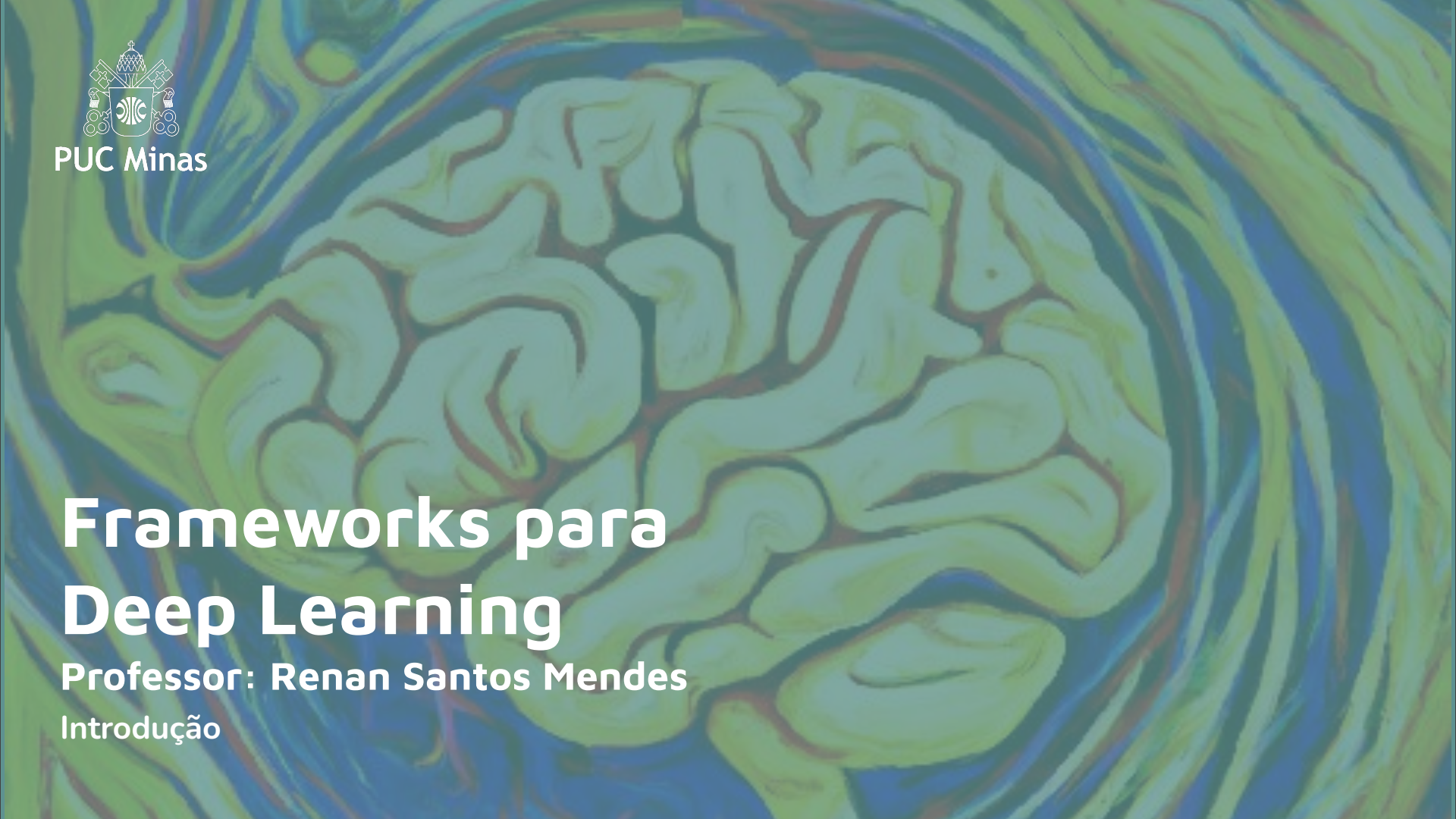


PUC Minas

Frameworks para Deep Learning

Professor: Renan Santos Mendes

Introdução

Índice

Tópicos abordados nessa aula:

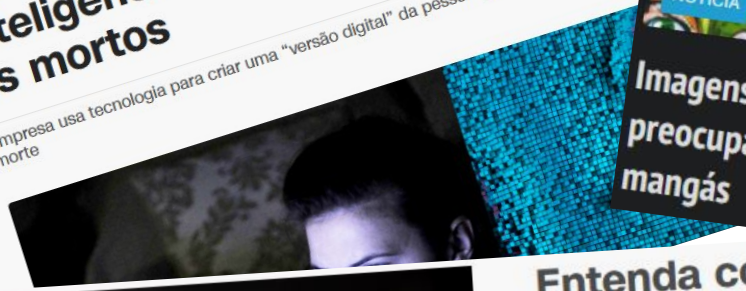
- Introdução
- Conceitos de POO
- Rede Neural Artificial
- Introdução ao Keras + Prática

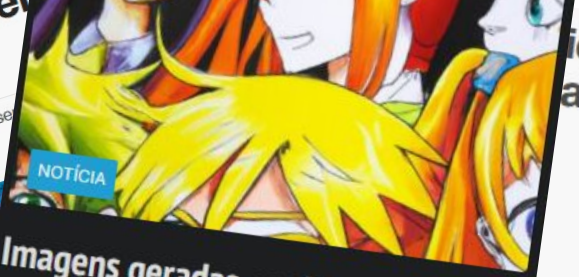
PUC Minas

Introdução

sso é muito “Black Mirror”: A
Inteligência Artificial permite conve
os mortos

Empresa usa tecnologia para criar uma “versão digital” da pessoa que pode viver para se
morte





