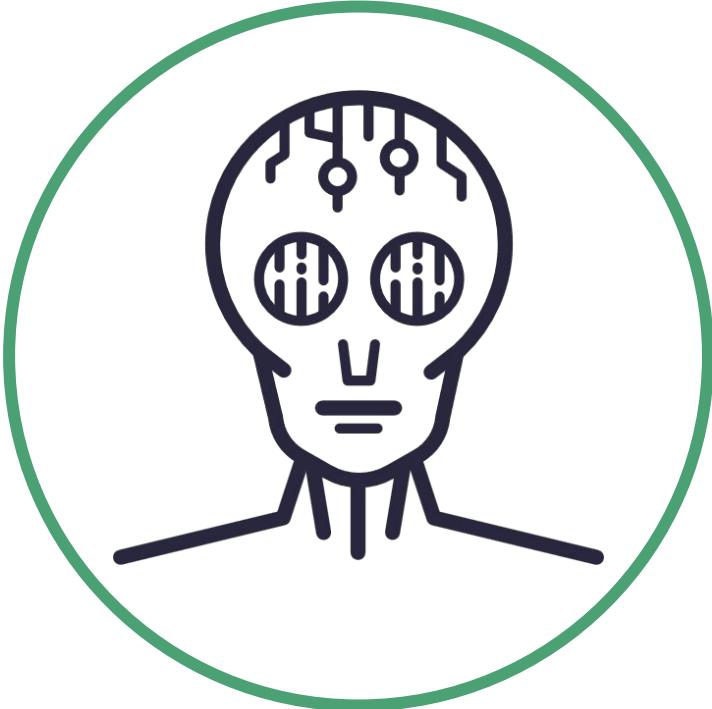


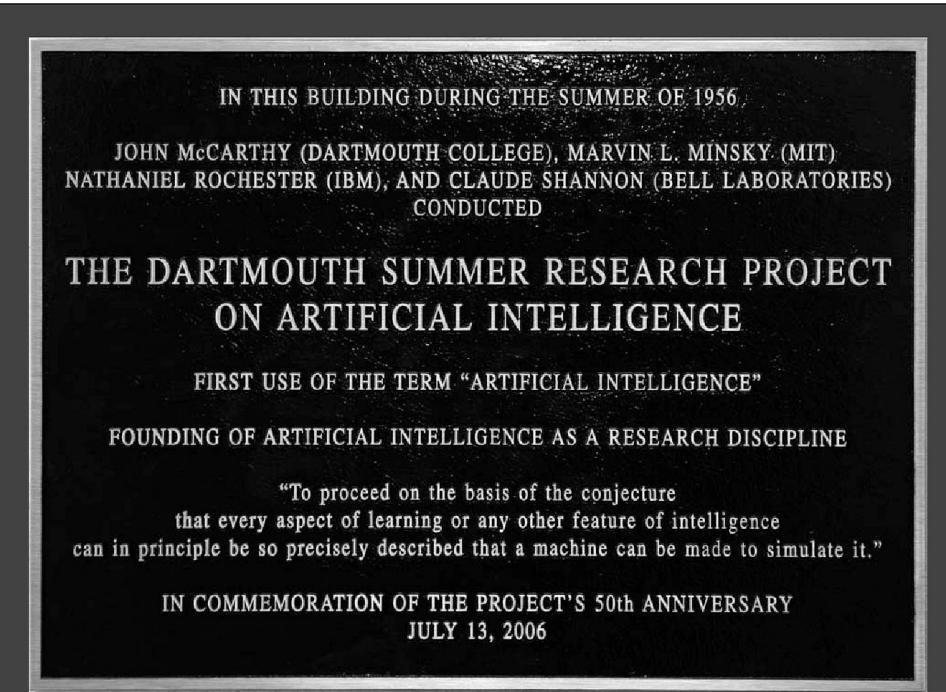
Figure 1. The neuron  $c_i$  is always marked with the numeral  $i$  upon the body of the cell, and the corresponding action is denoted by " $N^i$ " with  $i$  subscript, as in the text:

- (a)  $N_1(t) \equiv N_1(t-1)$ ;
- (b)  $N_2(t) \equiv N_1(t-1) \vee N_2(t-1)$ ;
- (c)  $N_3(t) \equiv N_1(t-1), N_2(t-1)$ ;
- (d)  $N_4(t) \equiv N_1(t-1), \sim N_2(t-1)$ ;
- (e)  $N_5(t) \equiv N_1(t-1), \vee N_2(t-3), \sim N_2(t-2)$ ;
- (f)  $N_6(t) \equiv N_1(t-2), N_2(t-1)$ ;
- (g)  $N_7(t) \equiv \sim N_1(t-1), N_2(t-1) \vee N_3(t-1), \vee N_1(t-1)$ .
- (h)  $N_8(t) \equiv N_2(t-2), N_3(t-1)$ ;
- (i)  $N_9(t) \equiv N_2(t-2), \sim N_1(t-3)$ ;
- (j)  $N_{10}(t) \equiv N_1(t-1), N_1(t-2)$ ;
- (k)  $N_{11}(t) \equiv N_2(t-1) \cdot \vee N_1(t-1), (Ex)t-1 \cdot N_1(x) \cdot N_2(x)$ .



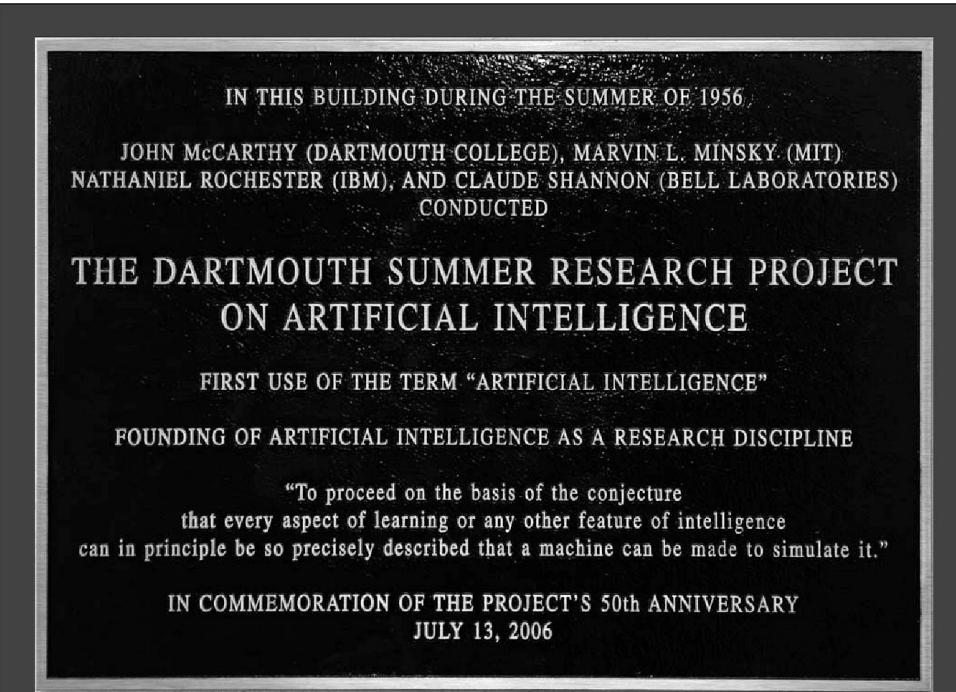
QUANDO SURGE  
O CONCEITO  
DE  
**INTELIGÊNCIA  
ARTIFICIAL?**

# Nascimento do termo “Inteligência Artificial” e a Conferência de Dartmouth



O termo “Inteligência Artificial” foi oficialmente cunhado em 1956, durante a **Conferência de Dartmouth**, organizada por John McCarthy, Marvin Minsky, Nathaniel Rochester e Claude Shannon. Realizada no Dartmouth College, nos Estados Unidos, essa conferência reuniu matemáticos, engenheiros e cientistas da computação **visando investigar como máquinas poderiam simular qualquer aspecto da inteligência**.

# Nascimento do termo “Inteligência Artificial” e a Conferência de Dartmouth

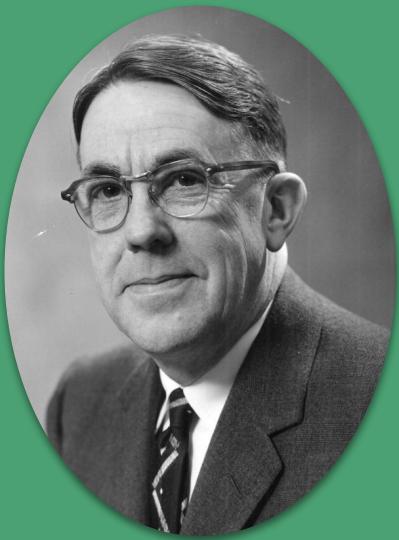


***“A ciência e a engenharia de fazer máquinas inteligentes.”***

- Representação de conhecimento e raciocínio lógico
- Aprendizado e adaptação de máquinas
- Resolução automática de problemas
- Simulação de processos cognitivos humanos

**Importância:** Considerada o **marco inicial da pesquisa em IA**, estabelecendo fundamentos teóricos para o desenvolvimento futuro da área.

# NASCIMENTO DO TERMO APRENDIZADO DE MÁQUINA

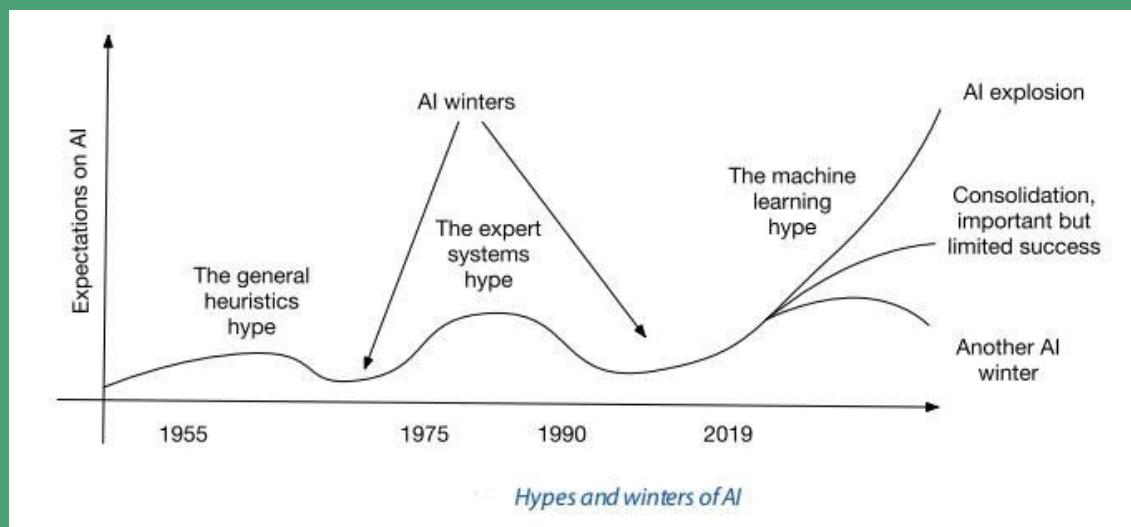


## Arthur Samuel (1901 - 1990)

Arthur Samuel gostava de jogar damas e escreveu um programa de computador para que ele pudesse jogar contra o computador. O problema, no entanto, é que sempre que ele jogava, ele vencia. Decidiu, então, escrever um programa que aprenderia estratégias a partir de jogos anteriores.

