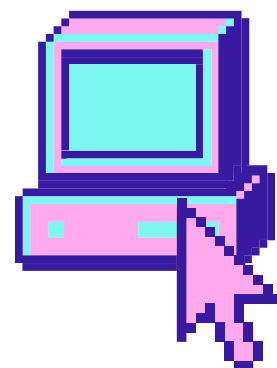
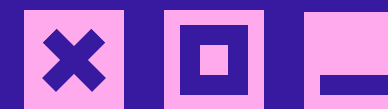


#PYBR2020



REDES NEURAIIS PYTORCH

Camila Laranjeira da Silva



MILA LARANJEIRA

CRIADORA DO PEIXE BABEL

DOUTORANDA EM COMPUTAÇÃO

INSTRUTORA NA ALURA CURSOS ONLINE



U F *m* G





FRAMEWORK DE MACHINE LEARNING

BASEADO NO TORCH, DESENVOLVIDO PELO
FACEBOOK'S AI RESEARCH LAB.

GRATUITO E OPEN SOURCE

DOCUMENTAÇÃO: [HTTPS://PYTORCH.ORG/DOCS/STABLE/INDEX.HTML](https://pytorch.org/docs/stable/index.html)

CÓDIGO FONTE: [HTTPS://GITHUB.COM/PYTORCH/PYTORCH](https://github.com/pytorch/pytorch)

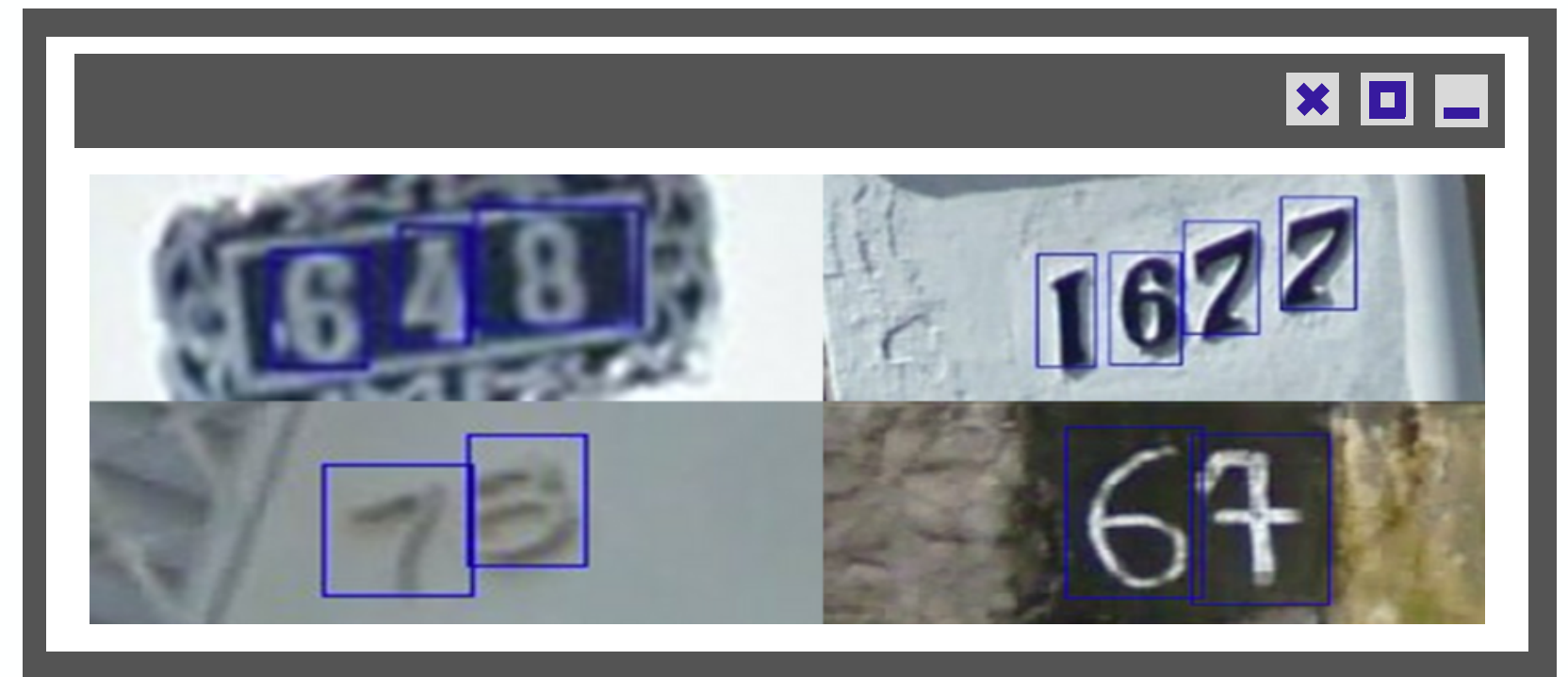
COMUNIDADE ATIVA

STACK OVERFLOW BEM RECHEADO ;)

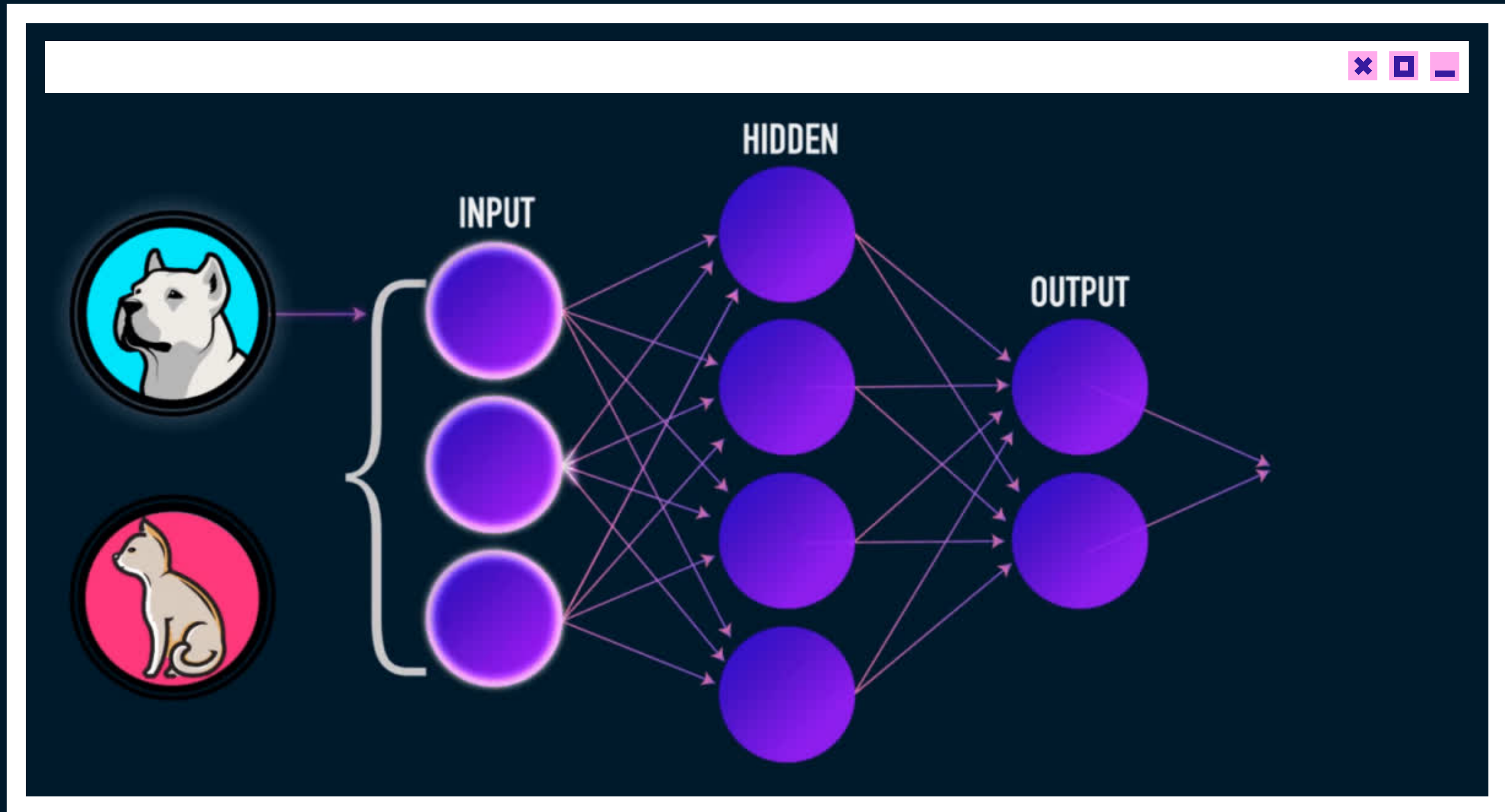
O QUE VAMOS FAZER HOJE?



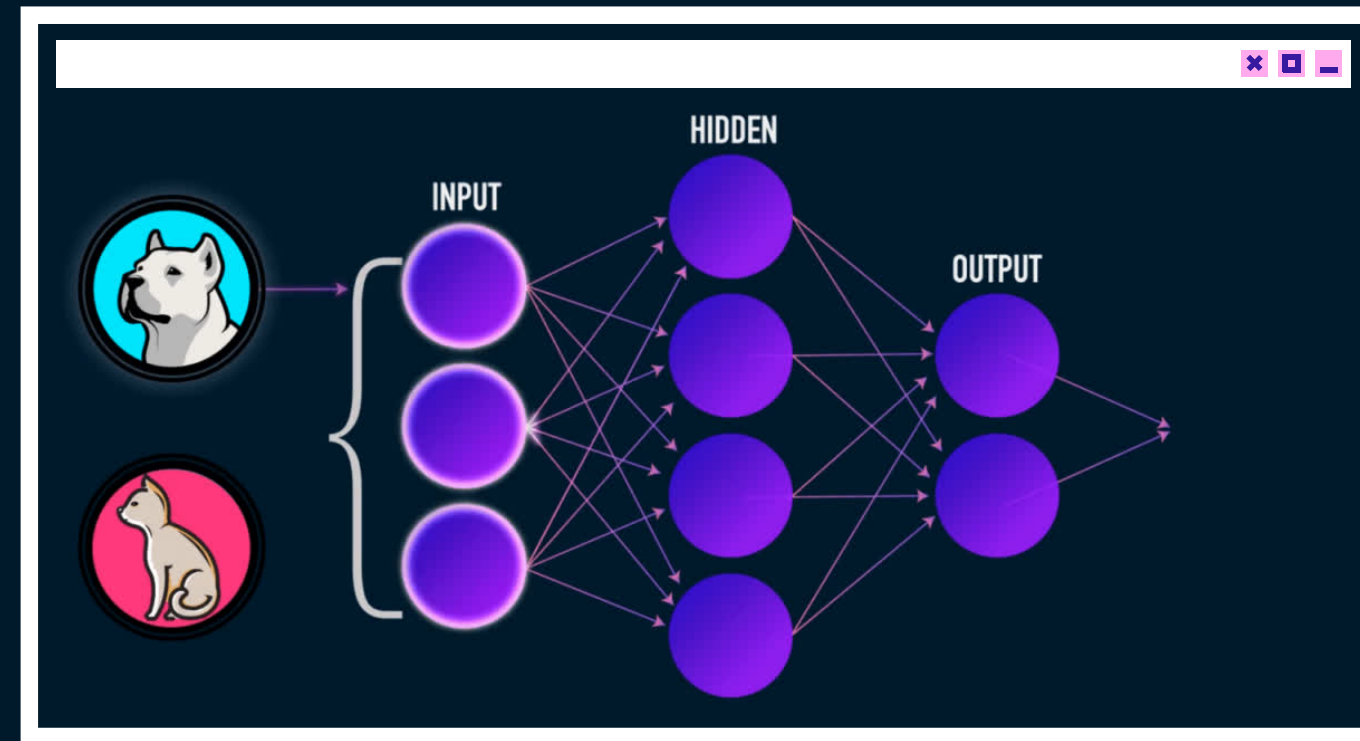
- MNIST: DÍGITOS ESCRITOS A MÃO
- 10 CLASSES: 0-9
- VERSÃO SIMPLIFICADA DE UMA APLICAÇÃO REAL