NOTÍCIA

Imagens geradas por IA já preocupam até artistas de animes e mangás

icrosoft planeja retorno do Clippy
as agora com inteligência artificia

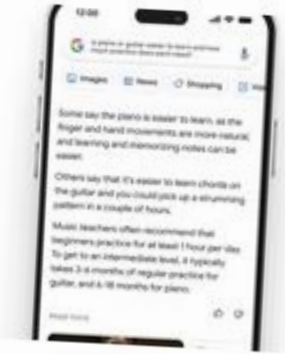
29/01/2023 às 04:00

BUSINESS

Entenda como o longo período de domínio online do Google pode terminar

28/01/2023 às 14:00

BUS



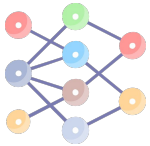
Bard é nova plataforma de IA do Google para rivalizar com ChatGPT

Corretores de imóveis nos EUA: Nã
imaginamos como trabalhar sem o
ChatGPT agora

28/01/2023 às 20:51

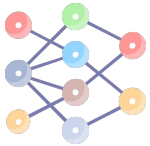
Frameworks

Frameworks



- Frameworks de deep learning **são bibliotecas de software para criação, treinamento e implantação de modelos** de redes neurais profundas em tarefas de aprendizado de máquina.
- Eles fornecem uma interface para definir, configurar e treinar modelos de deep learning, bem como realizar outras tarefas comuns.
- Os frameworks **oferecem implementações eficientes** de algoritmos de backpropagation e outras técnicas de otimização, o que ajuda a acelerar o processo de treinamento.
- Alguns exemplos de frameworks de deep learning populares incluem TensorFlow, PyTorch, Keras, Caffe, Theano, MXNet, entre outros.

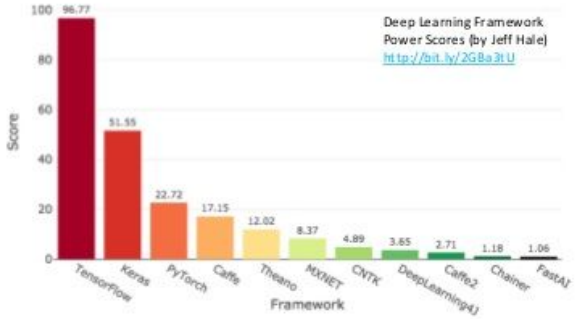
Frameworks



- Cada framework tem suas próprias características, pontos fortes e fracos, mas todos eles compartilham o objetivo comum de **tornar mais fácil e acessível a criação de modelos de deep learning** poderosos e eficientes.
- Os frameworks de deep learning são usados em diversas aplicações, como **visão computacional**, **NLP**, **speech**, entre outras áreas de **inteligência artificial**.
- Eles permitem que pesquisadores e engenheiros criem modelos de deep learning mais **rapidamente** e com **menos esforço**, permitindo que eles foquem em tarefas mais desafiadoras e criativas.

Frameworks

Deep Learning Frameworks



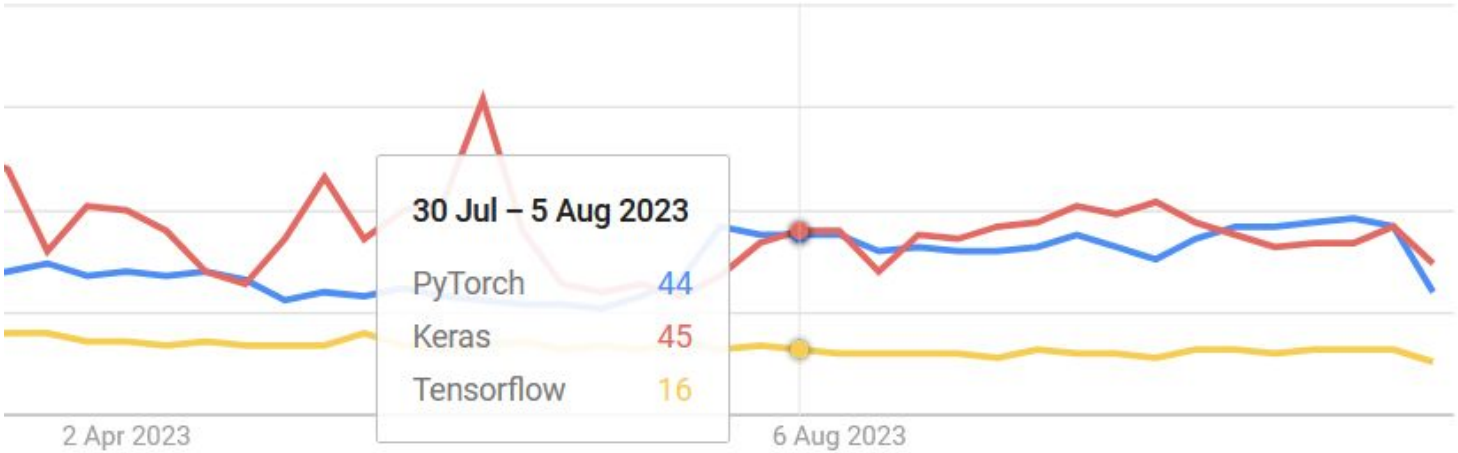
Factors to consider:

- Learning curve
- Speed of development
- Size and passion of community
- Number of papers implemented in framework
- Likelihood of long-term growth and stability
- Ecosystem of tooling

1. TensorFlow
2. Keras
3. PyTorch
4. Caffe
5. theano
6. mxnet
7. CNTK
8. DL4J
9. Caffe2
10. Chainer
11. fast.ai

PUC Minas

Frameworks

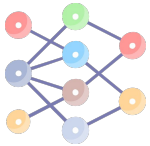


PUC Minas

Antes, precisamos de aprender alguns conceitos importantes de programação

Programação Orientada a Objetos (POO)

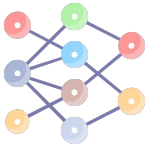
POO - Introdução



- **Programação Orientada a Objetos** é um **paradigma de programação** baseado no conceito de **objetos**, que representam **entidades do mundo real**.
- Os objetos possuem **propriedades** (atributos) e **comportamentos** (métodos) específicos.
- O programa é organizado em **classes**, que são **modelos para a criação de objeto**s, e **objetos**, que são **instâncias de uma classe**.
- Os objetos interagem entre si por meio de mensagens, que são enviadas de um objeto para outro.
- Cada objeto possui seu próprio estado e comportamento, e pode se comunicar com outros objetos para realizar ações mais complexas.