***“Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed” – (Arthur Samuel, 1959)***

# O INVERNO DA INTELIGÊNCIA ARTIFICIAL



**Expectativas exageradas** – Promessas não cumpridas sobre máquinas inteligentes.

**Limitações computacionais** – Computadores lentos e pouca memória.

**Falta de dados** – Poucos dados digitais disponíveis para aprendizado.

**Algoritmos imaturos** – Técnicas de aprendizado pouco eficientes.

**Corte de investimentos** – Financiadores reduziram recursos e pesquisas foram interrompidas.

# O INVERNO DA INTELIGÊNCIA ARTIFICIAL

## IBM 3090 (mainframe, 1985)

**Clock:** ~69 a 103 MHz (megahertz, não gigahertz)

**Processadores:** até 6 (em alguns modelos multiprocessados).

**RAM:** variava de 32 MB até 512 MB.

**Uso:** grandes bancos, seguradoras, órgãos governamentais, cálculos científicos.

**Tamanho:** uma sala inteira, consumindo kW de energia e exigindo refrigeração pesada.

**Preço:** dezenas de milhões de dólares.

## iPhone 15 Pro (smartphone, 2023)

**Clock:** até 3,78 GHz (cerca de 40x mais rápido por núcleo que o IBM 3090)

**Processadores:** 6 núcleos (2 de alto desempenho + 4 de eficiência).

**RAM:** até 8 GB (16x mais memoria).

**Uso:** cabe no bolso, roda apps de IA, gráficos 3D, vídeo 4K, internet.

**Tamanho:** ~200 gramas.

**Preço:** ~US\$ 1000.

**Anos 90**

# UMA NOVA ESPERANÇA: DEEP BLUE E KASPAROV (1997)

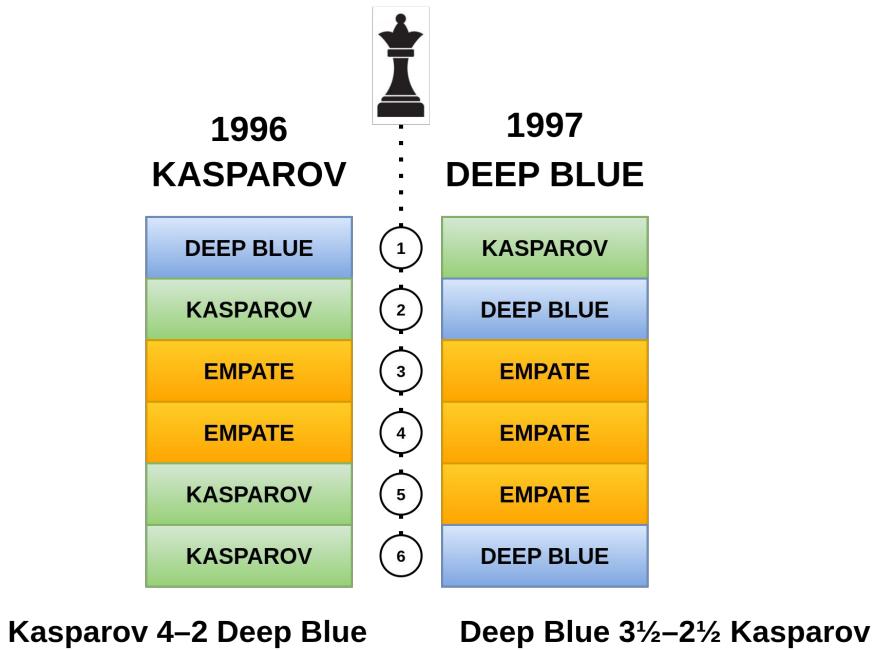


## Garry Kasparov – História Resumida

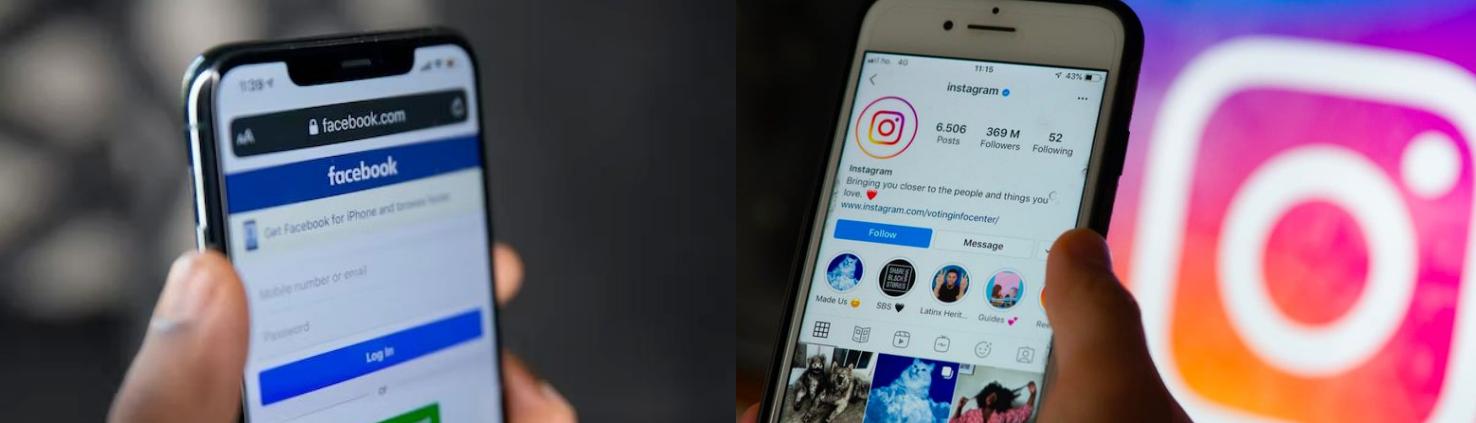
- Nascimento: 13 de abril de 1963, em Baku, Azerbaijão (então União Soviética)
- Campeão Mundial: Em 1985, aos 22 anos, tornou-se o mais jovem campeão mundial de xadrez da história ao derrotar Anatoly Karpov
- Em 1996 e 1997, Kasparov enfrentou o supercomputador Deep Blue, da IBM.
- Em 1997, perdeu o match histórico, marcando um ponto simbólico na evolução da inteligência artificial.

**Curiosidade:** Kasparov é muitas vezes citado como exemplo da mente humana contra a máquina, principalmente nas discussões sobre inteligência artificial e aprendizado de máquinas.

# UMA NOVA ESPERANÇA: DEEP BLUE E KASPAROV

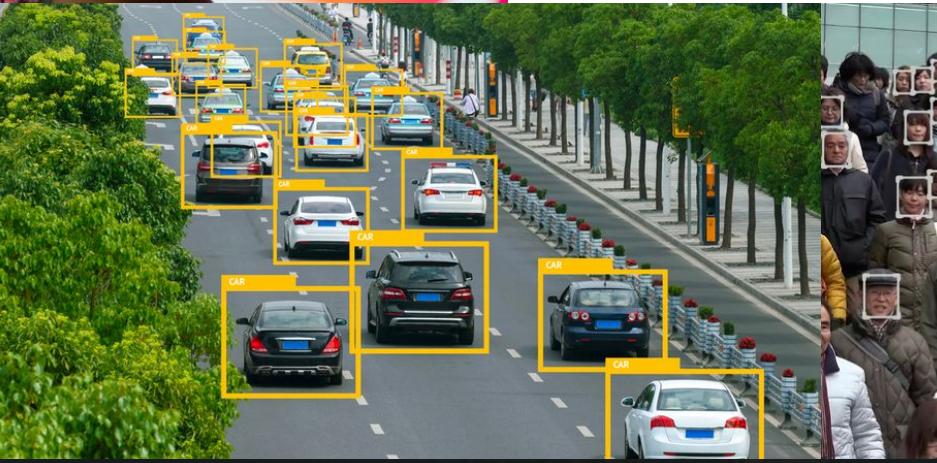


1997



how many episodes in season

Google S



2

3

[LOC]