REDES NEURAIIS



REDES NEURAIIS



1. O QUE É UM TENSOR?

2. CRIANDO UMA REDE NEURAL NO PYTORCH

3. TREINANDO UMA REDE NEURAL NO PYTORCH

REDES NEURAIIS



<https://github.com/peixebabel/RedesNeuraisPyBR>

1. O QUE É UM TENSOR?

2. CRIANDO UMA REDE NEURAL NO PYTORCH

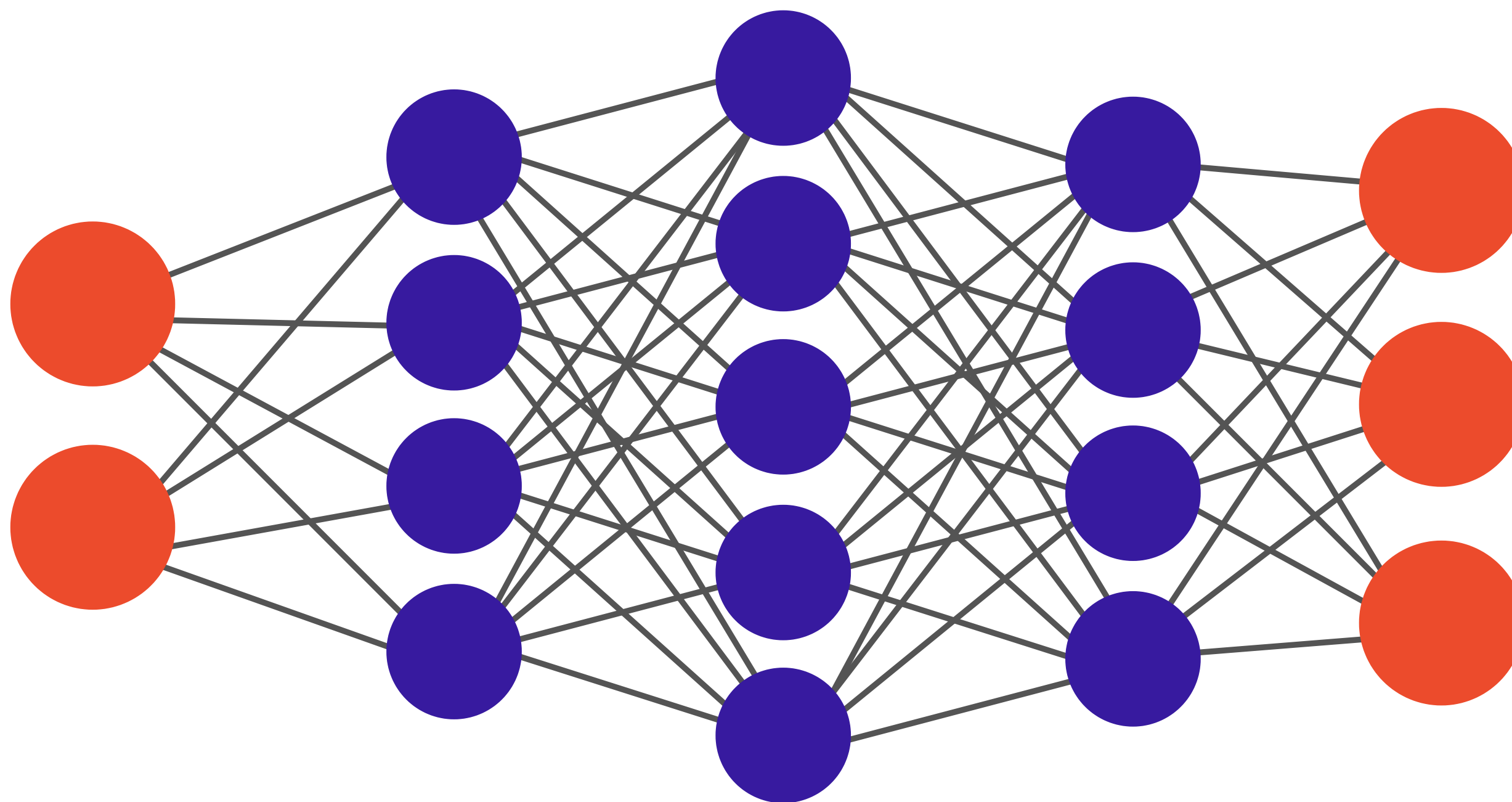
3. TREINANDO UMA REDE NEURAL NO PYTORCH

O QUE É UM TENSOR?



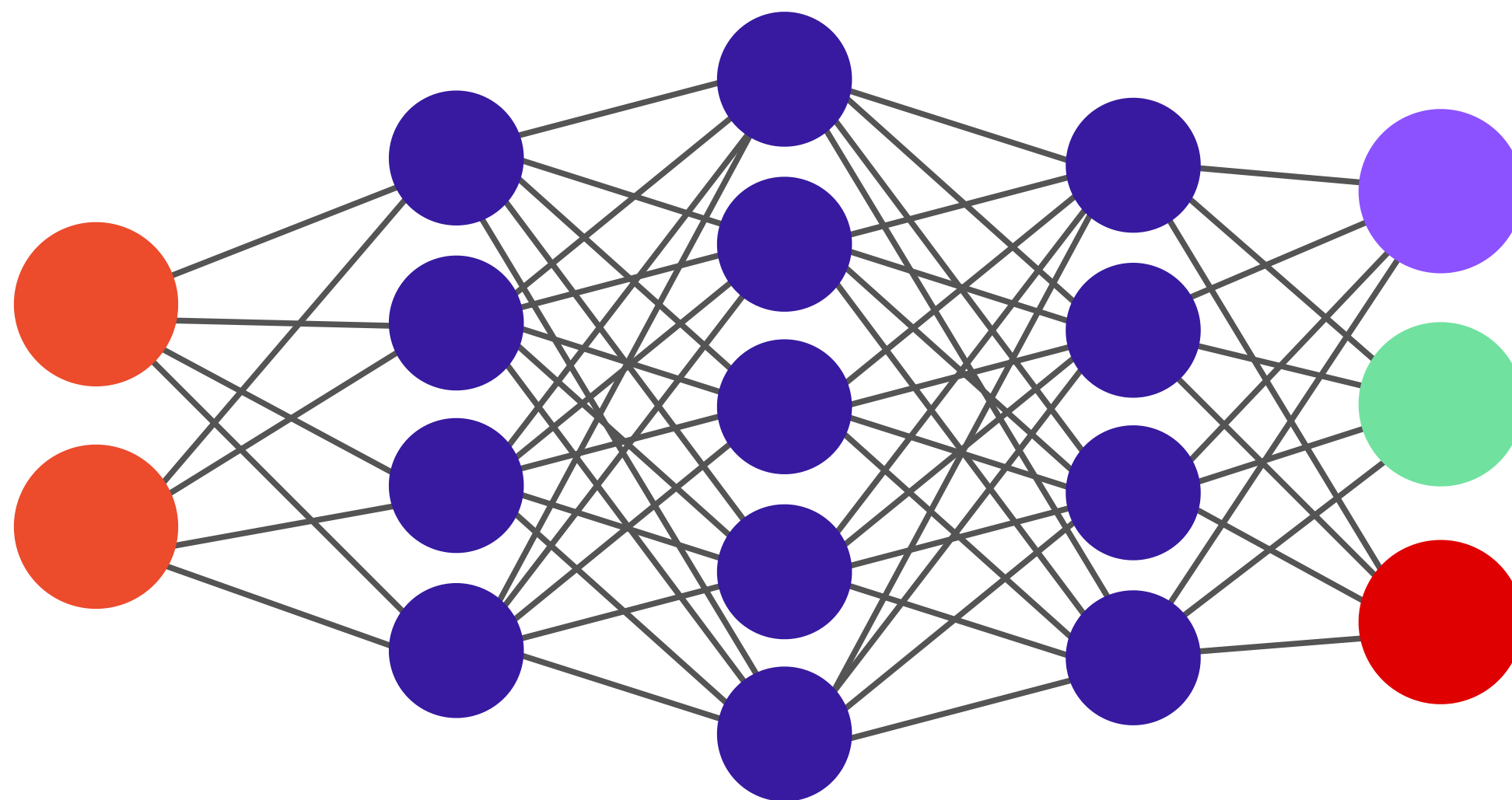
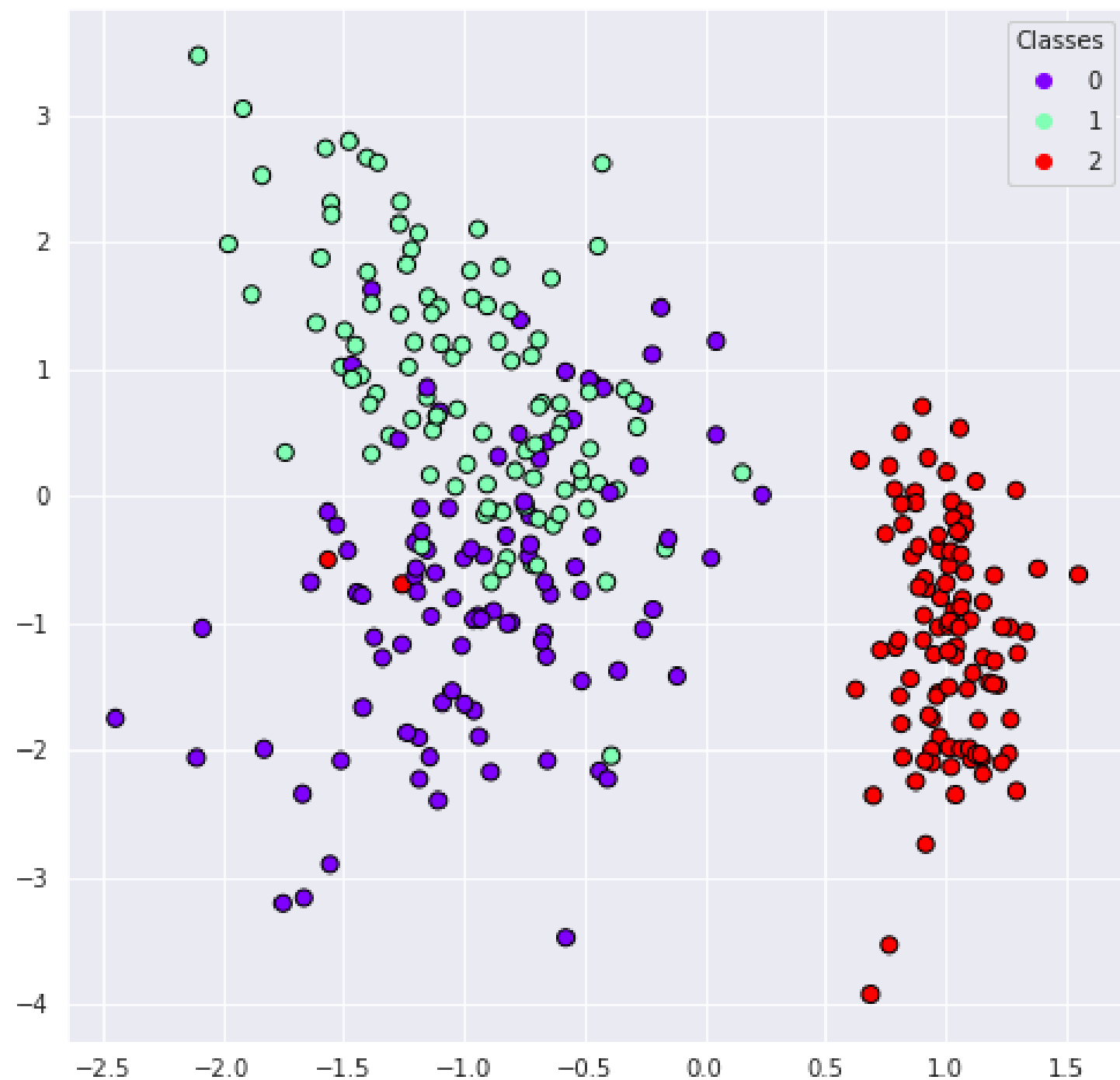
TENSORES.IPYNB

ARQUITETURA DA REDE NEURAL



ARQUITETURA DA REDE NEURAL

ENTRADA COM 2 DIMENSÕES. SAÍDA COM 3 CLASSES.