PUC Minas

POO - Introdução

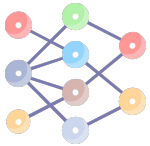


- A POO aumenta a **eficiência**, **modularidade**, **flexibilidade** e **reutilização de código** em programas mais complexos.
- Conceitos como **herança**, **encapsulamento**, **polimorfismo** e **abstração** permitem criar hierarquias de classes, esconder informações internas de uma classe, criar objetos com comportamentos diferentes a partir de uma mesma classe e simplificar o código ao definir interfaces mais abstratas.
- A POO é utilizada em diversas linguagens de programação, como Python, Java, C++, Ruby, entre outras.
- É especialmente útil em projetos de grande escala, onde a organização e reutilização do código são essenciais.

PUC Minas

Conceitos Fundamentais

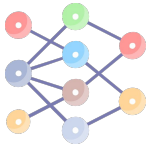
POO - Conceitos



- **Classes e objetos**: **classes** são estruturas que definem as características e comportamentos dos objetos. Os **objetos**, por sua vez, são instâncias de uma classe, ou seja, são criados a partir de uma classe.
- **Atributos e métodos**: **atributos** são as características dos objetos, como nome, idade, cor, entre outros. Já os **métodos** são as ações que os objetos podem realizar, como andar, falar, calcular, entre outros.

PUC Minas

POO - Conceitos



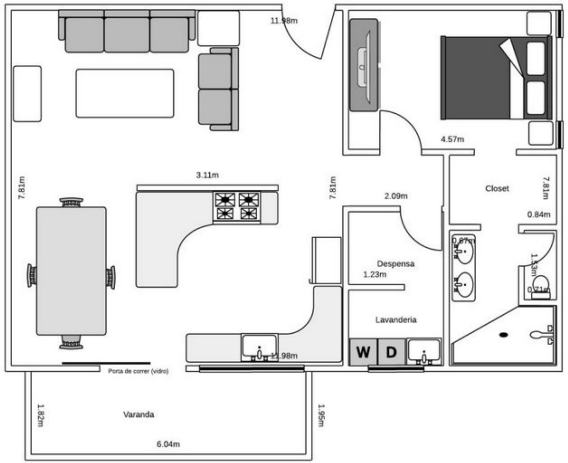
- **Herança:** é o processo de **criar uma nova classe** a **partir de uma classe existente**, mantendo suas características e comportamentos. A nova classe (chamada de classe filha ou subclasse) pode adicionar novos atributos e métodos ou sobrescrever os existentes.
- **Encapsulamento**: é o princípio de **esconder a implementação** de um objeto de outros objetos. É possível tornar certos atributos e **métodos privados**, para que só possam ser acessados e modificados pela própria classe, garantindo maior segurança e controle de acesso às informações.

PUC Minas

Diferenciando classes de objetos

POO - Conceitos

Classe



Planta → Projeto

Criação de
Uma Instância

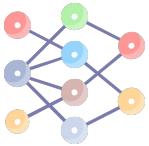
Objeto

Casa → Instância

PUC Minas

Como fazemos isso em Python?

POO - Implementação

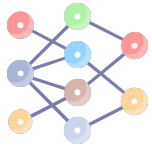


```python
class Carro(object):
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def acelerar(self):
        print(f'{self.marca} {self.modelo} está acelerando...')
```

PUC Minas

POO - Implementação



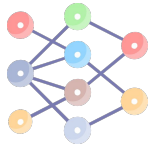
Criação da Classe

```python
class Carro(object):
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def acelerar(self):
        print(f'{self.marca} {self.modelo} está acelerando...')
```

PUC Minas

POO - Implementação



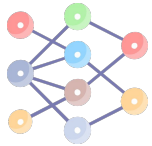
Criação do Construtor

```
class Carro(object):
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def acelerar(self):
        print(f'{self.marca} {self.modelo} está acelerando...')
```

PUC Minas

POO - Implementação



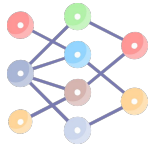
Uso da
Palavra “self”

```
class Carro(object):
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def acelerar(self):
        print(f'{self.marca} {self.modelo} está acelerando...')
```

PUC Minas

POO - Implementação



Criação dos Atributos

```
class Carro(object):
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def acelerar(self):
        print(f'{self.marca} {self.modelo} está acelerando...')
```

PUC Minas

POO - Implementação

```python
class Carro(object):
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def acelerar(self):
        print(f'{self.marca} {self.modelo} está acelerando...')
```

Criação dos Métodos

PUC Minas

POO - Implementação

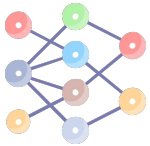
```python
class Carro(object):
    def __init__(self, marca, modelo, ano):
        self.marca = marca
        self.modelo = modelo
        self.ano = ano

    def acelerar(self):
        print(f'{self.marca} {self.modelo} está acelerando...')
```