Emmy-winning US TV Shows



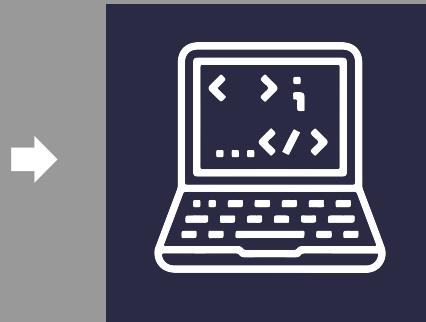
G

# Fundamentos do Aprendizado de Máquina (Machine Learning)

# PROGRAMAÇÃO TRADICIONAL



DADOS  
MÓDULO



PROGRAMAÇÃO  
TRADICIONAL

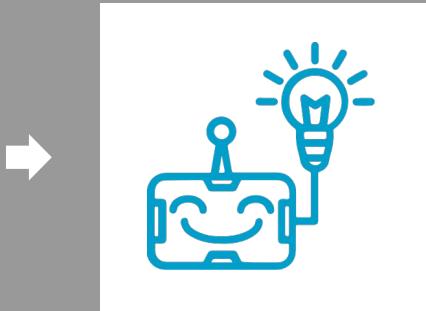


OUTPUT

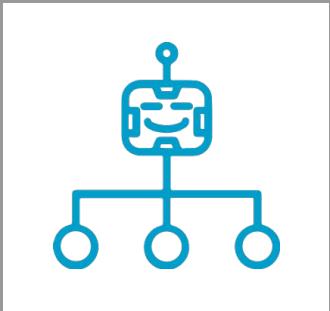
# APRENDIZADO DE MÁQUINA



DADOS  
SAÍDA

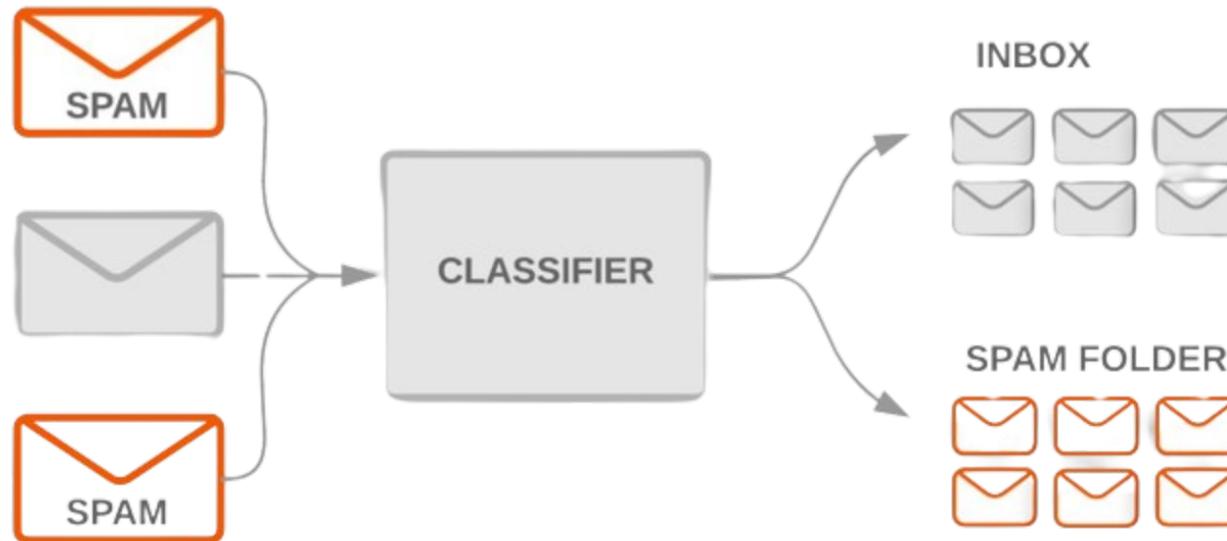


APRENDIZADO  
DE MÁQUINA



MÓDULO

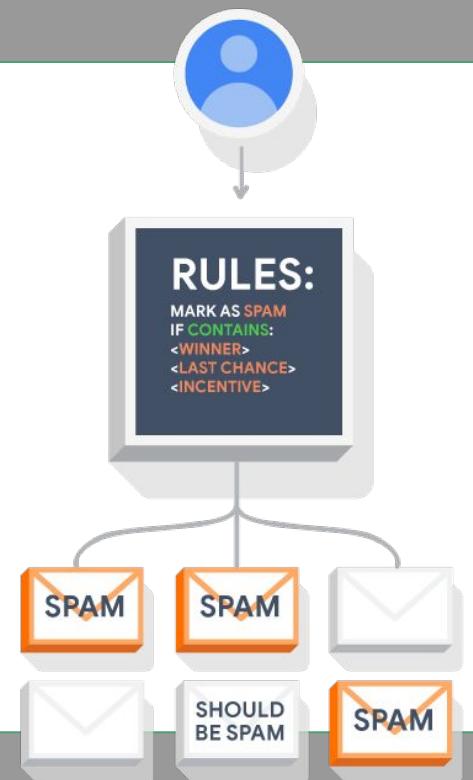
# PROGRAMAÇÃO TRADICIONAL VS APRENDIZADO DE MÁQUINA



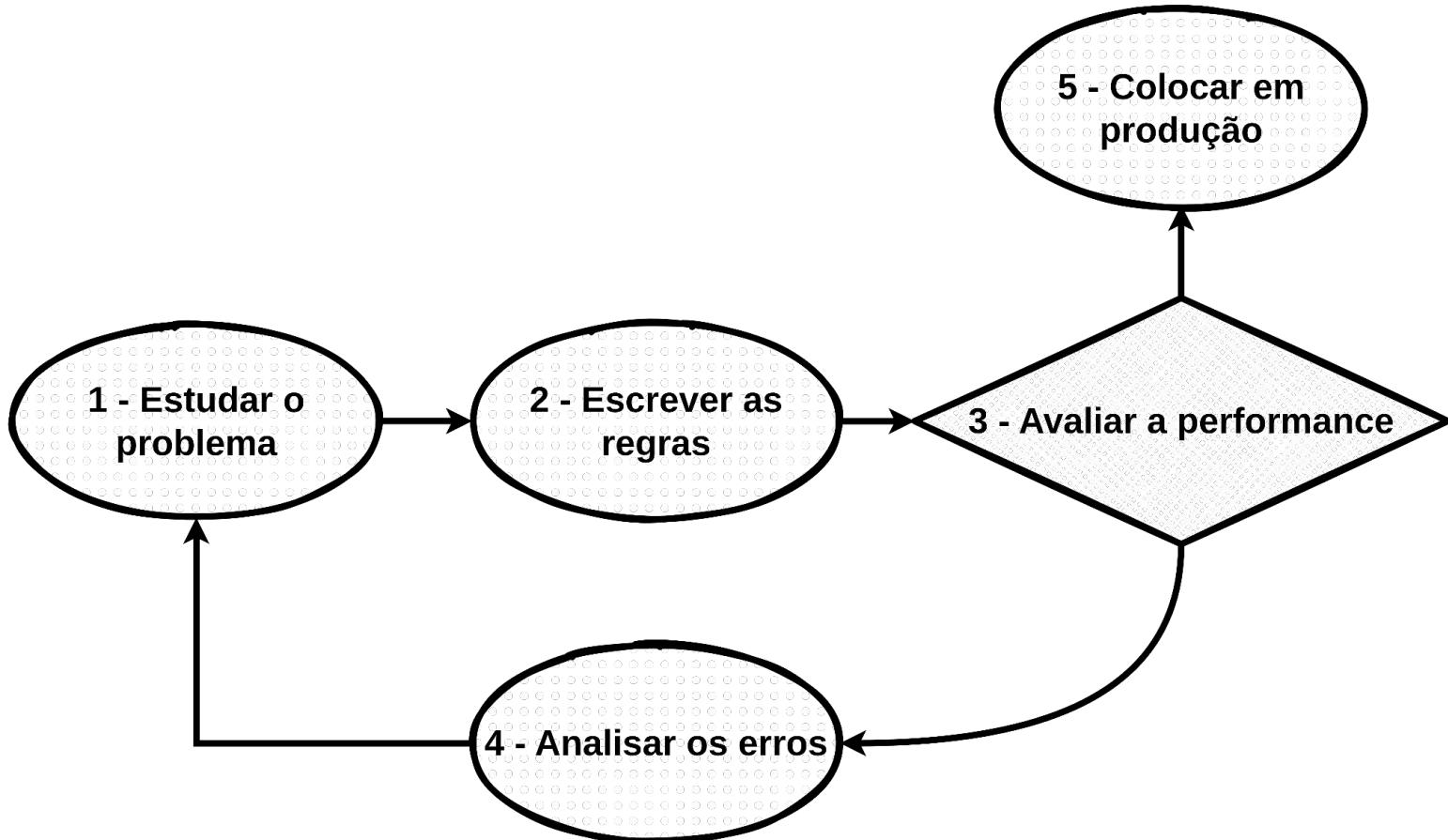
# PROGRAMAÇÃO TRADICIONAL VS APRENDIZADO DE MÁQUINA

## SOLUÇÃO 1

Na programação tradicional, a máquina uma programação explícita de um conjunto definido de regras para obter a saída desejada.



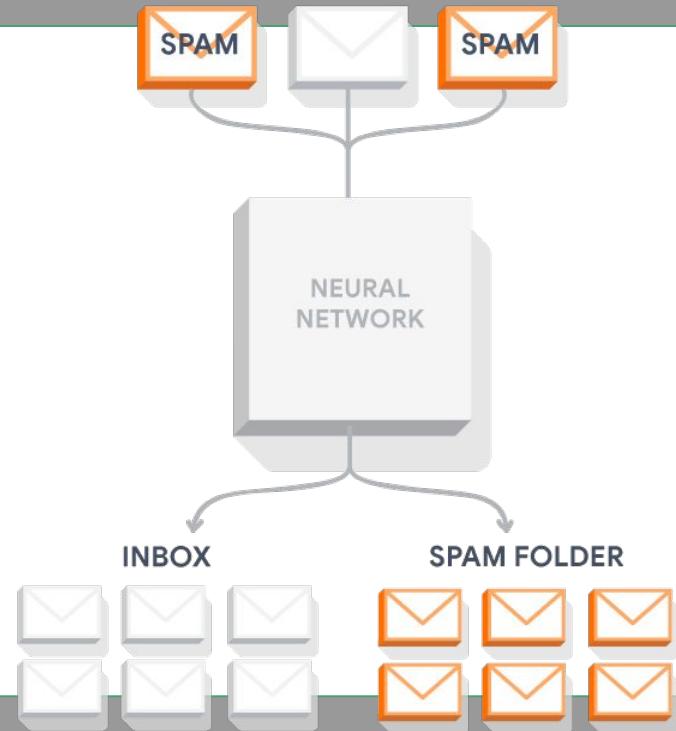
# PROGRAMAÇÃO TRADICIONAL



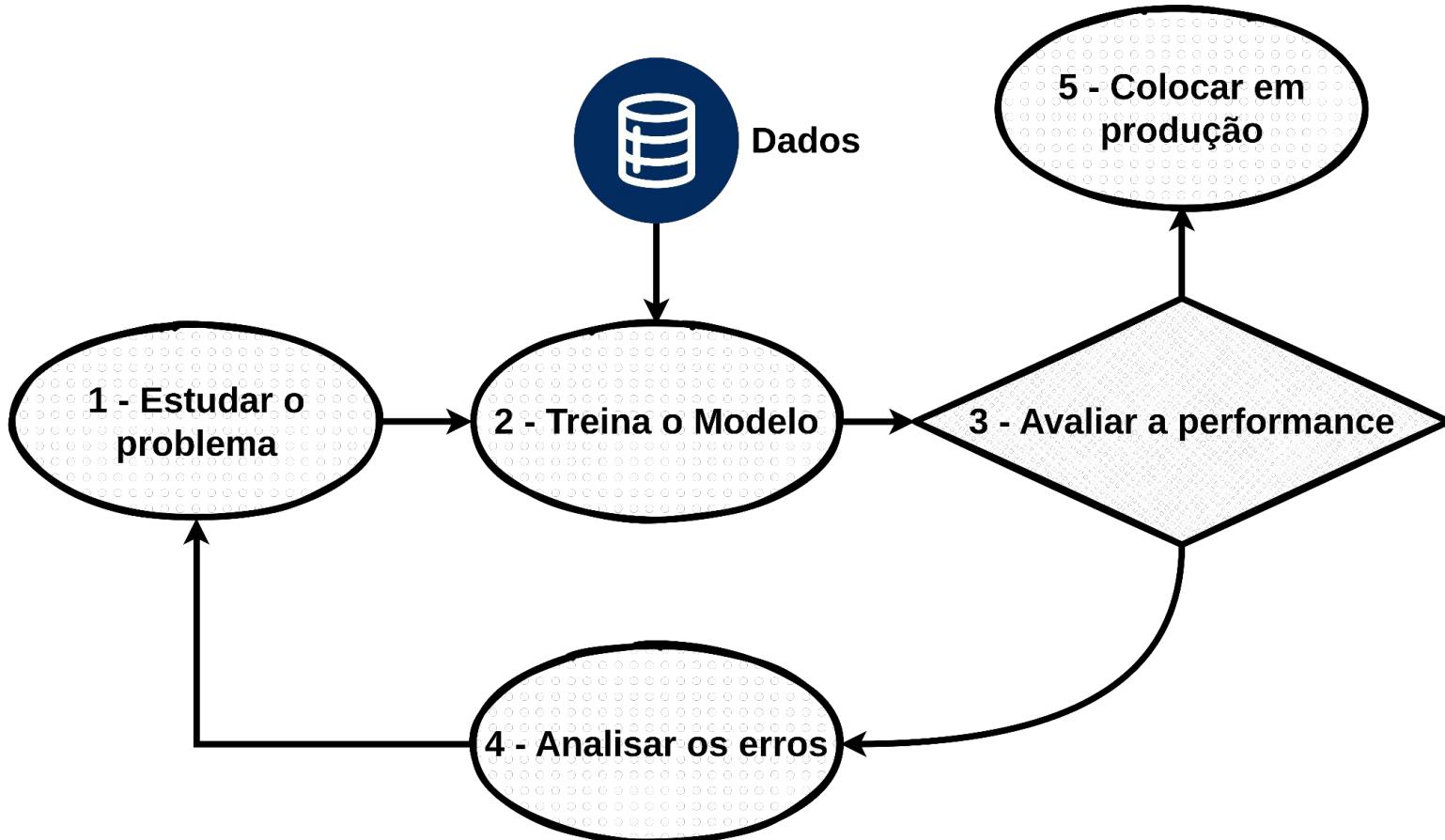
# PROGRAMAÇÃO TRADICIONAL VS APRENDIZADO DE MÁQUINA

## SOLUÇÃO 2

Os algoritmos de Aprendizado de máquina, por sua vez, aprendem automaticamente com os dados e, com base nesse aprendizado, gera seu próprio conjunto de regras (i.e. o modelo).