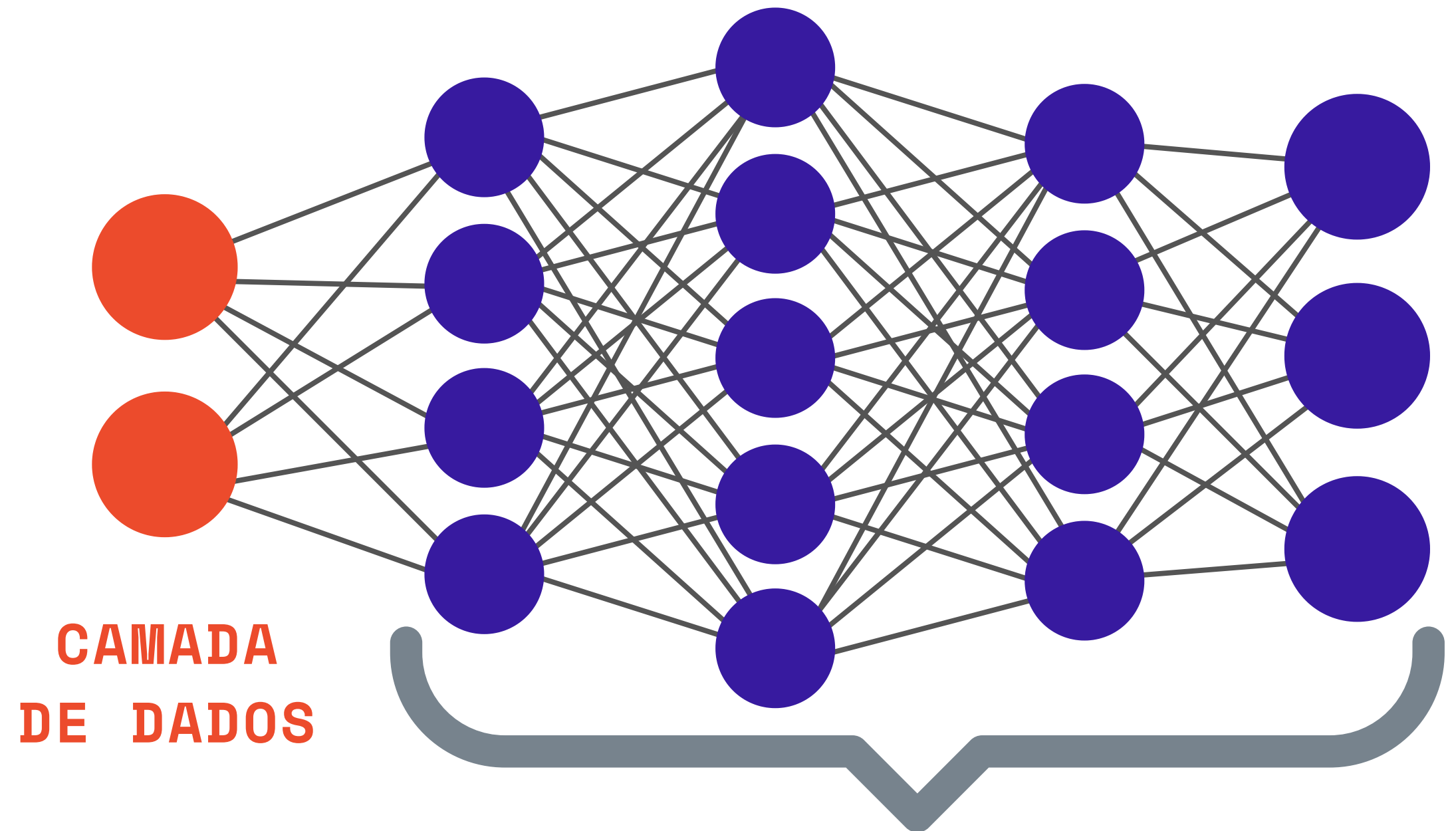


CAMADAS DE NEURÔNIOS

```
nn.Linear(in_size, out_size)
```

PACOTE torch.nn

- Neural Networks

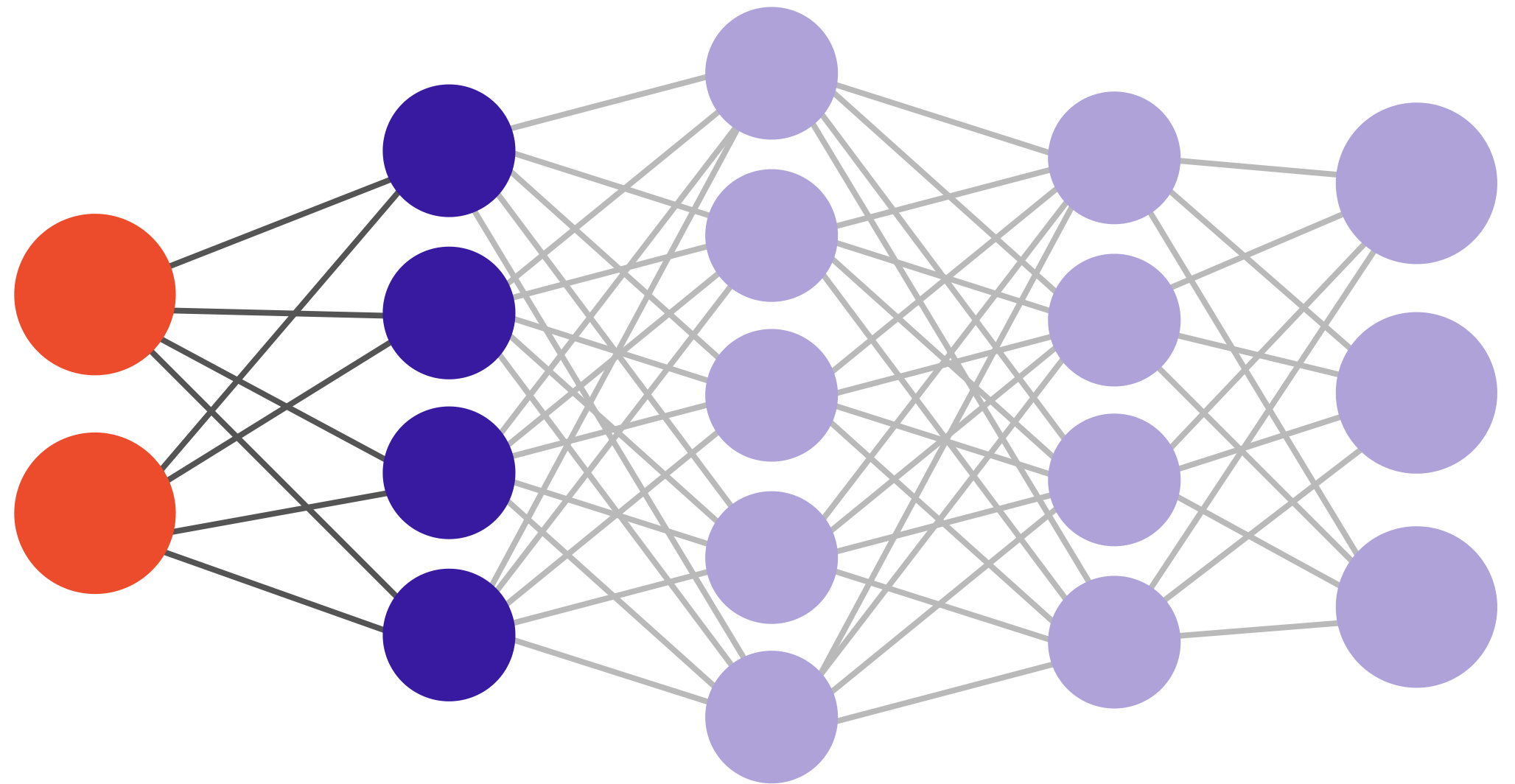


CAMADAS DE NEURÔNIOS

CAMADAS DE NEURÔNIOS

`nn.Linear(in_size, out_size)`

`nn.Linear(2, 4)`

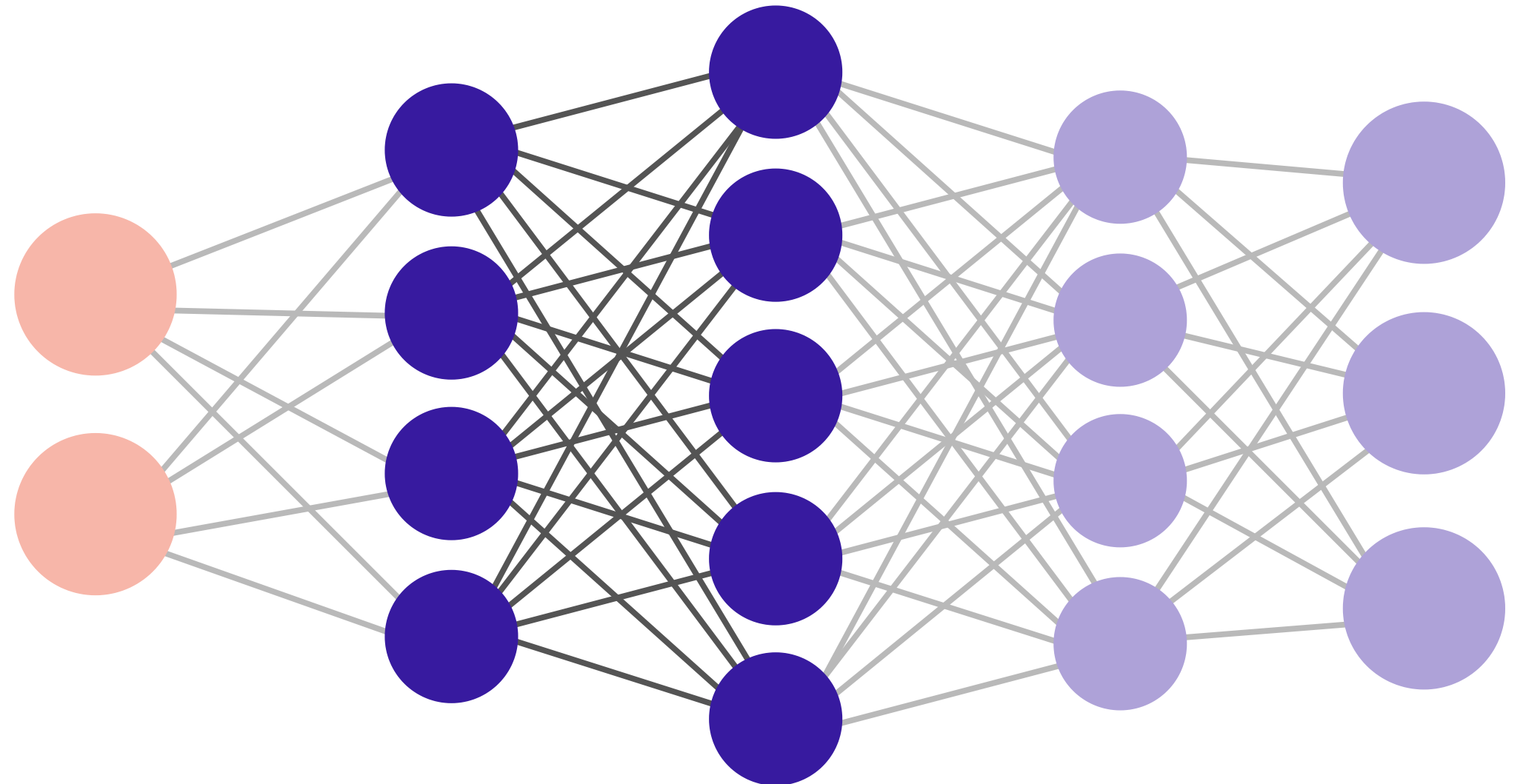


CAMADAS DE NEURÔNIOS

```
nn.Linear(in_size, out_size)
```

```
nn.Linear(2, 4)
```

```
nn.Linear(4, 5)
```