Uso dos Atributos
dentro dos métodos

PUC Minas

POO - Implementação



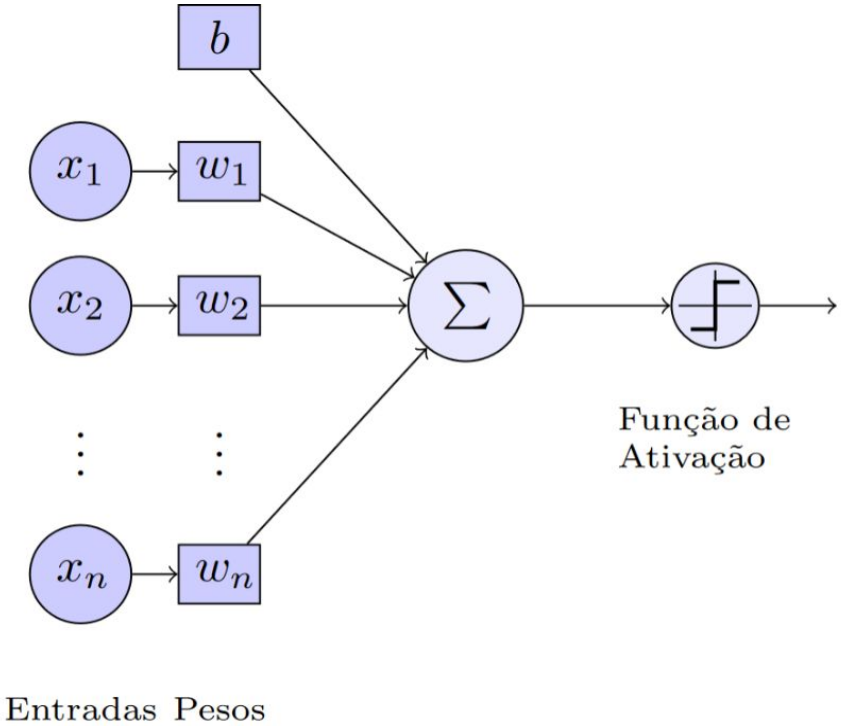
- Uma classe é definida pela **palavra “class”** e em seguida seu nome;
- Usa-se os parênteses para indicar qual a classe mãe/pai;
- O método **__init__** é chamado de construtor;
- A palavra reservada **self** é usada para **acessar os atributos** e **métodos do objeto**;
- Para se criar um método, deve-se utilizar a palavra **def** (igual criamos uma função) e utilizar a palavra “**self**” como primeiro argumento;
- Para acessar um atributo dentro de um método, usa-se a palavra **self + nome do atributo**, por exemplo: self.nome_do_atributo

Neurônio Artificial

Recapitulando

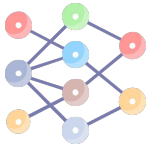
Neurônio Artificial

Neurônio Artificial



PUC Minas

Neurônio Artificial

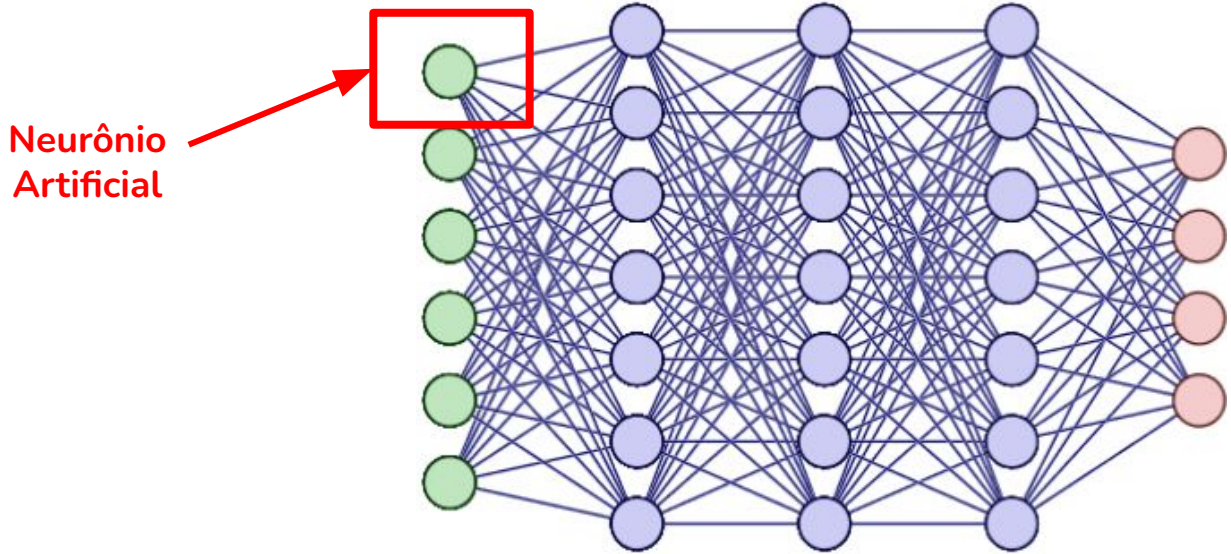


Elementos básicos do neurônio artificial:

1. **Conjunto de sinapses** em que cada uma é caracterizada por um peso (ou força) entre os neurônios com valores podendo ser positivos ou negativos;
2. **Somador** para agregar os sinais que chegam ao neurônio, ponderados pelos respectivos pesos (combinação linear);
3. **Função de ativação** que limita a amplitude da saída de um neurônio.