# APRENDIZADO DE MÁQUINA



“Um programa de computador aprende se ele é capaz de melhorar seu **desempenho (P)** em determinada **tarefa (T)**, sob alguma medida de **avaliação (P)**, a partir de **experiências passadas (E)**.”

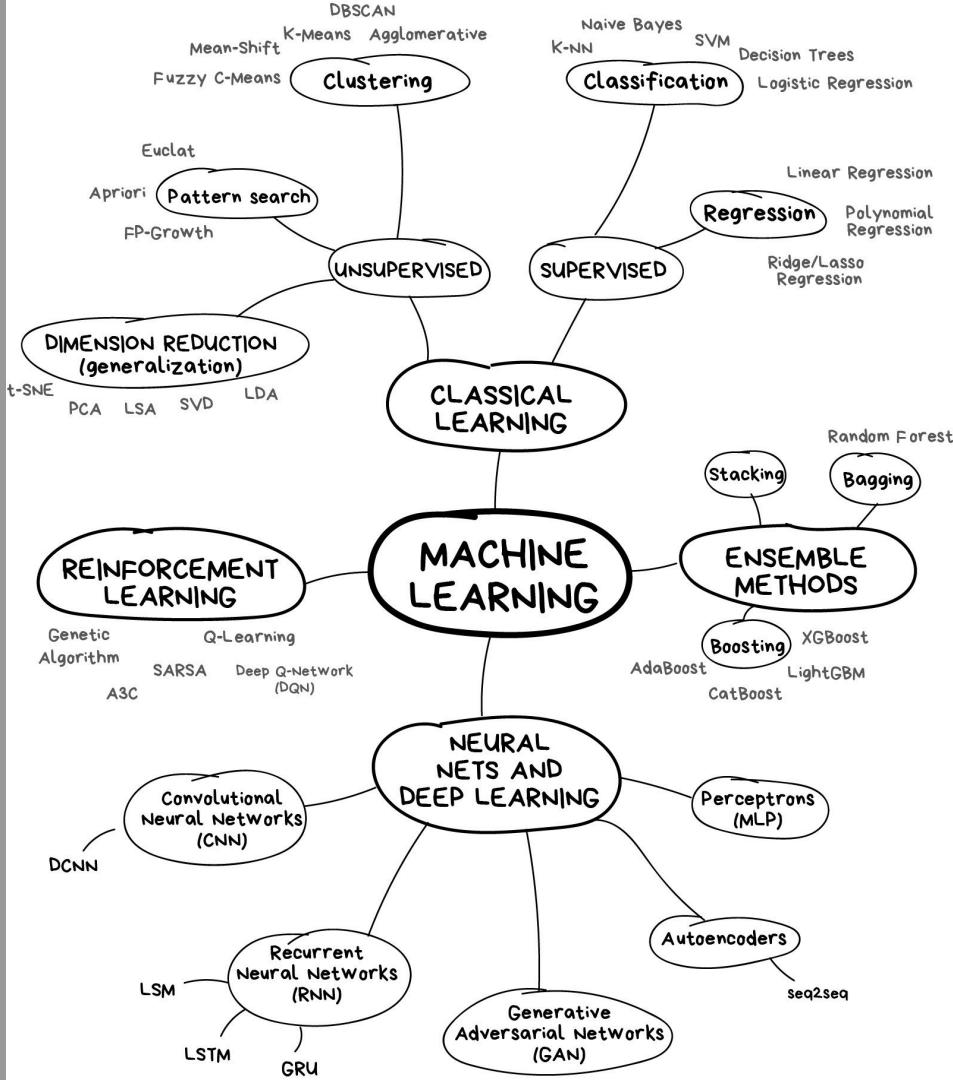
(Tom Mitchell)



# APRENDIZADO DE MÁQUINA

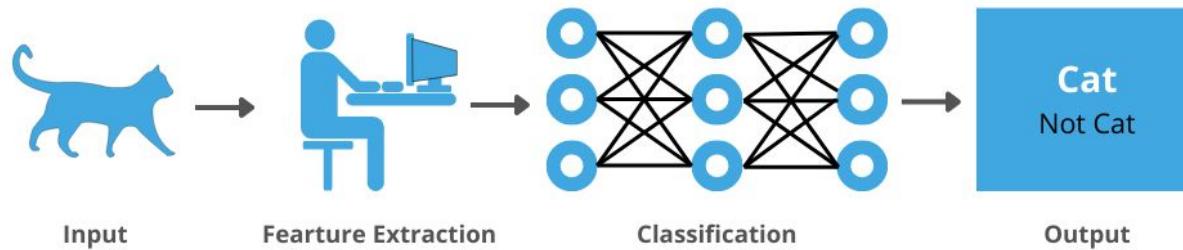
Mas, o que pode ser feito com cada uma dessas técnicas?

Quais são as aplicações?



# Aprendizado Profundo (Deep Learning)

## Machine Learning



## Deep Learning



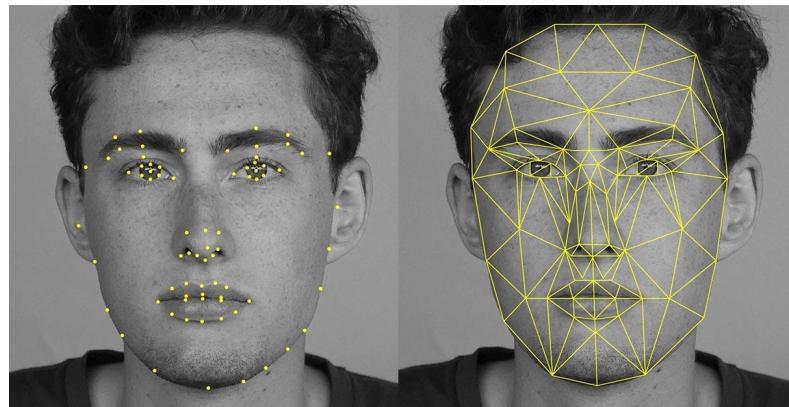
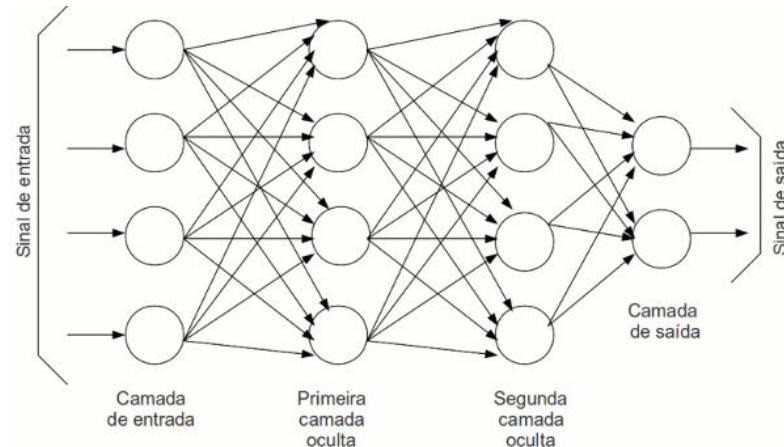
# REDES NEURAIS ARTIFICIAIS

## DEFINIÇÃO

Toda rede neural é uma coleção de **neurônios** e conexões entre eles.

## EXEMPLOS

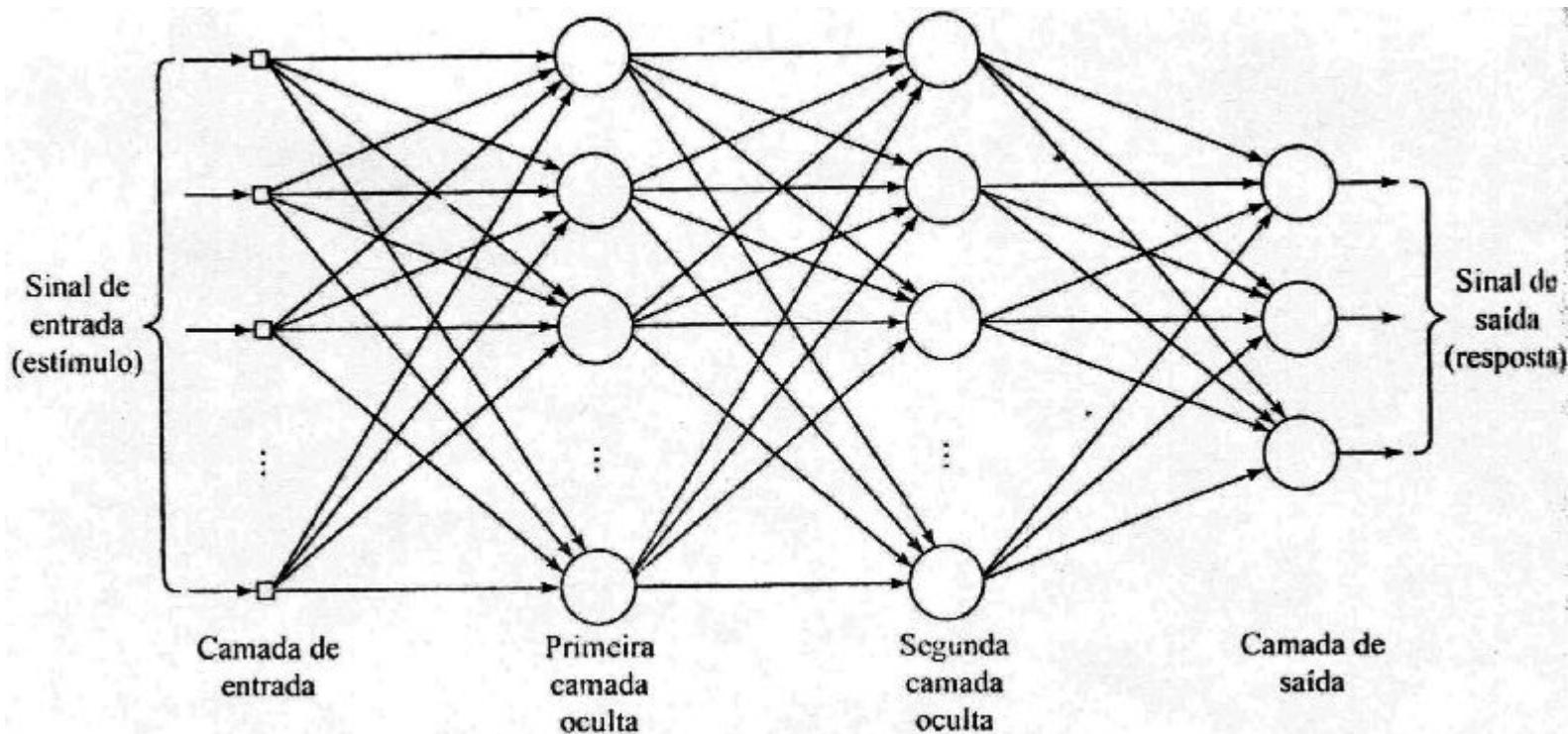
- Reconhecimento facial
- detecção de objetos
- reconhecimento de fala
- tradução



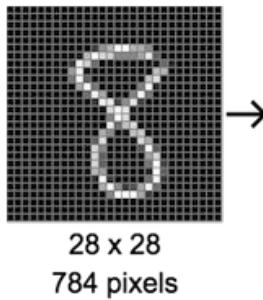
# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS



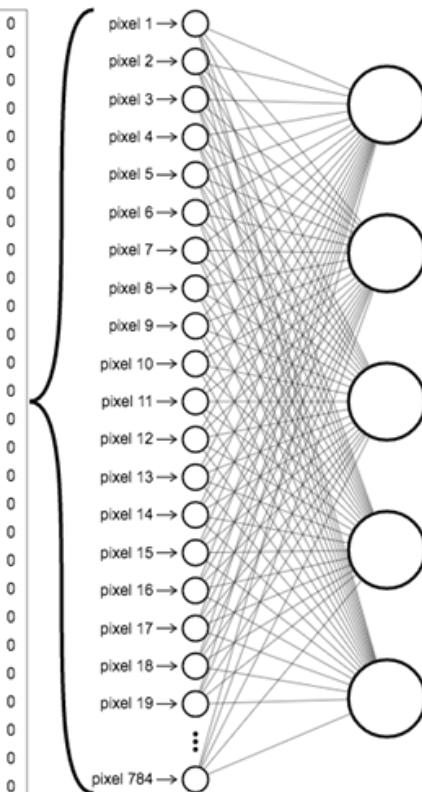
# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS



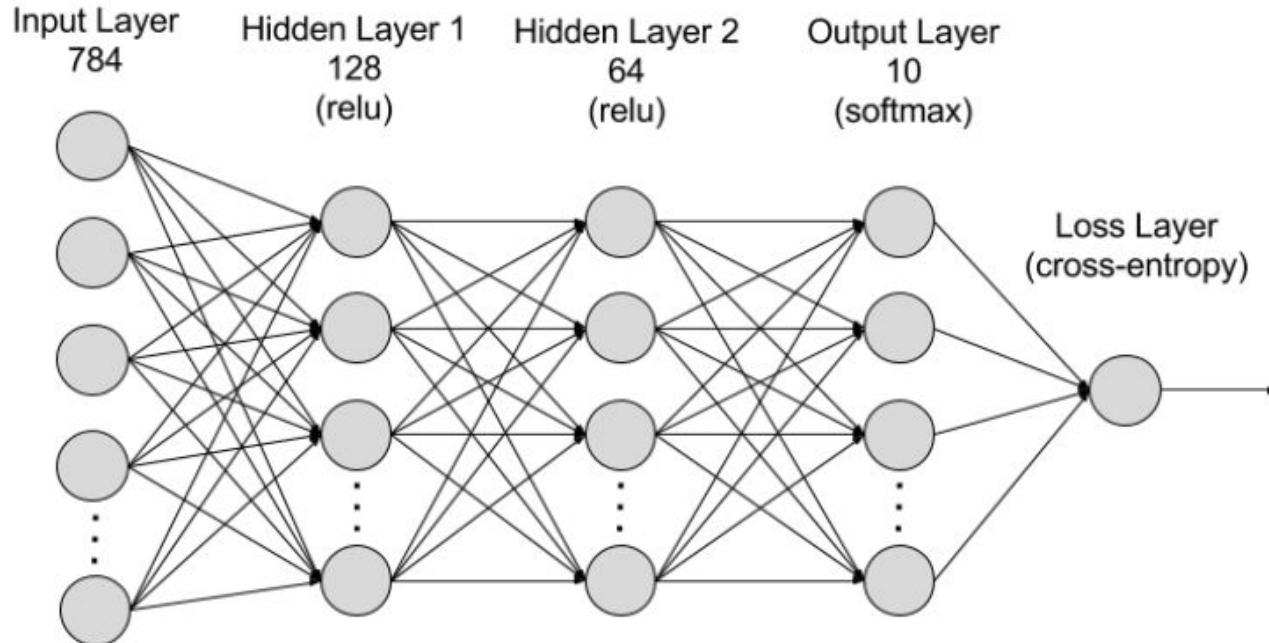
# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS



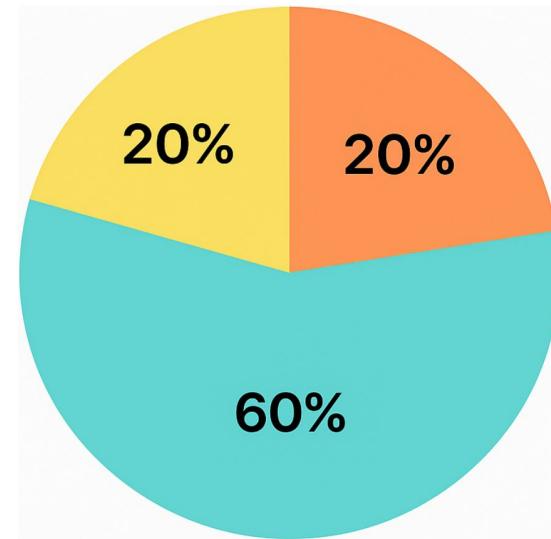
28 x 28  
784 pixels



# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS

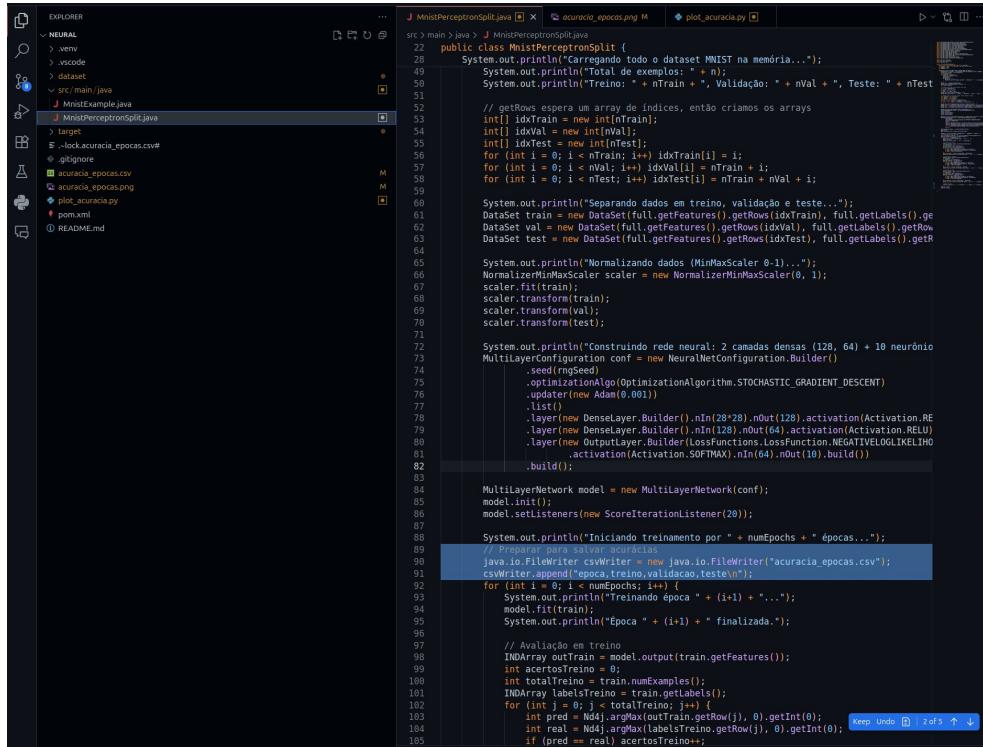


# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS



10.000 - Dígitos  
manuscritos

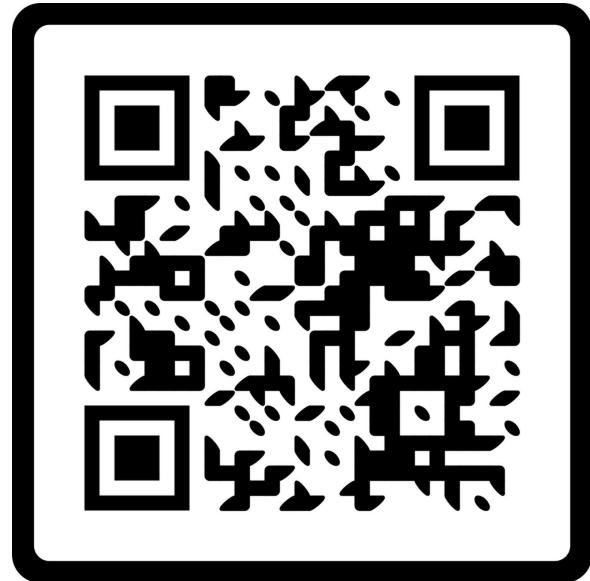
# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS



The screenshot shows a Java development environment with the following details:

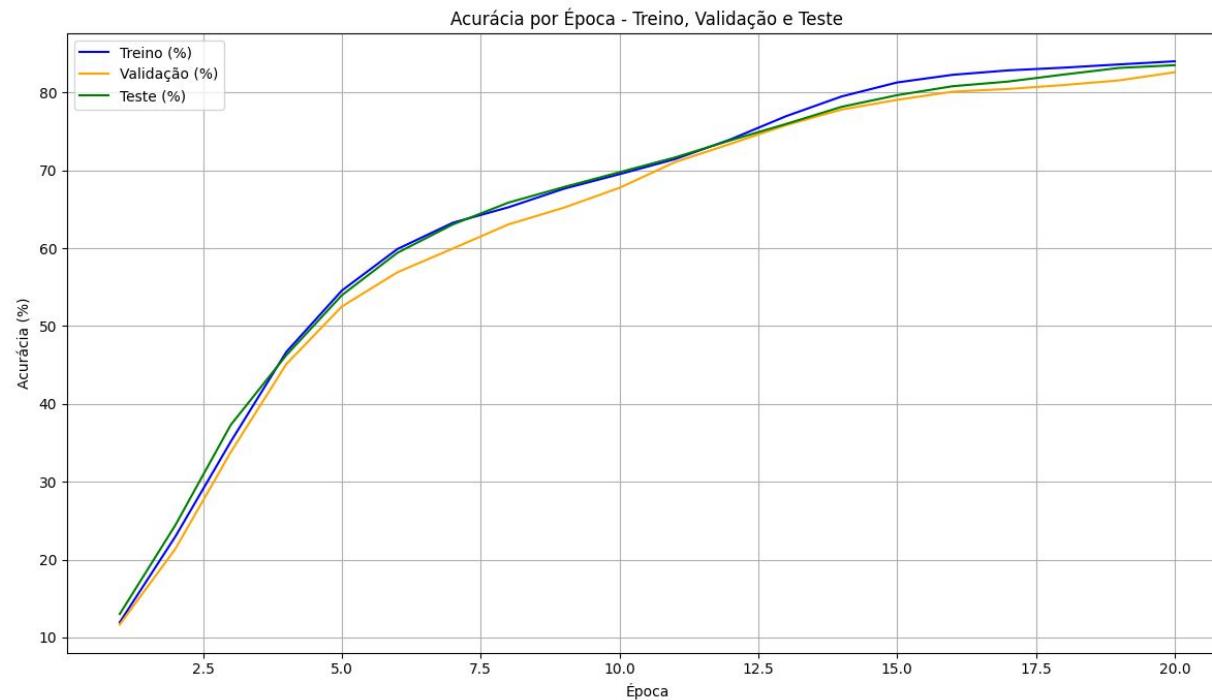
- Project Structure:** NEURAL, .venv, .vscode, dataset, src/main/java.
- Opened File:** MnistPerceptronSplit.java
- Code Preview:** The code implements a neural network for digit recognition using the MNIST dataset. It includes data loading, normalization, and training logic. A specific section of the code is highlighted in blue, showing the training loop and accuracy logging.

```
src > main > java > J MnistPerceptronSplit.java
22 public class MnistPerceptronSplit {
23     System.out.println("Carregando todo o dataset MNIST na memória...");
24     System.out.println("Total de exemplos: " + n);
25     System.out.println("Treino: " + ntrain + ", Validação: " + nVal + ", Teste: " + nTest);
26
27     // getRows espera um array de índices, então criamos os arrays
28     int[] idxTrain = new int[nTrain];
29     int[] idxVal = new int[nVal];
30     int[] idxTest = new int[nTest];
31
32     for (int i = 0; i < nTrain; i++) idxTrain[i] = i;
33     for (int i = 0; i < nVal; i++) idxVal[i] = nTrain + i;
34     for (int i = 0; i < nTest; i++) idxTest[i] = nTrain + nVal + i;
35
36     System.out.println("Separando dados em treino, validação e teste...");
37     DataSet train = new DataSet(full.getFeatures(), getRows(idxTrain), full.getLabels().getRows(idxTrain));
38     DataSet val = new DataSet(full.getFeatures(), getRows(idxVal), full.getLabels().getRows(idxVal));
39     DataSet test = new DataSet(full.getFeatures(), getRows(idxTest), full.getLabels().getRows(idxTest));
40
41     System.out.println("Normalizando dados (MinMaxScaler 0-1)...");
42     NormalizerMinMaxScaler scaler = new NormalizerMinMaxScaler(0, 1);
43     scaler.fit(train);
44     scaler.transform(train);
45     scaler.transform(val);
46     scaler.transform(test);
47
48     System.out.println("Construindo rede neural: 2 camadas densas (128, 64) + 10 neurônio");
49     MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
50         .seed(rngSeed)
51         .optimizationAlgorithm(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
52         .updater(Updater.Adam(0.001))
53         .list()
54         .layer(new DenseLayer.Builder().nIn(28*28).nOut(128).activation(Activation.RELU))
55         .layer(new DenseLayer.Builder().nIn(128).nOut(64).activation(Activation.RELU))
56         .layer(new OutputLayer.Builder().nIn(64).nOut(10).build());
57
58     conf.setLms();
59     conf.setEta(0.01);
60     conf.setBackprop(true);
61     conf.setLearnRate(0.01);
62     conf.setBatchSize(100);
63     conf.setNesterov(true);
64
65     MultilayerNetwork model = new MultilayerNetwork(conf);
66     model.init();
67     model.setListeners(new ScoreIterationListener(20));
68
69     System.out.println("Iniciando treinamento por " + numEpochs + " épocas...");
70     // Progresso para validar acurácia
71     java.io.FileWriter csvWriter = new java.io.FileWriter("acuracia_epocas.csv");
72     csvWriter.append("epoca,treino,validacao,teste\n");
73
74     for (int i = 0; i < numEpochs; i++) {
75         System.out.println("Treinando época " + (i+1) + "...");
76         model.fit(train);
77         System.out.println("Epoca " + (i+1) + " finalizada.");
78
79         // Avaliação em treino
80         INDArray outTrain = model.output(train.getFeatures());
81         int acertosTreino = 0;
82         int totalTreino = train.numExamples();
83         INDArray maxLabelsTreino = train.maxLabels();
84
85         for (int j = 0; j < totalTreino; j++) {
86             int pred = Nd4j.argmax(outTrain.getRow(j), 0).getInt(0);
87             int real = Nd4j.argmax(maxLabelsTreino.getRow(j), 0).getInt(0);
88             if (pred == real) acertosTreino++;
89
90         }
91
92     }
93
94 }
```