CAMADAS DE NEURÔNIOS

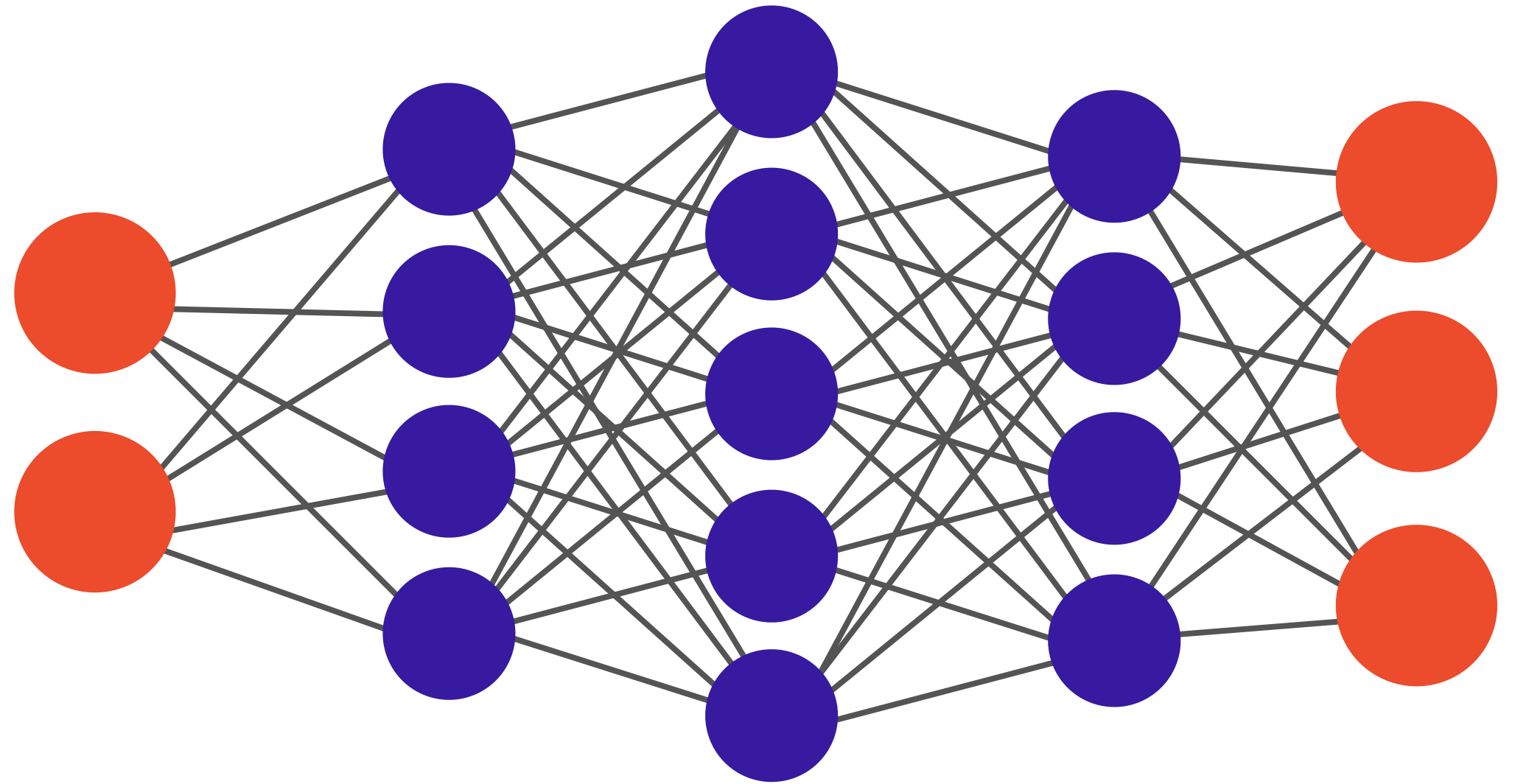
```
nn.Linear(in_size, out_size)
```

```
nn.Linear(2, 4)
```

```
nn.Linear(4, 5)
```

```
nn.Linear(5, 4)
```

```
nn.Linear(4, 3)
```



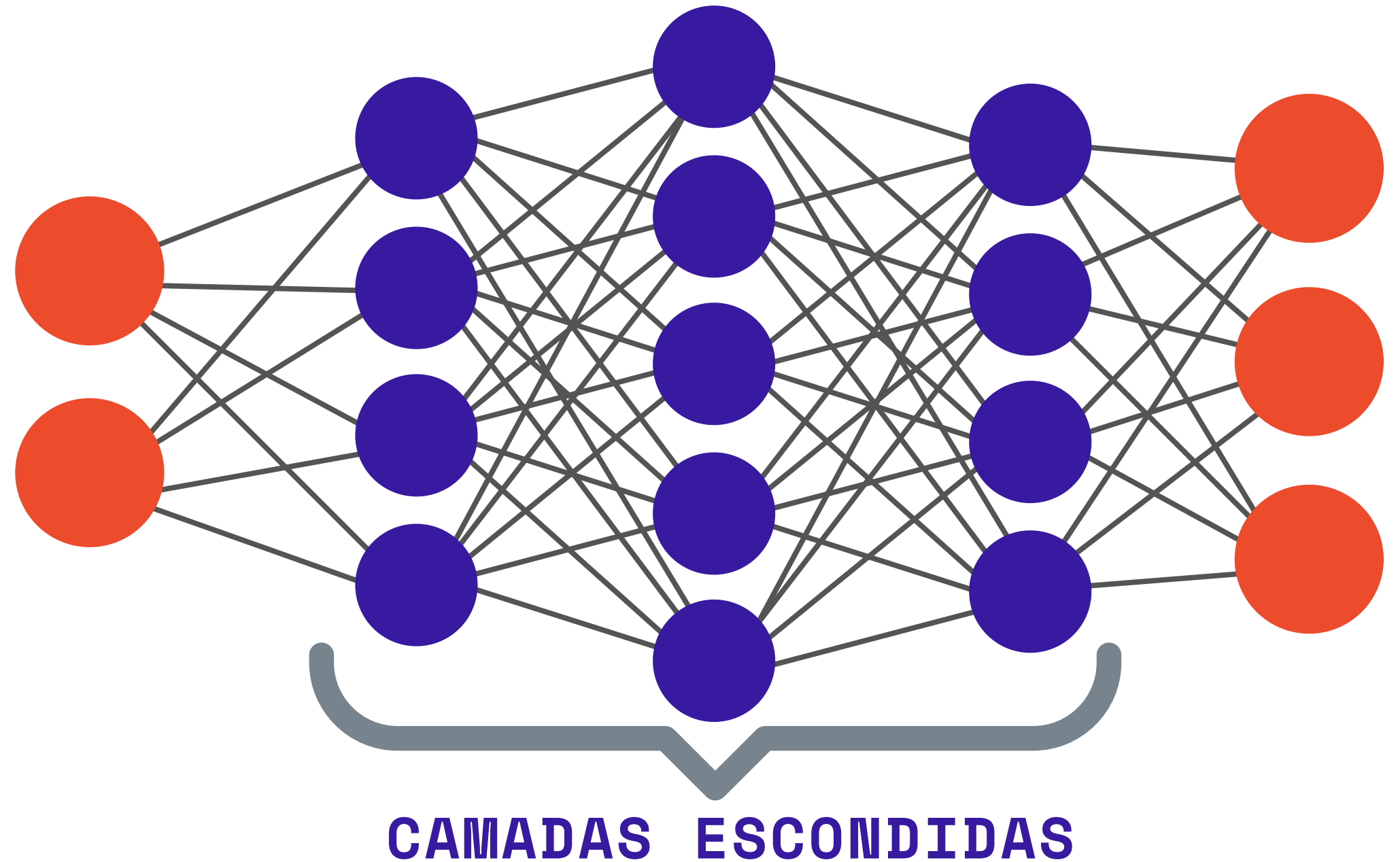
CAMADAS DE NEURÔNIOS

```
nn.Linear(2, 4)
```

```
nn.Linear(4, 5)
```

```
nn.Linear(5, 4)
```

```
nn.Linear(4, 3)
```



PRIMEIRA E ÚLTIMA DIMENSÕES SÃO RELATIVAS AO PROBLEMA.

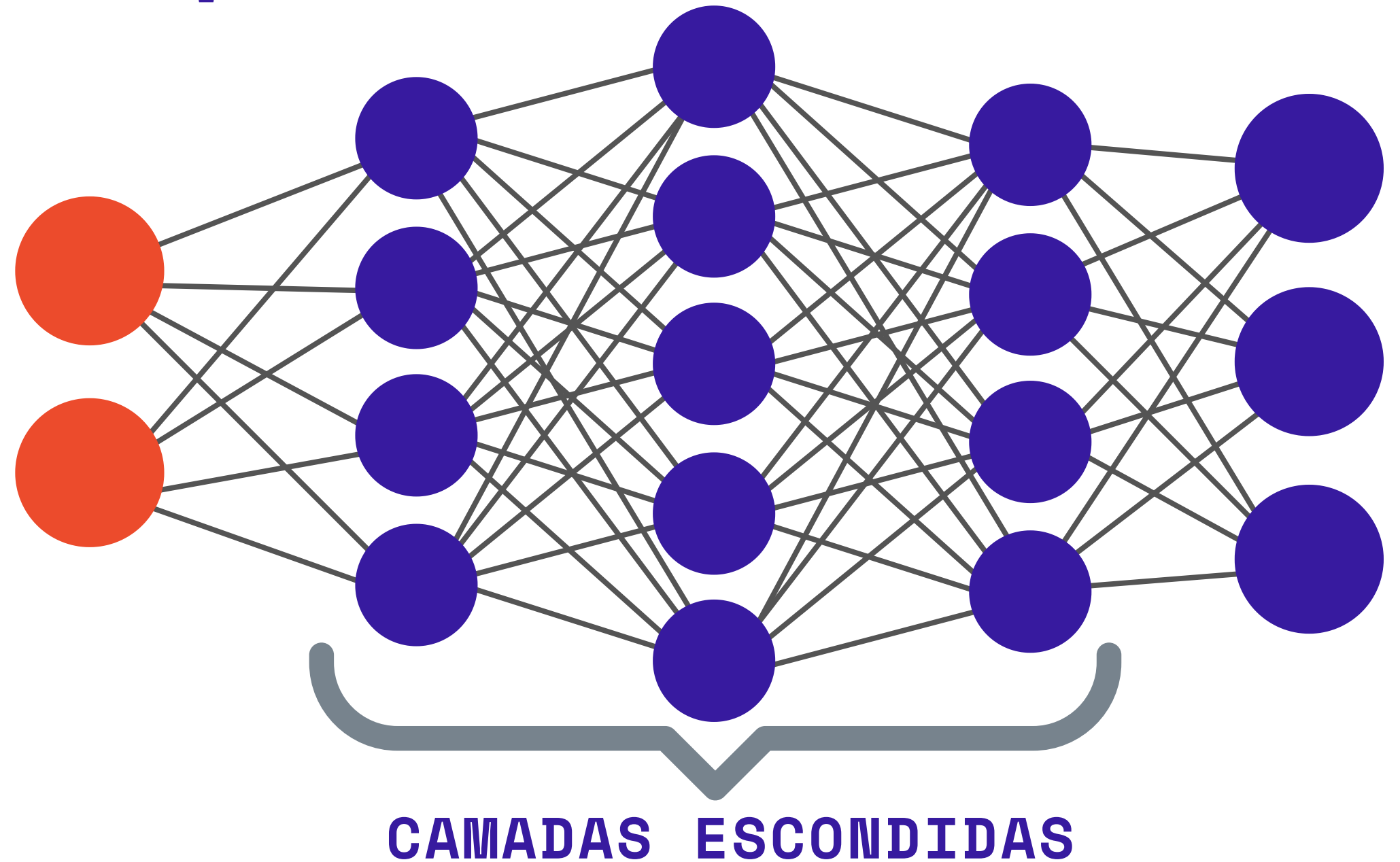
AS INTERMEDIÁRIAS SÃO ESCOLHA DE QUEM PROJETA.