Rede Neural Artificial

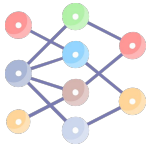
Rede Neural Artificial



PUC Minas

Nomenclatura

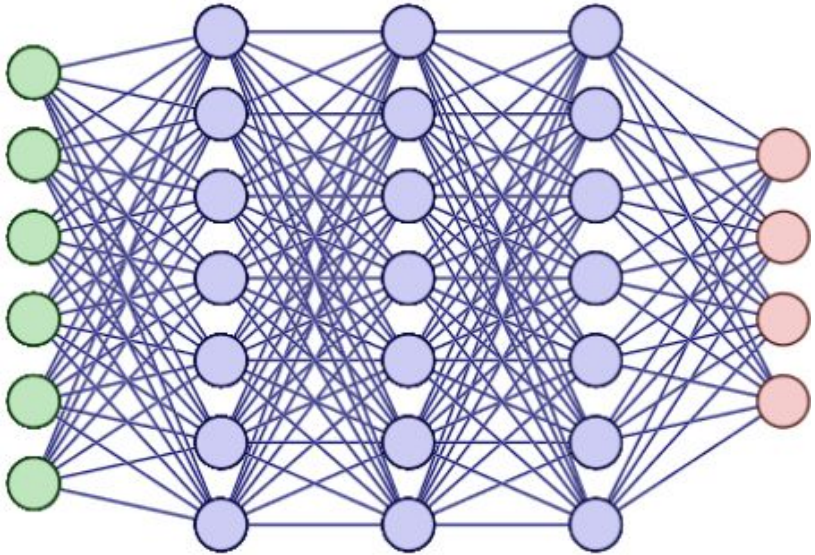
Nomenclatura



- **Camada de entrada**: camada pela qual os dados entram na rede;
- **Camada oculta:** uma ou mais camadas intermediárias;
- **Camada de saída**: camada por onde o processamento feito pela rede é obtido;
- **Conexões:** ligações entre os neurônios (pesos da rede).