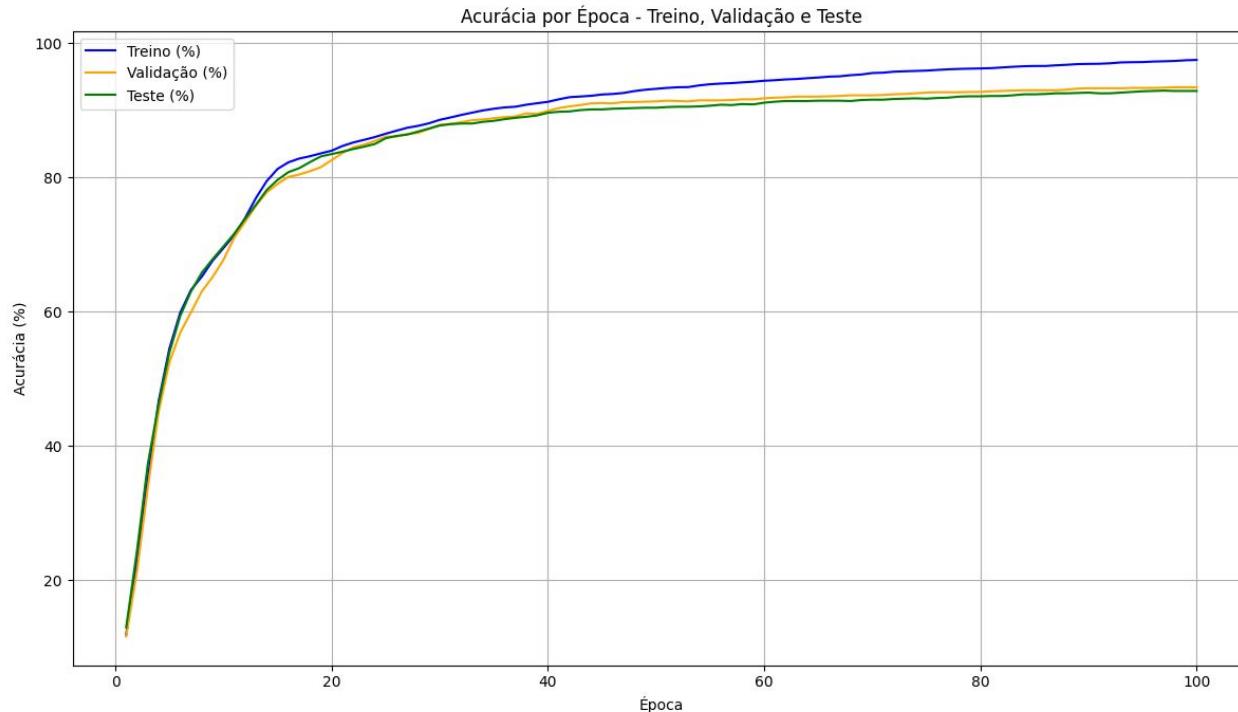


SCAN ME

# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS

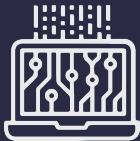


# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS



# Métricas de Avaliação

O que podemos  
melhorar?



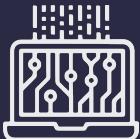
## ACURÁCIA

desempenho geral do modelo. Dentre **todas** as classificações, quantas o modelo acertou



## PRECISÃO

dentre todas as classificações de classe Positivo **que o modelo fez**, quantas estão corretas;



## RECALL

dentre todas as situações de classe Positivo **como valor esperado**, quantas estão corretas;

### ACURÁCIA:

$$\frac{VP + VN}{VP + VN + FP + FN}$$

### PRECISÃO:

$$\frac{VP}{VP + FP}$$

### RECALL:

$$\frac{VP}{VP + Fn}$$

### LEGENDA:

**VP:** verdadeiro positivo

**VN:** verdadeiro negativo

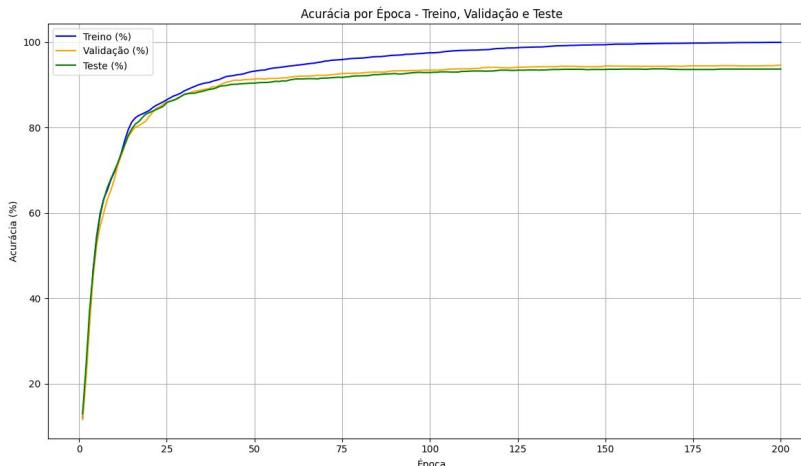
**FP:** falso positivo

**FN:** falso negativo

# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS

## Aumentar o número de épocas

O treinamento está limitado a 20 épocas. Tente aumentar para 50, 100 ou até 150, monitorando para evitar overfitting.



```
Época 196 finalizada.  
Acurácia treino após época 196: 0.9993333333333333 (99.93%)  
Acurácia validação após época 196: 0.945 (94.50%)  
Acurácia teste após época 196: 0.937 (93.70%)  
Treinando época 197...  
Época 197 finalizada.  
Acurácia treino após época 197: 0.9995 (99.95%)  
Acurácia validação após época 197: 0.945 (94.50%)  
Acurácia teste após época 197: 0.937 (93.70%)  
Treinando época 198...  
Época 198 finalizada.  
Acurácia treino após época 198: 0.9995 (99.95%)  
Acurácia validação após época 198: 0.945 (94.50%)  
Acurácia teste após época 198: 0.937 (93.70%)  
Treinando época 199...  
Época 199 finalizada.  
Acurácia treino após época 199: 0.9995 (99.95%)  
Acurácia validação após época 199: 0.946 (94.60%)  
Acurácia teste após época 199: 0.937 (93.70%)  
Treinando época 200...  
Época 200 finalizada.  
Acurácia treino após época 200: 0.9995 (99.95%)  
Acurácia validação após época 200: 0.946 (94.60%)  
Acurácia teste após época 200: 0.937 (93.70%)
```

[THEA]

# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS

## Ajustar a arquitetura da rede

**Adicione mais camadas ocultas ou aumente o número de neurônios.**

**2 camadas ocultas com 256, 128 na camada escondida**

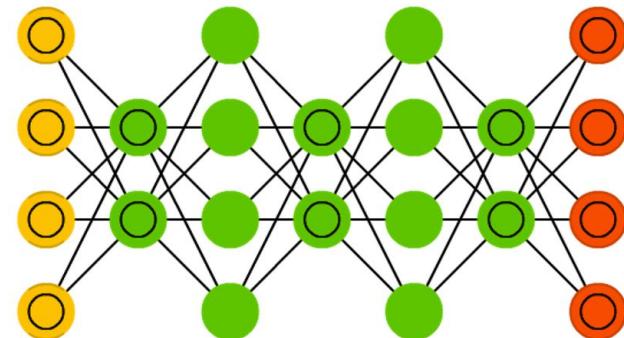
```
Época 9 finalizada.  
Acurácia treino após época 9: 0.8231666666666667 (82.32%)  
Acurácia validação após época 9: 0.8055 (80.55%)  
Acurácia teste após época 9: 0.8135 (81.35%)  
Treinando época 10...  
Época 10 finalizada.  
Acurácia treino após época 10: 0.8353333333333334 (83.53%)  
Acurácia validação após época 10: 0.8135 (81.35%)  
Acurácia teste após época 10: 0.8335 (83.35%)  
[ENTER]
```

```
Época 9 finalizada.  
Acurácia treino após época 9: 0.6763333333333333 (67.63%)  
Acurácia validação após época 9: 0.652 (65.20%)  
Acurácia teste após época 9: 0.6785 (67.85%)  
Treinando época 10...  
Época 10 finalizada.  
Acurácia treino após época 10: 0.695 (69.50%)  
Acurácia validação após época 10: 0.6775 (67.75%)  
Acurácia teste após época 10: 0.6975 (69.75%)  
[ENTER]
```

# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS

## Ajustar a arquitetura da rede

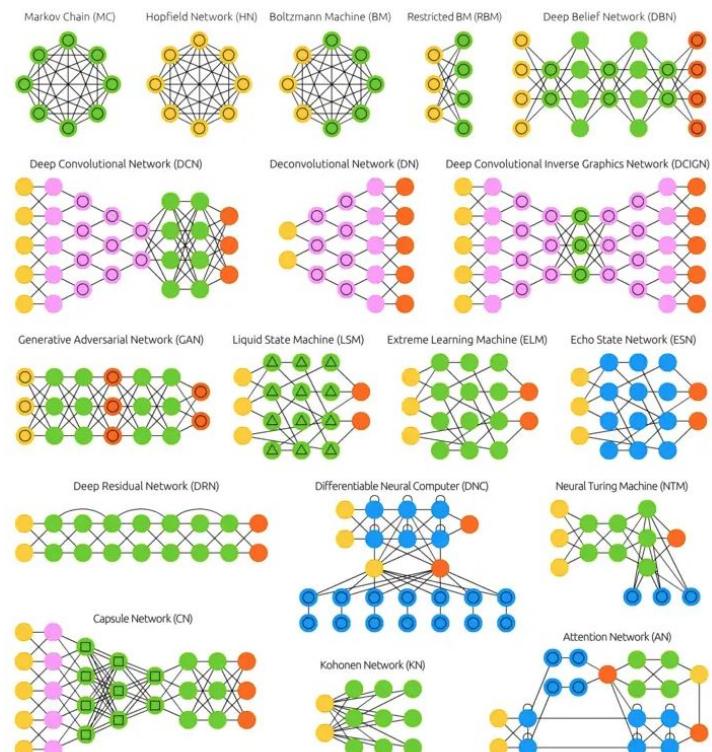
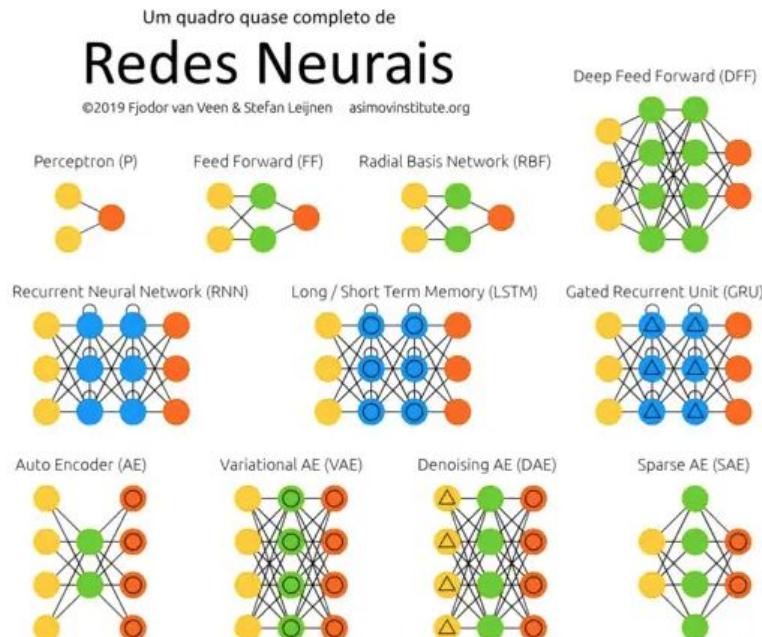
Mudar o tipo da rede neural, perceptron para Deep belief network (DBN) - Construindo rede neural: 6 camadas densas (512, 256, 128, 64, 32, 16) + 10 neurônios de saída.



```
Época 9 finalizada.  
Acurácia treino após época 9: 0.5678333333333333 (56.78%)  
Acurácia validação após época 9: 0.569 (56.90%)  
Acurácia teste após época 9: 0.5485 (54.85%)  
Treinando época 10...  
Época 10 finalizada.  
Acurácia treino após época 10: 0.5935 (59.35%)  
Acurácia validação após época 10: 0.5965 (59.65%)  
Acurácia teste após época 10: 0.572 (57.20%)
```

# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS

## Ajustar a arquitetura da rede



# EXEMPLO: RECONHECIMENTO DE DÍGITOS NUMÉRICOS

## Adicionar regularização

Use Dropout ou L2 regularization para evitar overfitting.

## Ajustar taxa de aprendizado

Teste valores menores para o Adam, como 0.0005 ou 0.0001.

## Aumentar o tamanho do batch

Experimente valores como 256 ou 512, se houver memória suficiente.

## Data augmentation

Para MNIST, pequenas rotações ou deslocamentos podem ajudar, mas isso exigiria modificar o pipeline de dados.

# Mercado de trabalho

Cientista de Dados	Engenheiro de Dados
Exploração e análise dos dados.	Infraestrutura e processamento de dados.
Cria modelos estatísticos e de Machine Learning. Descobre padrões, gera previsões e <i>insights</i> de negócio.	Constrói pipelines de dados (ETL/ELT). Garante qualidade, escalabilidade e segurança dos dados.
Relatórios analíticos, dashboards, modelos preditivos.	SQL, Python, Spark, Databricks, AWS/Azure/GCP.
Mais analítico, próximo do negócio.	Mais técnico, próximo da arquitetura de sistemas.
R\$ 9k–17k/mês (pleno).	R\$ 8k–15k/mês (pleno).

# Mercado de trabalho



O **Engenheiro de Dados** garante que os **dados estejam organizados, limpos e acessíveis**



O **Cientista de Dados** usa esses dados para **criar inteligência e apoiar decisões estratégicas**.

McKinsey & Company

Careers Home Explore Jobs Interviewing Our People Students Tech Careers Careers Beyond Consulting Experienced Professionals EN

Filter By: Locations Interests Industries Capabilities Brazil Technology Clear All

8 Jobs Available

**Data Engineer - QuantumBlack**

You will be part of interdisciplinary teams of data scientists, engineers, designers, and business experts to help solve complex and impactful challenges using data. As a Data Engineer, you will be responsible for designing and building the data...

Sao Paulo

**Junior Data Engineer - QuantumBlack**

You will work in multi-disciplinary teams harnessing data to deliver real-world impact for organizations globally. In this role, you will assist in designing and building data pipelines that support data science projects, following...

Sao Paulo

**Junior Data Scientist - QuantumBlack**

As a Junior Data Scientist at QuantumBlack, you'll be part of a multi-disciplinary team using data to drive real-world impact for leading organizations across the globe. In this role, you'll help solve high-impact business problems by...

Sao Paulo

**Principal Data Engineer I - Platform GenAI**

As a Principal Data Engineer, you will play a pivotal role in shaping the technical foundation of our data infrastructure while driving innovation and excellence within our global Platform GenAI team. Your expertise will guide critical design...

Lisbon | Prague | San Jose | Sao Paulo

Refine your search

Brazil Worldwide Technology and Engineering Clear All

Country/Region +

City +

Category +

Company +

Create Job Alert

NOTE: Use refine search filters above to get better job alerts

Email Address

Enter email address

You'll get emails

Weekly

Please review BCG's recruiting privacy policy, which we will update from time to time, for additional information.

I consent to receive email and/or SMS messaging communications from BCG about insights and career opportunities.

I agree to BCG retaining my personal information for up to five years including any sensitive information I choose to share for future employment opportunities as part of BCG's optional recruiting programs.

By proceeding, I understand that my personal data will be processed in accordance with the Company Data Privacy Policy.

Create Job Alert

Manage Alerts

Search jobs, skills or qualifications

5 results | Sort by Most relevant

**Software Engineer, Brazil - X Delivery**

Available in 2 locations Associated with 2 categories 51534

Join a dynamic team where you'll leverage your software development expertise to bridge technology and strategy, collaborating on innovative international projects. Bring your strong coding skills in multiple languages, web development experience, and cloud computing knowledge to create impactful solutions in a fast-paced environment.

Apply Now

**Data Engineer, Brazil - X Delivery**

Available in 2 locations Associated with 2 categories 51532

Seeking skilled professionals with expertise in data engineering to design and implement robust data pipelines. Ideal candidates should possess strong analytical skills, experience with data modeling, and proficiency in relevant programming languages. Join a dynamic team dedicated to transforming data into actionable insights.

Apply Now

**Software Engineer, Salesforce, Brazil - X Delivery**

Sao Paulo, Sao Paulo, Brazil Associated with 2 categories 51531

Seeking a dynamic Salesforce Developer to bridge technology and strategy in groundbreaking projects. Leverage your expertise in Salesforce clouds, Apex, and Agile methodologies to drive impactful solutions. Join a diverse team dedicated to creating innovative solutions that transform industries and enhance lives.

Apply Now

**Software Engineer, QA, Brazil - X Delivery**

Available in 2 locations Associated with 2 categories 54795

We are looking for a skilled Software Developer in Test to enhance software quality through automated testing and CI/CD processes. Join a dynamic team in Sao Paulo or Rio de Janeiro, leveraging your expertise in Python, JavaScript, and testing frameworks to drive impactful solutions.

Apply Now

**DevOps Software Engineer, Brazil - X Delivery**

Available in 2 locations Associated with 2 categories 54792

Seeking a collaborative DevOps expert to drive innovative software solutions in a dynamic environment. Leverage your skills in cloud technologies, CI/CD, and containerization while enhancing project efficiency. Join a diverse team dedicated to tackling complex challenges and making a positive impact on

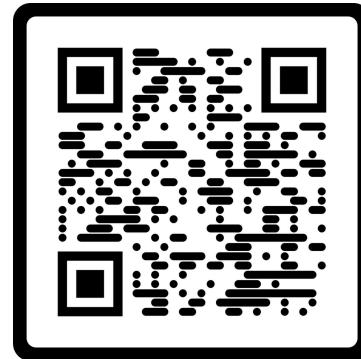
Apply Now

Hi Are you looking for a job?

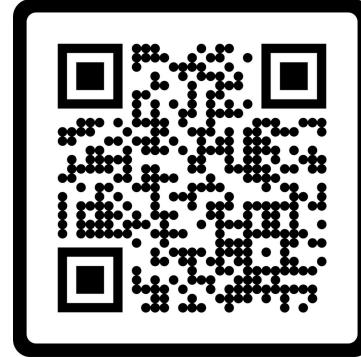
Find a job Ask a question

## CONTEÚDO

- Profissões que estarão em alta em 2025 para ganhar mais
  - 1. Analista de Business Intelligence (BI)
  - 2. Engenheiro de dados
  - 3. Engenheiro de software**
  - 4. Analista de privacidade
  - 5. Gerente de projetos
  - 6. Especialistas em marketing digital
  - 7. Especialista em CX
  - 8. Growth hacker
  - 9. Analista de desenvolvimento de sistemas
  - 10. Engenheiro de cibersegurança
  - 11. Especialista em inteligência artificial
  - 12. Desenvolvedor de realidade aumentada (AR) e realidade virtual (VR)
  - 13. Especialista em segurança da informação
  - 14. Consultor de ESG (ambiental, social e governança)
  - 15. Técnico em energias renováveis
  - 16. Analista de sustentabilidade
  - 17. Profissional de telemedicina
  - 18. Especialista em experiência do usuário (ux designer)
  - 19. Analista de cloud computing
  - 20. Especialista em automação industrial



SCAN ME



SCAN ME

# Transformers

# Inteligência Artificial

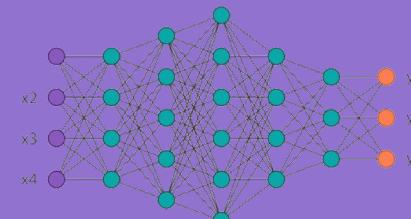
Programas com habilidade de agir como humanos.