CAMADAS DE ATIVAÇÃO

Camadas escondidas devem ser seguidas por uma camada de ativação.

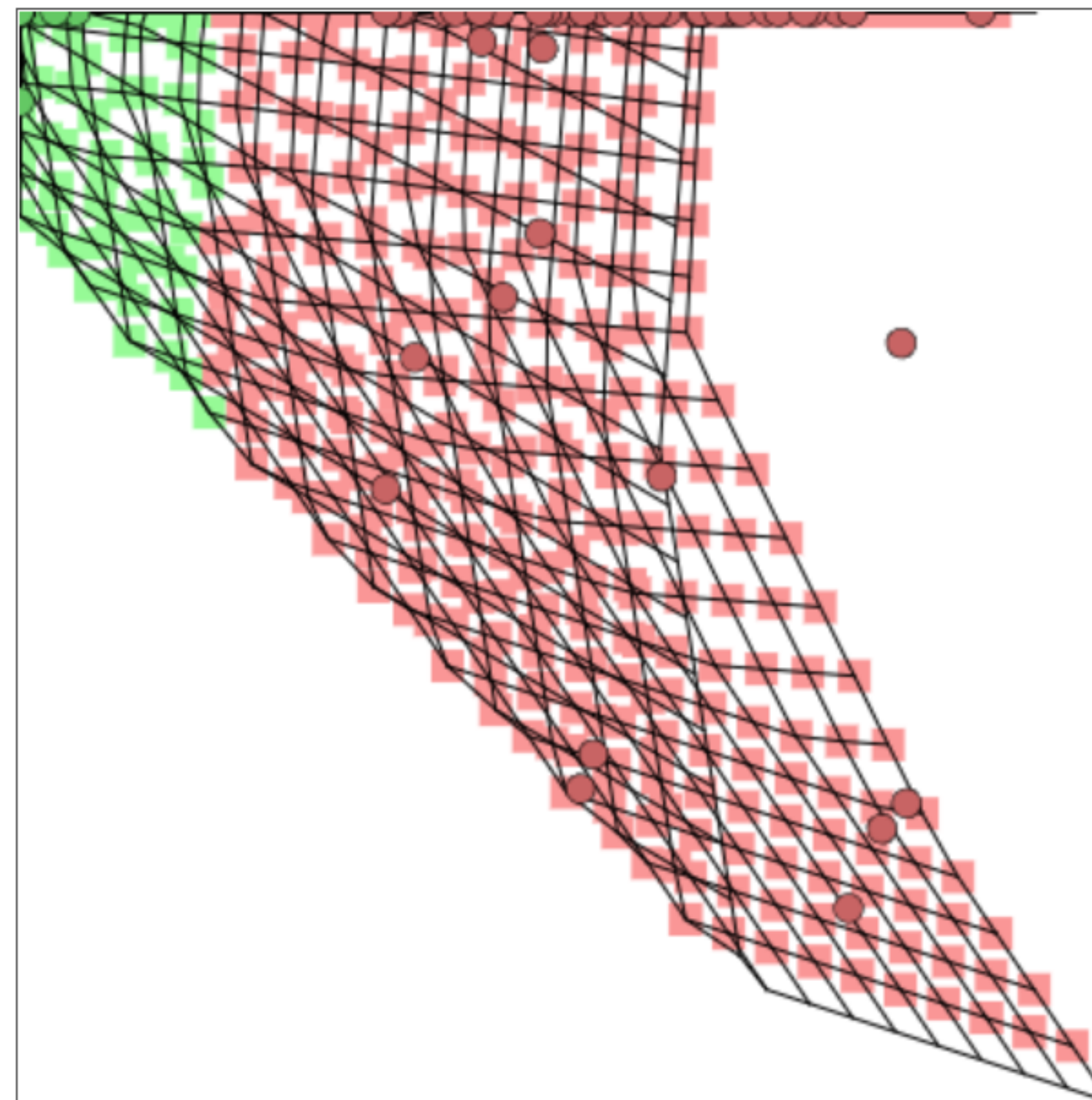
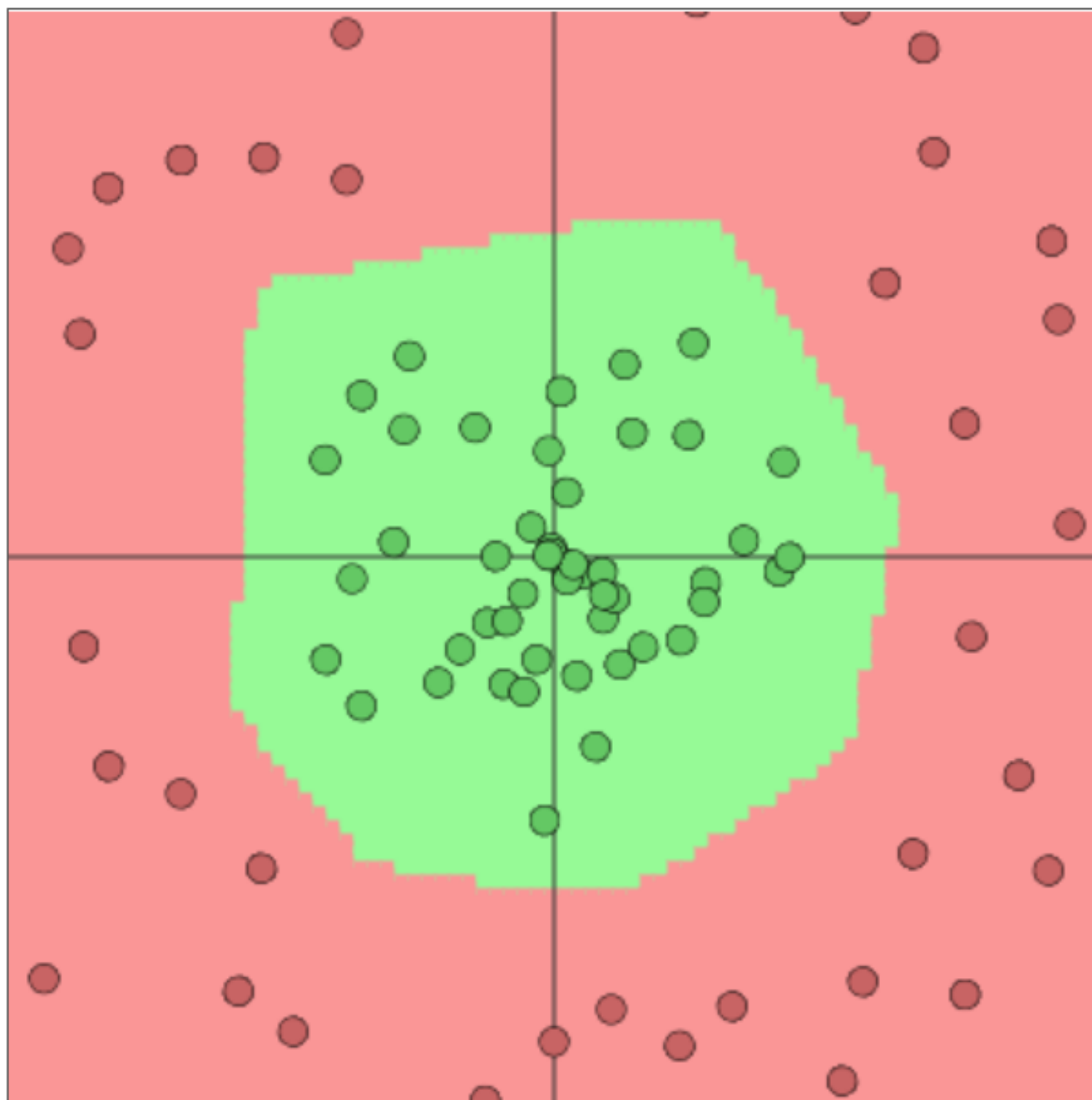
Em geral:

- `nn.ReLU()`
- `nn.Sigmoid()`
- `nn.Tanh()`



CAMADAS DE ATIVAÇÃO

<https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>



CAMADAS DE ATIVAÇÃO²

```
nn.Linear(2, 4)
```

```
nn.ReLU()
```

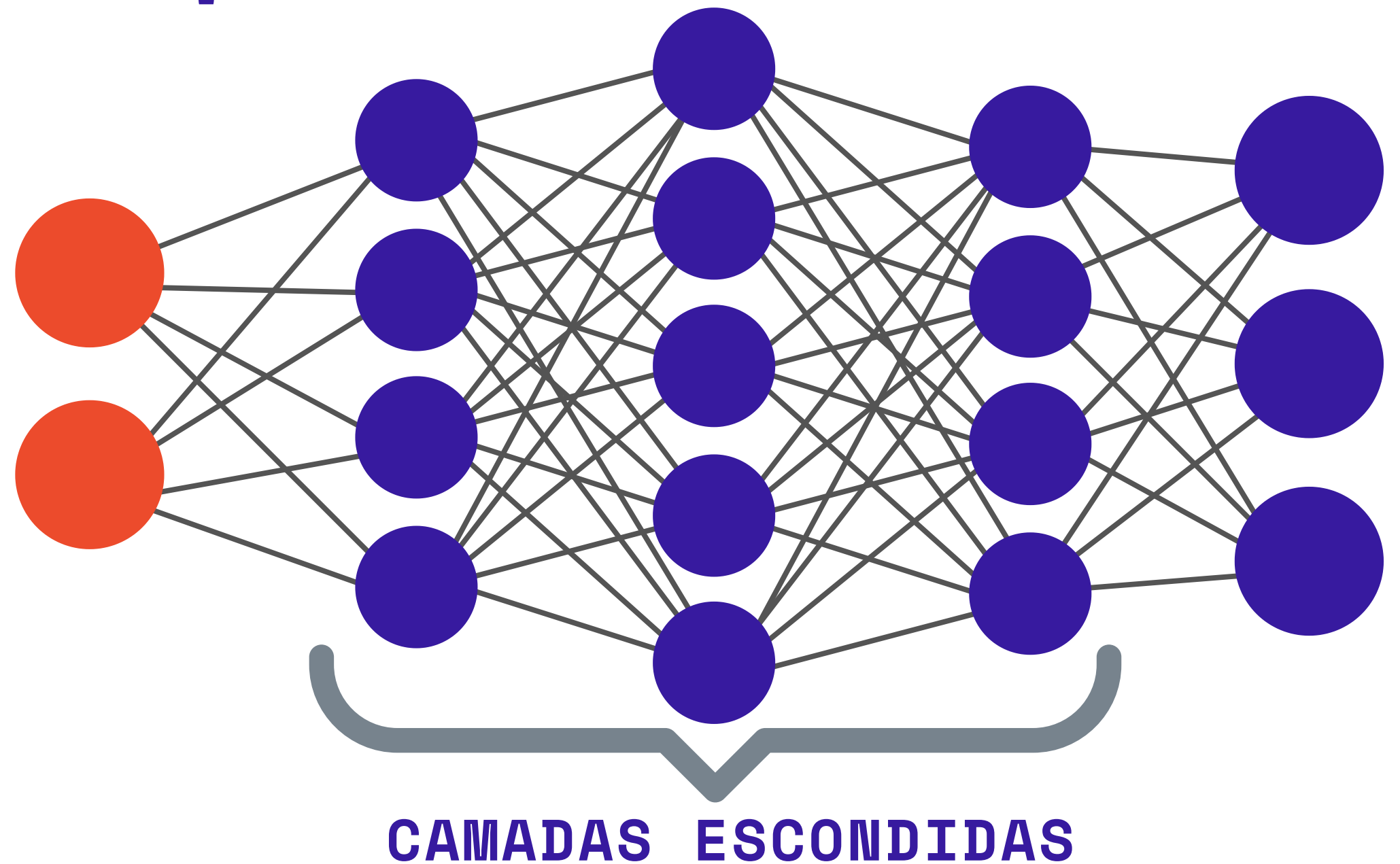
```
nn.Linear(4, 5)
```

```
nn.ReLU()
```

```
nn.Linear(5, 4)
```

```
nn.ReLU()
```

```
nn.Linear(4, 3)
```