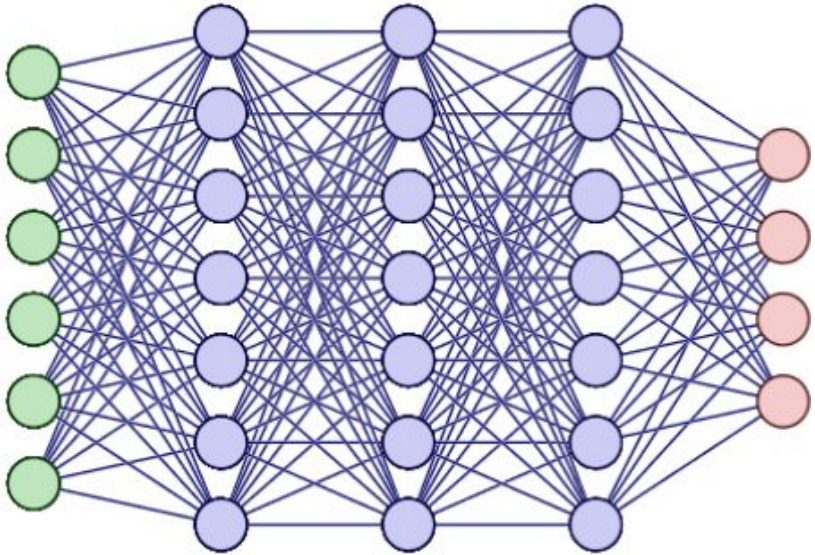
PUC Minas

Nomenclatura em uma Rede Neural



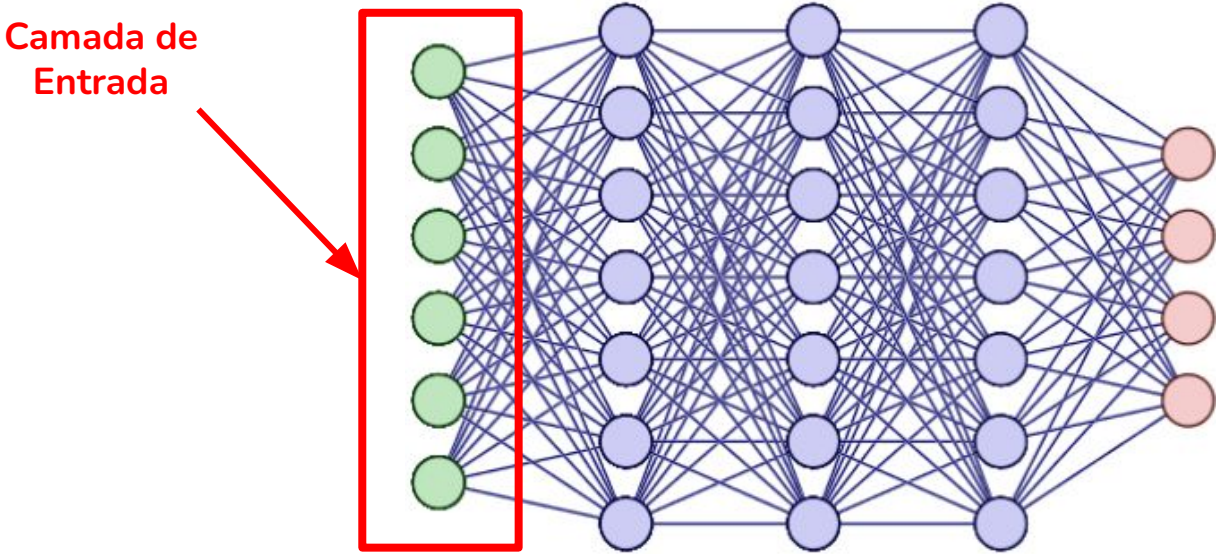
PUC Minas

Nomenclatura em uma Rede Neural

Dados →  → Saída

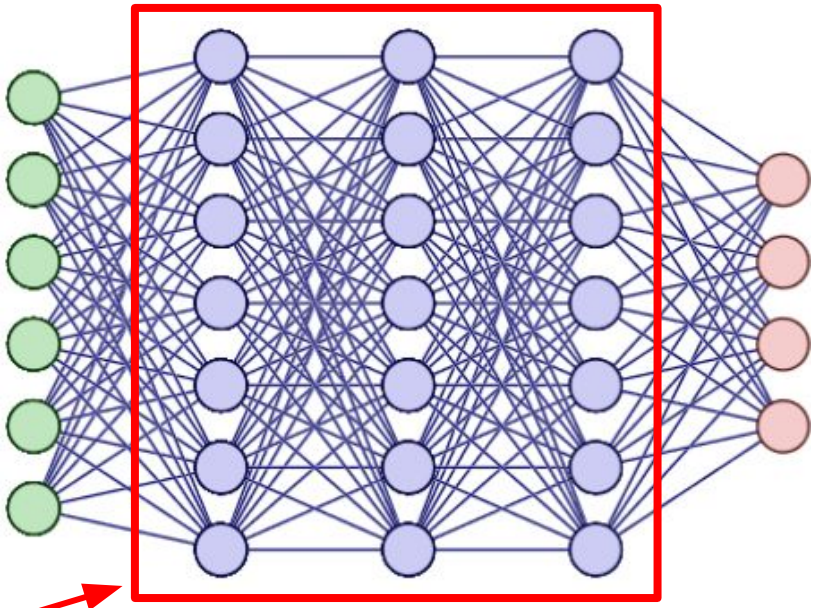
PUC Minas

Nomenclatura em uma Rede Neural



PUC Minas

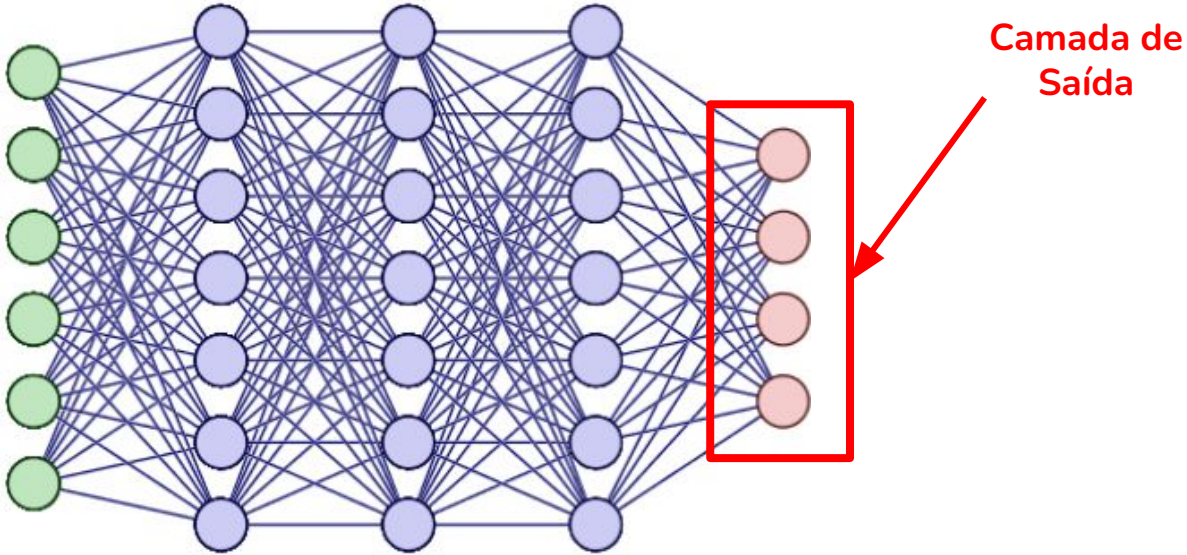
Nomenclatura em uma Rede Neural



Camadas Ocultas

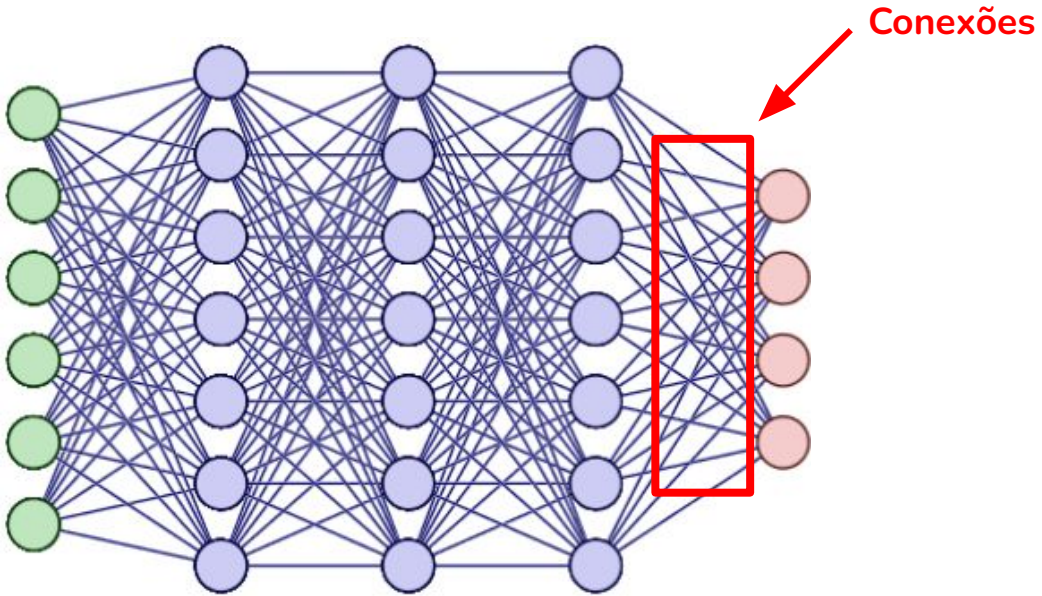
PUC Minas

Nomenclatura em uma Rede Neural



PUC Minas

Nomenclatura em uma Rede Neural



PUC Minas

Introdução ao Keras

Continuando

Introdução ao Keras

Introdução ao Keras

Características do framework:

- **Fácil prototipação** e criação de modelos para produção;
- Extensa documentação online com diversos guias e exemplos;
- Necessita **pouco código** - alto nível;
- **Simplifica o treino** do modelo;
- Classes podem ser herdadas e o treino customizado + callbacks;
- Uso da infra dos recurso do TensorFlow 2, sendo incorporado ao TF;

PUC Minas

Introdução ao Keras

Quem usa?