1950

# Aprendizado profundo

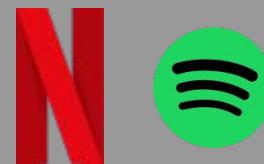
Redes neurais artificiais que aprendem através de um grande volume de dados.



2010

# Aprendizado de máquina

Algoritmos com habilidade de aprender sem programação expressa.



1980

# Transformers

Criação de novos conteúdos, como texto, imagens, música, áudio e vídeos.



2020

# Attention Is All You Need



## Attention Is All You Need

Ashish Vaswani\*  
Google Brain  
avaswani@google.com

Noam Shazeer\*  
Google Brain  
noam@google.com

Niki Parmar\*  
Google Research  
nikip@google.com

Jakob Uszkoreit\*

Google Research  
usz@google.com

Llion Jones\*  
Google Research  
llion@google.com

Aidan N. Gomez\* †  
University of Toronto  
aidan@cs.toronto.edu

Lukasz Kaiser\*  
Google Brain  
lukasz.kaiser@google.com

Ilia Polosukhin\* ‡  
illia.polosukhin@gmail.com

### Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

### 1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

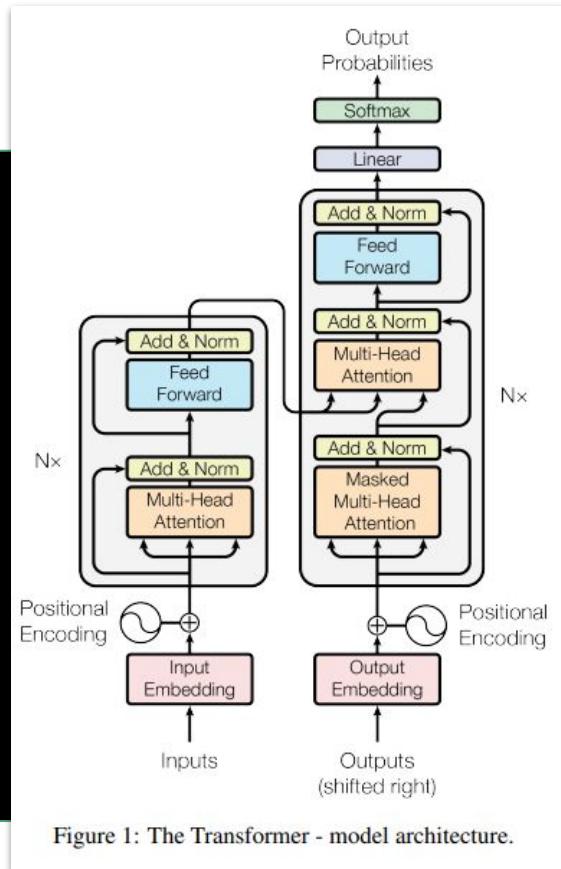
\*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

Transformer utiliza **self-attention** para capturar relações entre todos os elementos de uma sequência simultaneamente, permitindo entender dependências de longo alcance de forma eficiente. Combinado com **multi-head attention** e **positional encoding**, ele consegue analisar diferentes aspectos do contexto e manter a ordem dos tokens. A arquitetura **encoder-decoder** proposta possibilita tarefas como tradução automática, geração e compreensão de texto, com treinamento altamente paralelizável, tornando-se a base de modelos modernos como BERT, GPT e ViT.

# Attention Is All You Need



Qual Arquitetura que você utiliza?

*“Eu sou o Gemini, um modelo de linguagem grande (LLM) desenvolvido pelo Google AI. Minha arquitetura fundamental é baseada na revolucionária arquitetura Transformer, uma rede neural que se tornou o padrão de ouro para tarefas de processamento de linguagem natural e que foi, inclusive, introduzida pelo Google em 2017.” Gemini, 2025.*

# Attention Is All You Need

## 1. Tokenização e Representação Numérica (Embeddings)

Primeiro, a pergunta é quebrada em pedaços menores, chamados *tokens*:

```
["Agora", "o", "modelo", "precisa", "descobrir",
"quais", "palavras", "s o", "mais", "importantes",
"para", "entender", "o", "contexto", "de", "cada",
"palavra"]
```

# Attention Is All You Need

Para cada token, o modelo cria três vetores distintos a partir de seu *embedding* inicial:

- **Query (Consulta):** Representa a palavra atual que está buscando informações.  
Pense nisso como a palavra "perguntando" sobre seu contexto.
- **Key (Chave):** Representa a palavra que pode fornecer informação. É como um "rótulo" que descreve o que a palavra significa.
- **Value (Valor):** Contém o significado real ou a informação da palavra.

# Attention Is All You Need

Agora, o modelo precisa descobrir quais palavras são mais importantes para entender o contexto de cada palavra.

# Attention Is All You Need

Agora, o **modelo** precisa **descobrir**  
**quais palavras** são mais importantes para  
**entender** o contexto de cada palavra.

# Attention Is All You Need

Agora, o **modelo**  vai **descobrir**   
**quais palavras** são mais importantes para  
**entender** o contexto de cada palavra.

# Attention Is All You Need

Agora, o **modelo** precisa **descobrir**  
**quais palavras** são mais importantes para  
**entender** o contexto de cada palavra.



# Attention Is All You Need

Self-attention resolvido

Representação  
contextualizada da frase.

descobrir

# Attention Is All You Need

## Decodificação (Geração de Texto)

A resposta é gerada **token por token**:

1. Inicializa-se com um token especial de início.
2. Calcula-se, usando o contexto e os pesos do modelo, **qual o próximo token mais provável**.
3. O token é selecionado e adicionado à resposta.
4. O processo se repete, considerando também os tokens já gerados

## Controle de Estilo

- **Coerência** → cada frase conecta com a anterior.
- **Clareza** → evita ambiguidades.
- **Didática** → uso de listas, exemplos e analogias.

## Finalização

Quando o token de fim é gerado (ou atinge-se um limite de tokens), a resposta é concluída.  
O texto exibido é o resultado dessa sequência **probabilística**, guiada pelo contexto e pelo estilo aprendido.

# Links

## Recursos e Artigos sobre Inteligência Artificial

- **Artigo Fundamental:** [A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY](#)
- **Marco Histórico da IA:** [Demasiado humano: há 20 anos, Kasparov era esmagado por Deep Blue](#)
- **Banco de Dados Clássico:** [MNIST database](#)
- **Repositório de Código:** [GitHub - Redes Neurais](#)
- **Artigo Explicativo:** [Os Tipos de Redes Neurais](#)
- **Artigo da Arquitetura Transformer:** ["Attention Is All You Need"](#)

## Vagas de Emprego (Brasil)

- **Vagas em Dados/Tecnologia:** [McKinsey](#)
- **Vagas em Consultoria/Tecnologia:** [BCG](#)

# ANEXO 1

Token de Foco (Query)	Token Sendo Avaliado (Key)	Score de Atenção Esperado	Justificativa
descobrir	Agora	Baixo	"Agora" é um advérbio de tempo que não modifica diretamente a ação de "descobrir".
descobrir	o	Baixo	Artigo definido com pouca carga semântica para a ação.
descobrir	modelo	Alto	Quem precisa descobrir? O modelo. Há uma forte relação sujeito-verbo. O modelo é o agente da ação de descobrir.
descobrir	precisa	Médio	"Precisa" é o verbo auxiliar que modifica "descobrir". Existe uma ligação gramatical, mas "modelo" (o agente) e "palavras" (o objeto) são semanticamente mais importantes.
descobrir	descobrir	Médio	A palavra sempre tem alguma atenção a si mesma.
descobrir	quais	Alto	Descobrir o quê? "quais palavras...". O pronome "quais" inicia o objeto direto do verbo "descobrir", criando uma forte conexão sobre o que está sendo descoberto.

descobrir	palavras	Muito Alto	Descobrir o quê? "quais palavras". Este é o núcleo do objeto direto. A ação de "descobrir" está diretamente aplicada sobre as "palavras". Este provavelmente seria um dos scores mais altos.
descobrir	são	Baixo	Verbo de ligação dentro da oração subordinada, sem conexão direta com "descobrir".
descobrir	importantes	Médio-Alto	"Importantes" qualifica "palavras". Como "palavras" é crucial para "descobrir", seu qualificador também ganha importância. Descobrir (quais palavras) são importantes.
descobrir	para	Médio	A preposição "para" introduz a finalidade da ação de descobrir.
descobrir	entender	Alto	Descobrir para quê? "para entender". O verbo "entender" representa o propósito da ação de "descobrir", criando uma forte ligação de finalidade.
descobrir	o	Baixo	Artigo.
descobrir	contexto	Médio-Alto	Parte do propósito de "descobrir". "Entender o contexto". Como está ligado a "entender", também se liga indiretamente a "descobrir".
descobrir	de	Baixo	Preposição.
descobrir	cada	Baixo	Adjetivo que modifica "palavra" na oração final.
descobrir	palavra	Baixo	Está no final da frase e se refere a um contexto mais distante da ação principal "descobrir".

# **ANEXO 2 - Passo a passo de uma rede neural simples**

## **Rede Neural Simples**

- Entrada: 2 neurônios ( $x_1, x_2$ )
- Camada oculta: 2 neurônios ( $h_1, h_2$ )
- Saída: 1 neurônio ( $y$ )
- Função de ativação: Sigmoid
- Função de perda: MSE

# Passo a passo de uma rede neural simples

## Propagação para frente (Forward Pass)

- **Camada oculta:**
  - $h_1 = \text{sigmoid}(x_1W_1 + x_2W_1 + b_1) \rightarrow 0.525$
  - $h_2 = \text{sigmoid}(x_1W_1 + x_2W_1 + b_1) \rightarrow 0.550$
- **Saída:**
  - $y_{\text{pred}} = \text{sigmoid}(h_1W_2 + h_2W_2 + b_2) \rightarrow 0.644$

**Objetivo:** calcular a previsão da rede.

# Passo a passo de uma rede neural simples

## Cálculo do Erro

### Erro (Loss)

- Fórmula:  $0.5 * (y_{\text{pred}} - y_{\text{true}})^2$
- Exemplo:  $L \approx 0.065$
- **Objetivo:** medir diferença entre previsão e valor real.

# Passo a passo de uma rede neural simples

## Backpropagation

### Retropropagação do erro

- **Erro na saída:**  $\delta_{\text{out}} = (y_{\text{pred}} - y_{\text{true}}) * y_{\text{pred}} * (1 - y_{\text{pred}}) \approx -0.082$
- **Erro na camada oculta:**
  - $\delta_{h1} \approx -0.010$
  - $\delta_{h2} \approx -0.012$

**Objetivo:** calcular contribuição de cada neurônio para o erro.

# Passo a passo de uma rede neural simples

## Atualização de Pesos

**Regra de atualização:**

- Novo peso = peso antigo - learning\_rate \*  $\delta$  \* entrada
- Exemplo:
  - $W2[0] \approx 0.504$
  - $W2[1] \approx 0.605$
- **Objetivo:** ajustar pesos para reduzir o erro na próxima iteração.

# Passo a passo de uma rede neural simples

## Repetição e Aprendizado

- **Época:** 1 passagem completa pelo exemplo de treino
- Pesos são ajustados a cada época → erro diminui
- Objetivo: rede aprende a mapear entradas → saída corretamente