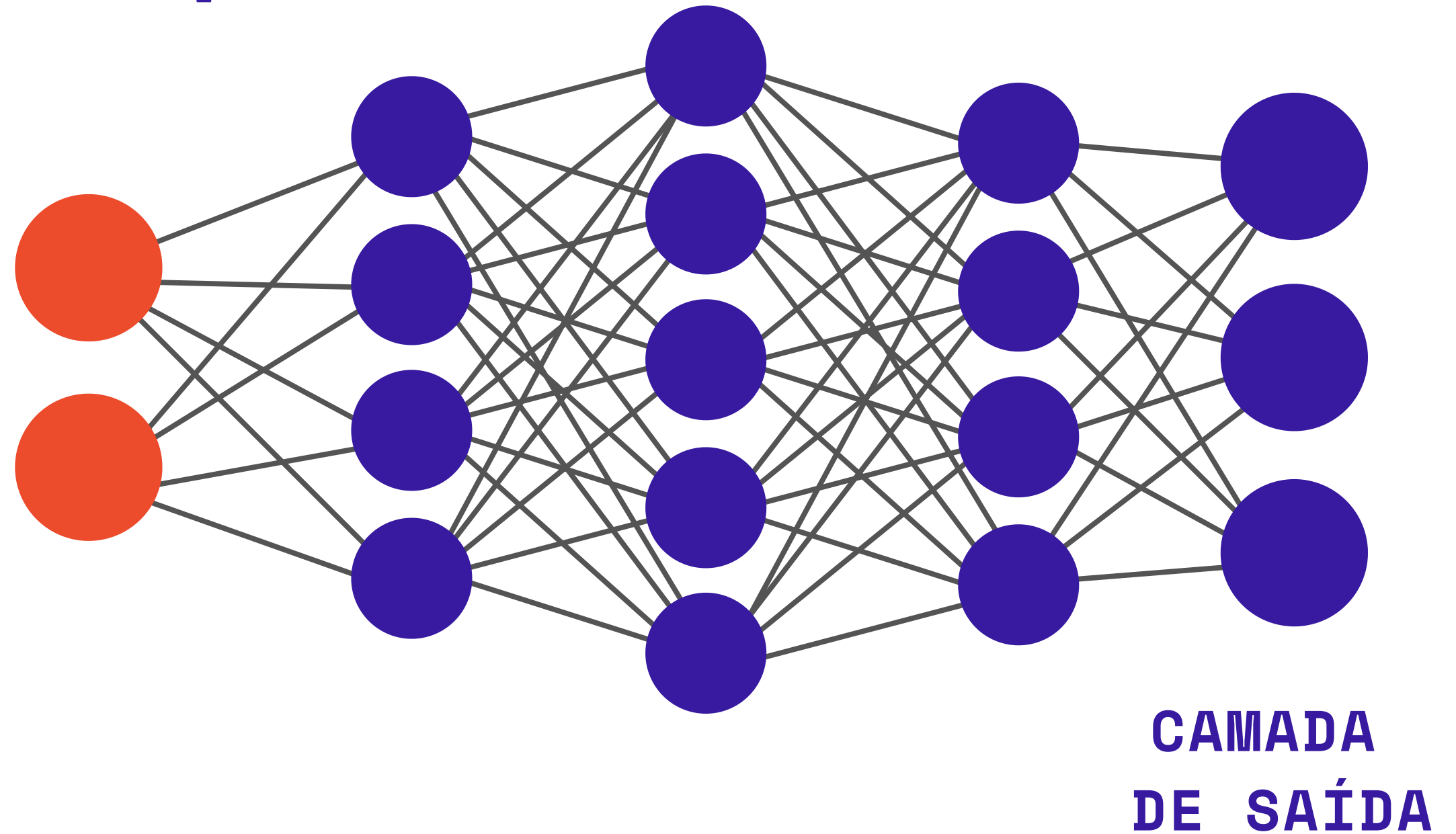


CAMADAS DE ATIVAÇÃO

Camada de saída pode ou não ser ativada. A ativação escolhida depende do problema.

Em geral:

- `nn.Softmax()`
- `nn.Sigmoid()`
- `nn.Tanh()`



CRIANDO UMA REDE NEURAL NO PYTORCH



ARQUITETURA.IPYNB

PRÁTICA: PARTE 1

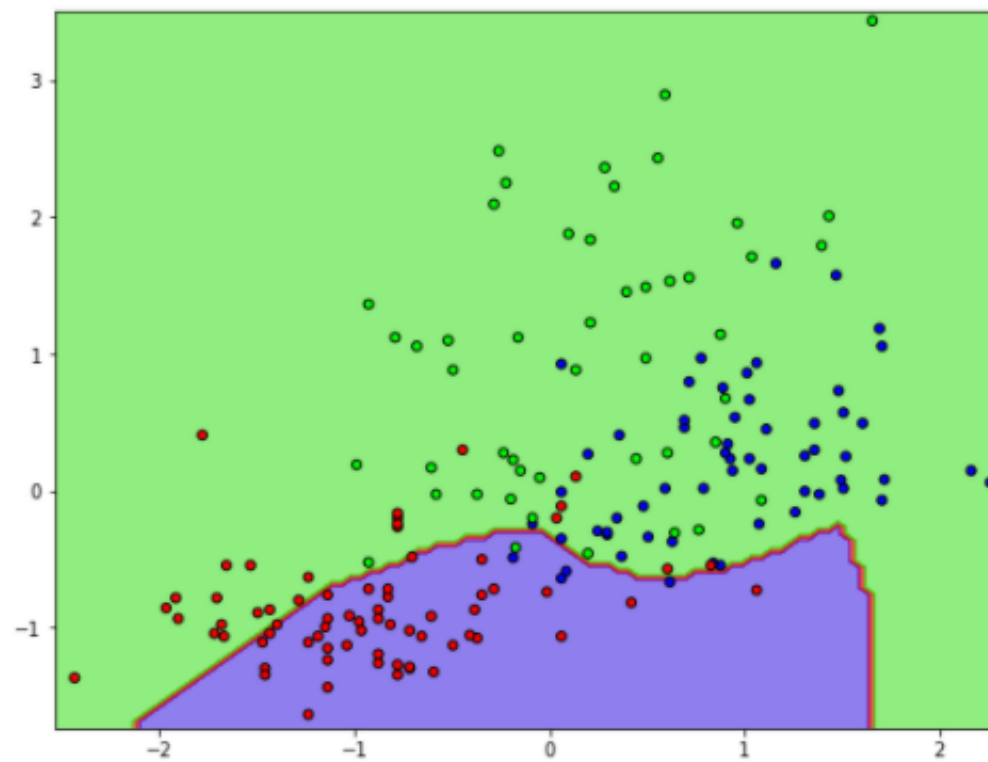


MLP_CLASSIFICATION_MNIST.IPYNB

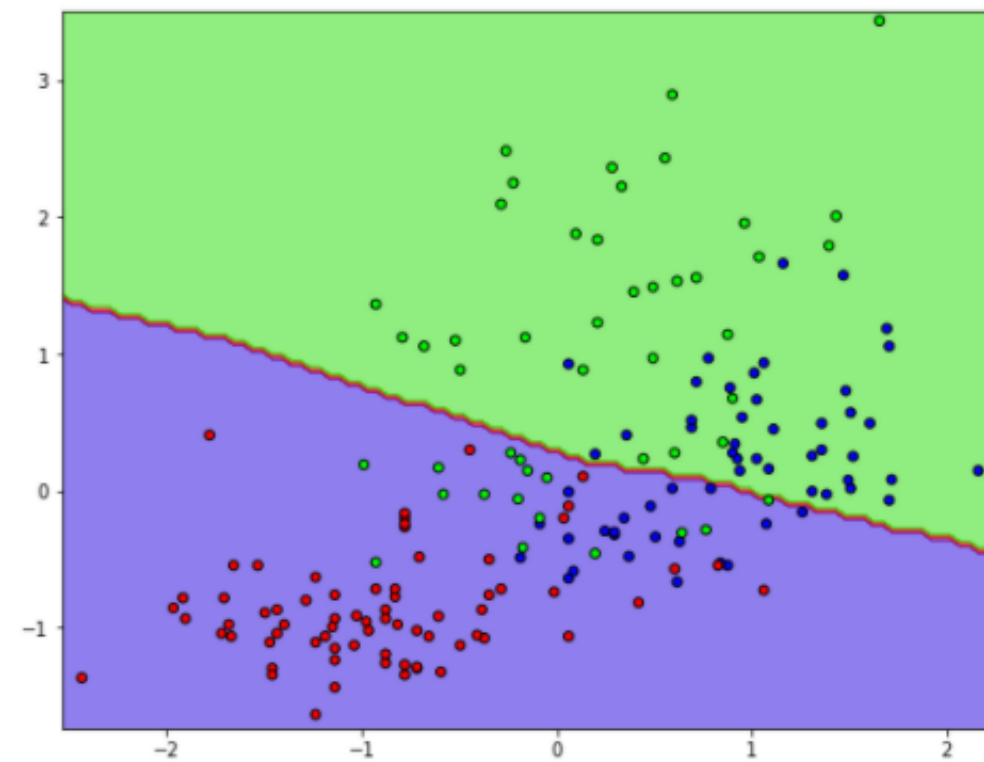
TREINANDO UMA REDE NEURAL NO PYTORCH

O TREINAMENTO É UM PROCESSO **ITERATIVO**

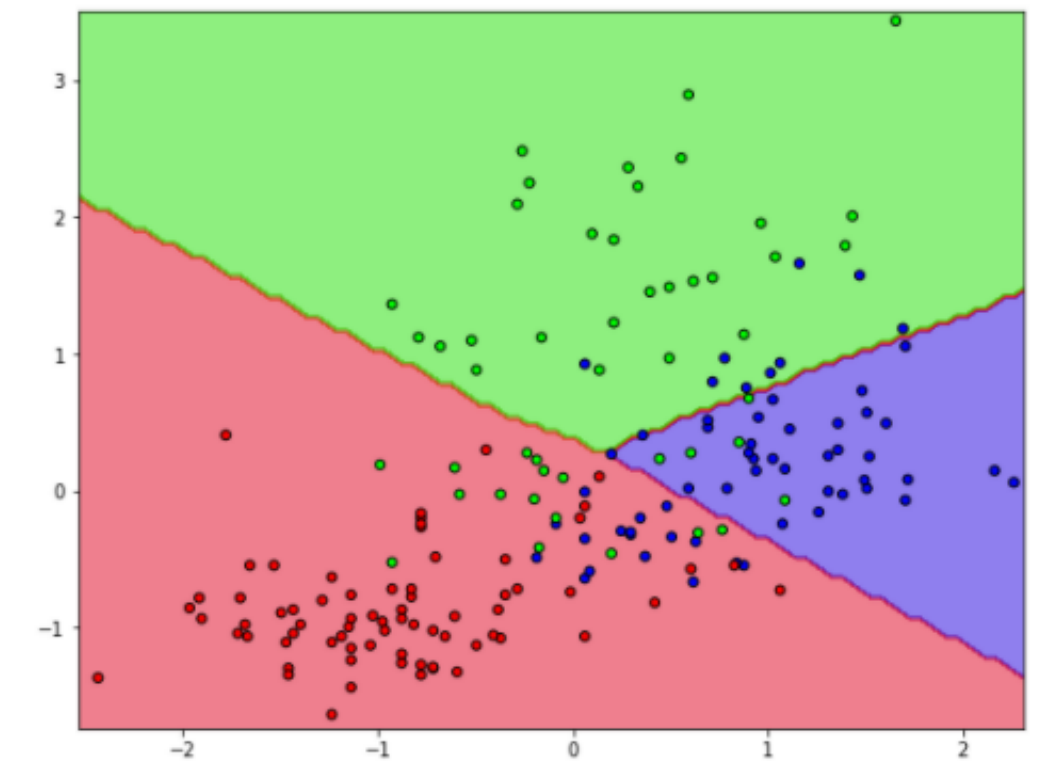
10 Iterações



50 Iterações



100 Iterações



TREINANDO UMA REDE NEURAL NO PYTORCH

O TREINAMENTO É UM PROCESSO **ITERATIVO**

Iteração

Batch

Época



TREINANDO UMA REDE NEURAL NO PYTORCH

O TREINAMENTO É UM PROCESSO ITERATIVO

A CADA **ITERAÇÃO**, A REDE PODE PROCESSAR

- TODOS OS DADOS
- ALGUNS DADOS
- UM ÚNICO DADO

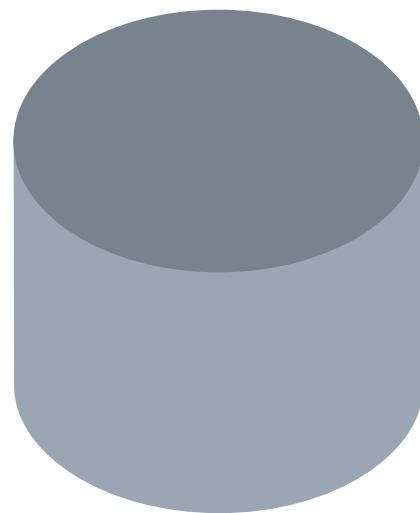


TREINANDO UMA REDE NEURAL NO PYTORCH

A CADA ITERAÇÃO, A REDE PODE PROCESSAR

- TODOS OS DADOS
- ALGUNS DADOS
- UM ÚNICO DADO

A QUANTIDADE DE DADOS QUE A REDE VAI PROCESSAR EM UMA ÚNICA ITERAÇÃO É O **BATCH**!



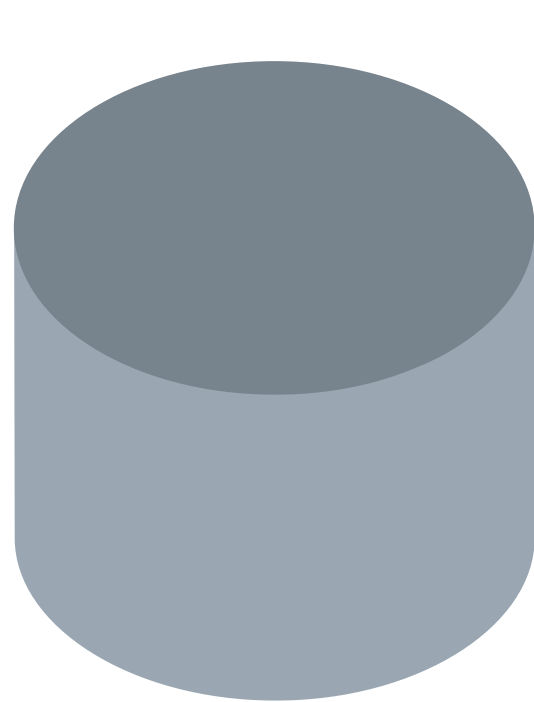
Dataset



Batches

TREINANDO UMA REDE NEURAL NO PYTORCH

O CONJUNTO DE TREINO PODE SER DIVIDIDOS EM **BATCHES** DE MESMO TAMANHO. É COMPLETADA 1 **ÉPOCA** QUANDO A REDE PROCESSOU TODOS OS BATCHES (VIU TODOS OS DADOS).