- CERN (LHC);
- NASA;
- NIH;
- Netflix;
- Uber.

PUC Minas

Introdução ao Keras

Para fazer a instalação local:

- conda create --name myenv python=3.9
- conda activate myenv
- pip install "tensorflow<2.11"

PUC Minas

Estrutura de Dados nos Frameworks

Introdução ao Keras

Um conceito muito importante para todos os frameworks de DL:

- **Tensor**: é uma estrutura de armazenamento de dados multidimensional;
- Tensores são usados para **facilitar as operações matemáticas** (multiplicações matriciais) em uma rede neural;
- O formato mais conhecido é uma **matriz** (tensor com duas dimensões);
- Para o Keras, são usados os ndarrays do Numpy (diferentemente do PyTorch);
- Manipulação do dado da mesma forma;
- Origem da área de exatas: **física e engenharia**.

PUC Minas

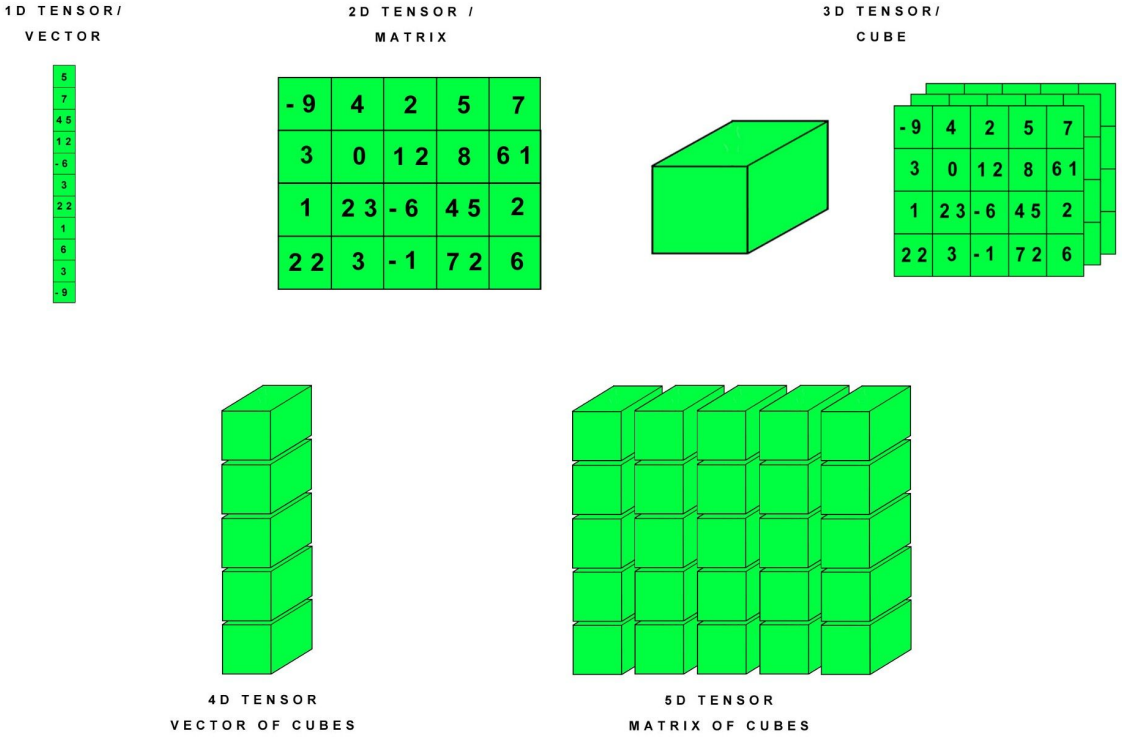
Introdução ao Keras

Algumas aplicações dos tensores :

- **0D:** escalares
- **1D**: vetores
- **2D**: matrizes
- **3D**: timeseries ou dados sequenciais
- **4D**: imagens (batch, altura, largura e canal)
- **5D**: vídeo (sample, frame, altura, largura e canal)

PUC Minas

Introdução ao Keras



PUC Minas

Introdução ao Keras

Módulos que utilizaremos durante a disciplina:

- **datasets**: fornece alguns conjuntos de dados simples (em Numpy) que podem ser usados para testar um modelo ou criar exemplos (MNIST, CIFAR10, CIFAR100,...);
- **models**: fornece o tipo de modelo que pode ser usado, sequencial, funcional ou implementar do zero;
- **layers**: As camadas são os blocos de construção básicos no Keras. Uma camada consiste em uma função que processa um tensor de entrada em um de saída e algum estado, mantido em variáveis do TensorFlow (os pesos da camada);
- **utils**: fornece funções e métodos extras;

Componentes dos Módulos

Introdução ao Keras

- **Models**:
  - **Sequencial**: modelo que adiciona camadas sequencialmente;
  - **Funcional**: modelo com maior customização;
- **Layers**:
  - **Dense**: camadas totalmente conectadas;
  - **InputLayer**: camada de entrada na rede;
  - **Conv2D**: camada convolucional em 2D;
  - **Flatten**: camada para “achatamento”;
  - **RNN**: camada recorrente;
  - **LSTM**: camada com células LSTM;
  - **GRU**: camada com células GRU;

PUC Minas

Exemplo de um modelo com Keras

Introdução ao Keras

Um modelo no Keras ficaria da seguinte forma:

```
model = Sequential()
model.add(Dense(100, input_shape=(21, ), activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(300, activation='relu'))
model.add(Dense(200, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              metrics=['accuracy'])

hist = model.fit(X_train,
          y_train,
          epochs=25,
          validation_split=0.2)
```

PUC Minas

Introdução ao Keras

Para criar e treinar um modelo de DL no Keras, é necessário seguir alguns passos básicos:

1. Importar bibliotecas
2. Leitura de dados
3. Processar dados
4. Criar modelo
5. Compilar modelo
6. Treinar modelo

Mais adiante, utilizaremos alguns complementos em alguns desses passos definidos acima.

PUC Minas

Introdução ao Keras

Para criar e treinar um modelo de DL no Keras, é necessário seguir alguns passos básicos:

1. Importar bibliotecas
2. Leitura de dados
3. Processar dados
4. Criar modelo
5. Compilar modelo
6. Treinar modelo

Preparação do ambiente

Mais adiante, utilizaremos alguns complementos em alguns desses passos definidos acima.

PUC Minas

Introdução ao Keras

Para criar e treinar um modelo de DL no Keras, é necessário seguir alguns passos básicos:

1. Importar bibliotecas
2. Leitura de dados
3. Processar dados
4. Criar modelo
5. Compilar modelo
6. Treinar modelo

Preparação dos dados

Mais adiante, utilizaremos alguns complementos em alguns desses passos definidos acima.

PUC Minas

Introdução ao Keras

Para criar e treinar um modelo de DL no Keras, é necessário seguir alguns passos básicos:

1. Importar bibliotecas
2. Leitura de dados
3. Processar dados
4. Criar modelo
5. Compilar modelo
6. Treinar modelo

Preparação do modelo

Mais adiante, utilizaremos alguns complementos em alguns desses passos definidos acima.

PUC Minas

Introdução ao Keras

Para criar e treinar um modelo de DL no Keras, é necessário seguir alguns passos básicos:

1. Importar bibliotecas
2. Leitura de dados
3. Processar dados
4. Criar modelo
5. Compilar modelo
6. Treinar modelo

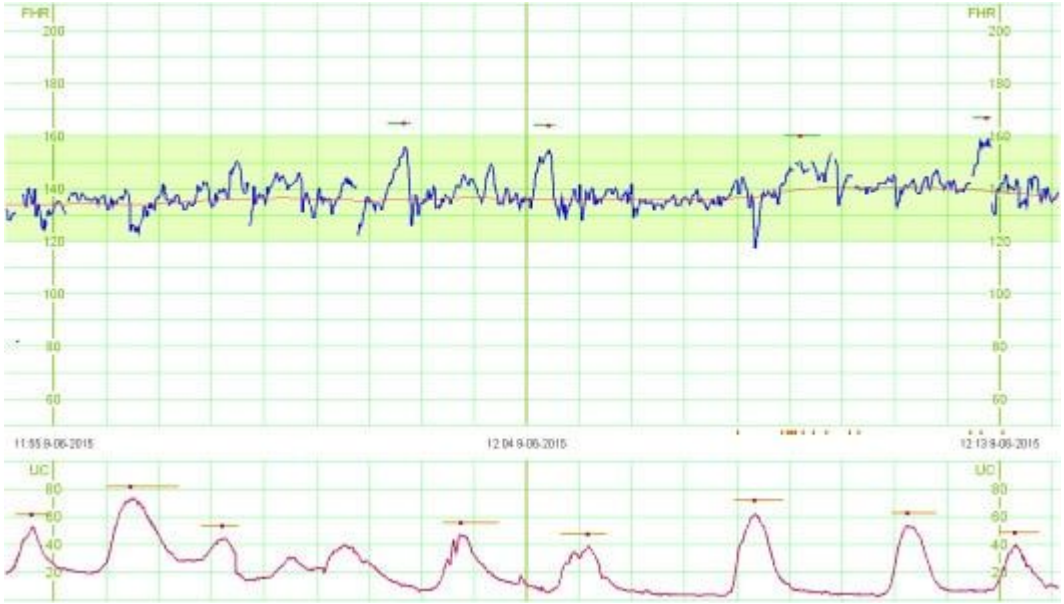
Treino do modelo

Mais adiante, utilizaremos alguns complementos em alguns desses passos definidos acima.

PUC Minas

Prática - Definição do Problema

Cardiotocografia



PUC Minas

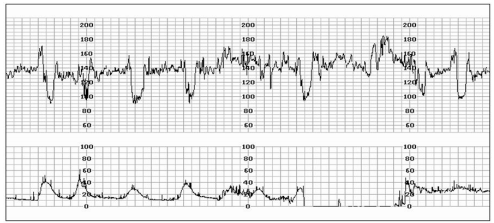
Cardiotocografia

- Cardiotocografias (CTGs) são opções simples e de baixo custo para avaliar a saúde fetal:
- Prevenção da mortalidade infantil e materna;
- Equipamento funciona enviando pulsos de ultrassom e lendo sua resposta;
- Frequência cardíaca fetal (FCF), movimentos fetais, contrações uterinas;
- Dados classificados em 3 classes:
  - Normal;
  - Suspeito;
  - Patológico;

PUC Minas

Cardiotocografia - Workflow

Dado/Exame



Extração de Features

- Frequência
- Movimentos
- Contrações
- Acelerações
- Desacelerações

Modelo

- **Normal**
- **Suspeito**
- **Doente**

PUC Minas

Colab