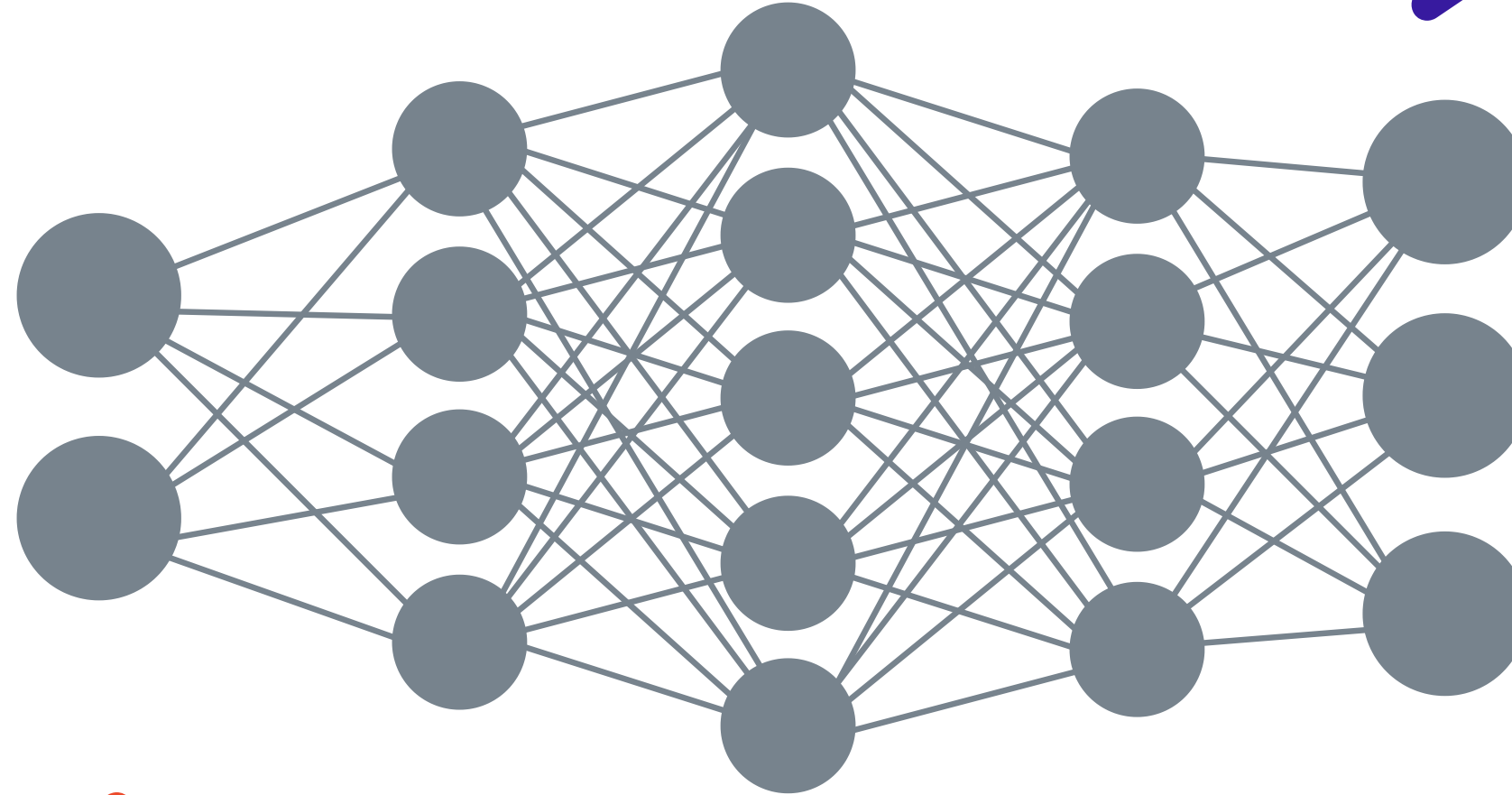
Dataset



Batches

TREINANDO UMA REDE NEURAL NO PYTORCH

FLUXO DO DADO



FLUXO DE OTIMIZAÇÃO

TREINANDO UMA REDE NEURAL NO PYTORCH

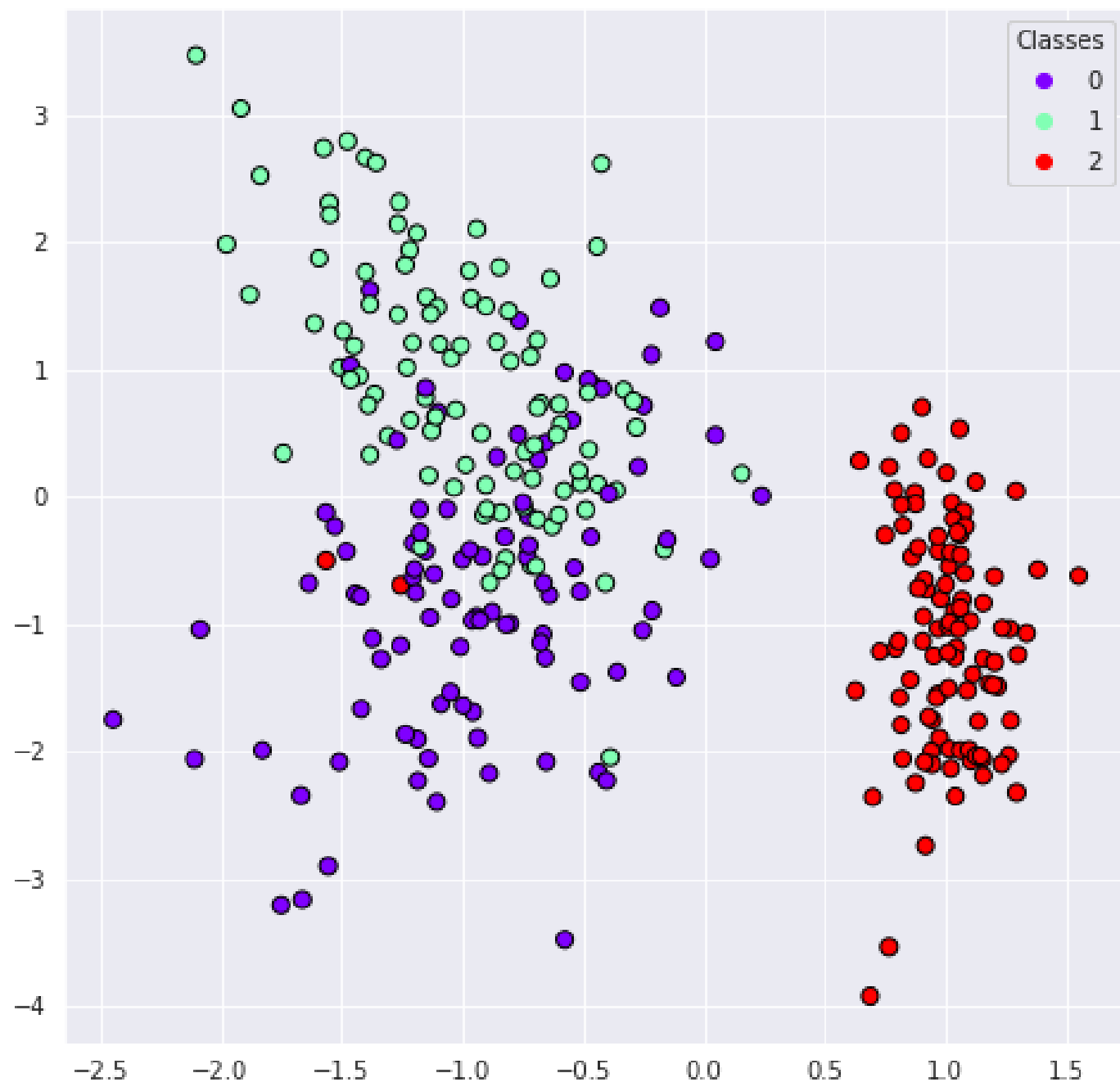
FORWARD



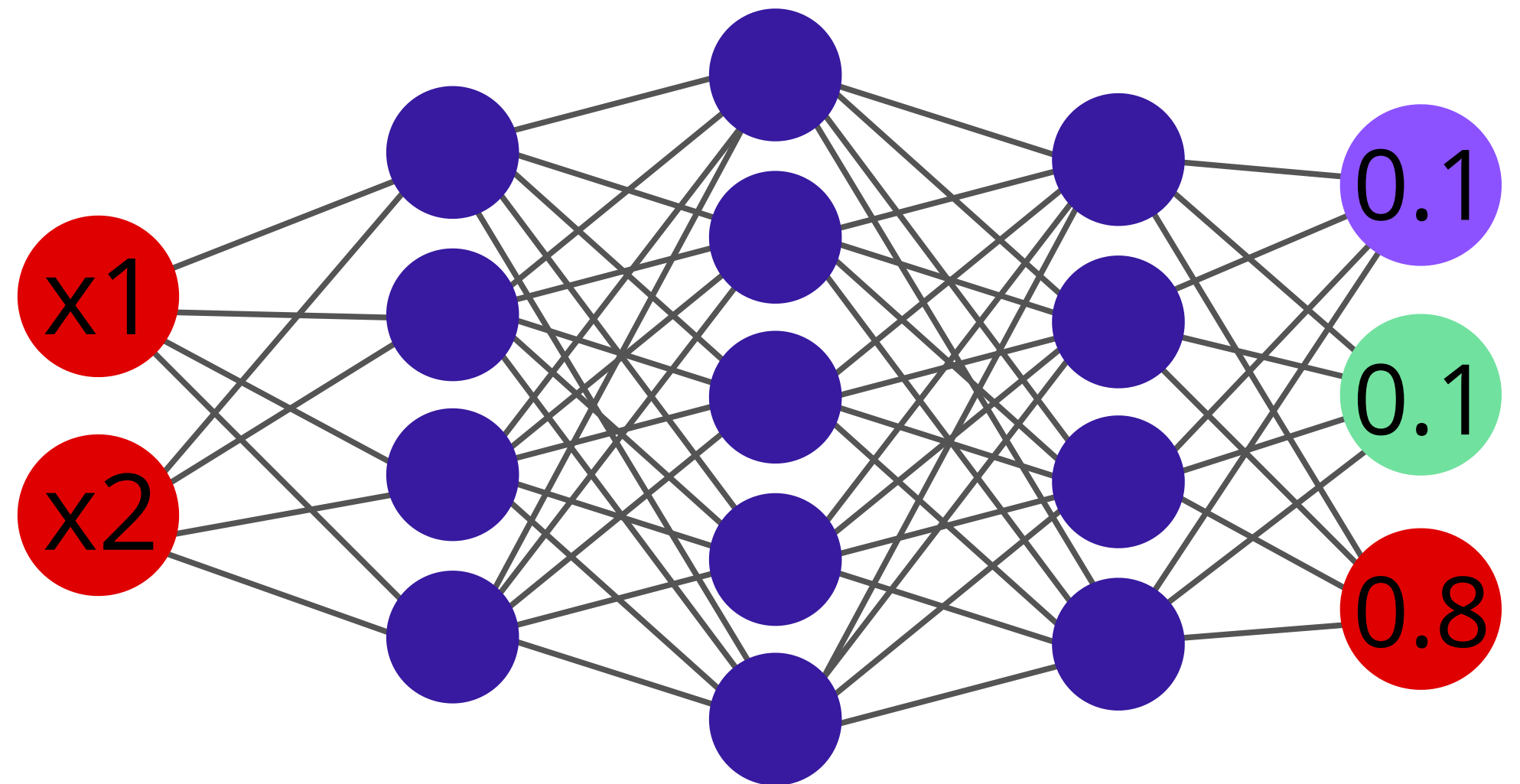
```
# Passar os dados na rede  
pred = net(X)  
  
# Calcular a qualidade da predição  
????
```


TREINANDO UMA REDE NEURAL NO PYTORCH

classes 0, 1, 2



```
pred = tensor([0.1, 0.1, 0.8])  
rotulo = tensor( 2 )
```



TREINANDO UMA REDE NEURAL NO PYTORCH

```
pred = tensor([0.1, 0.1, 0.8])  
rotulo = tensor( 2 )
```

- **Funções de perda**

- Critério de qualidade para avaliar a predição em relação ao rótulo.

TREINANDO UMA REDE NEURAL NO PYTORCH

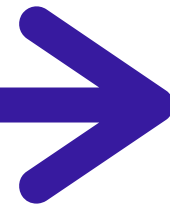
```
pred = tensor([0.1, 0.1, 0.8])  
rotulo = tensor( 2 )
```

- **Funções de perda**

- Distância Absoluta `nn.L1Loss()`
 $| \text{pred} - \text{rotulo} |$
- Distância Quadrática `nn.MSELoss()`
 $| \text{pred} - \text{rotulo} |^2$
- **Entropia Cruzada**
`nn.CrossEntropyLoss()`

TREINANDO UMA REDE NEURAL NO PYTORCH

FORWARD



```
# Define o critério de qualidade
```

```
criterio = nn.CrossEntropyLoss()
```

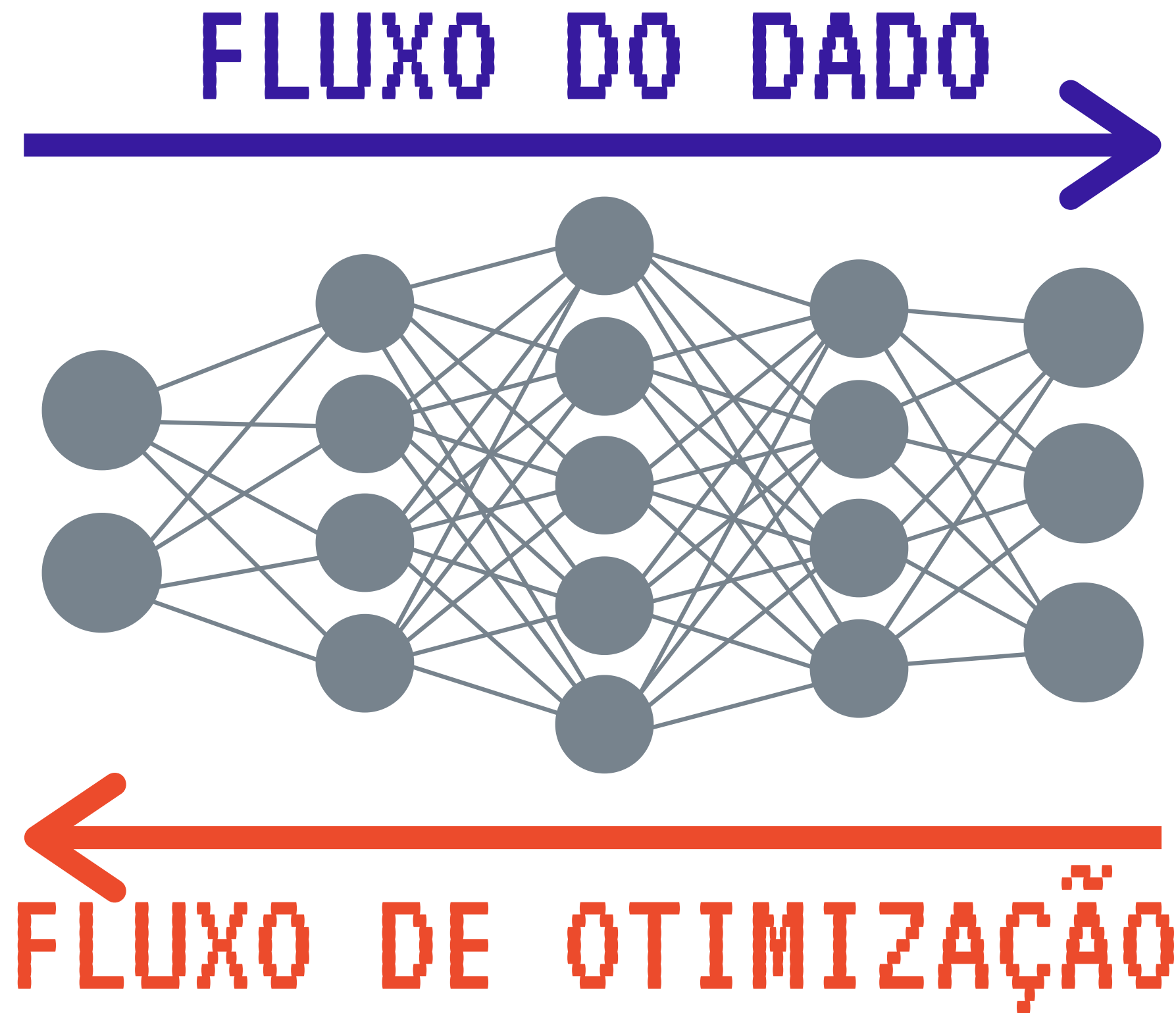
```
# Passar os dados na rede
```

```
pred = net(X)
```

```
# Calcular a qualidade da predição
```

```
loss = criterio(pred, rotulo)
```

TREINANDO UMA REDE NEURAL NO PYTORCH



TREINANDO UMA REDE NEURAL NO PYTORCH


FLUXO DE OTIMIZAÇÃO

```
# Define o otimizador  
  
# Zera o gradiente do otimizador  
  
# Calcula o novo gradiente  
  
# Atualiza os pesos da rede neural
```

O TAL DO GRADIENTE...

- A função de custo calcula a qualidade da predição de seu modelo.
 - `loss = critério(pred, rotulo)`
- A derivação automática que vimos mais cedo calcula a **direção e magnitude** da atualização dos pesos da rede.
 - `loss.backward()`
 - Os pesos vão aumentar + ou diminuir - ?
 - Quão grande foi o erro?

TREINANDO UMA REDE NEURAL NO PYTORCH



FLUXO DE OTIMIZAÇÃO

```
# Define o otimizador
optimizer = ???

# Zera o gradiente do otimizador
optimizer.zero_grad()

# Calcula o novo gradiente
loss.backward()

# Atualiza os pesos da rede neural
optimizer.step()
```

TREINANDO UMA REDE NEURAL NO PYTORCH

- **Otimizadores**

- O PyTorch possui uma extensa biblioteca de otimizadores

`torch.optim`

TREINANDO UMA REDE NEURAL NO PYTORCH

- **Otimizadores**

- O PyTorch possui uma extensa biblioteca de otimizadores
`torch.optim`

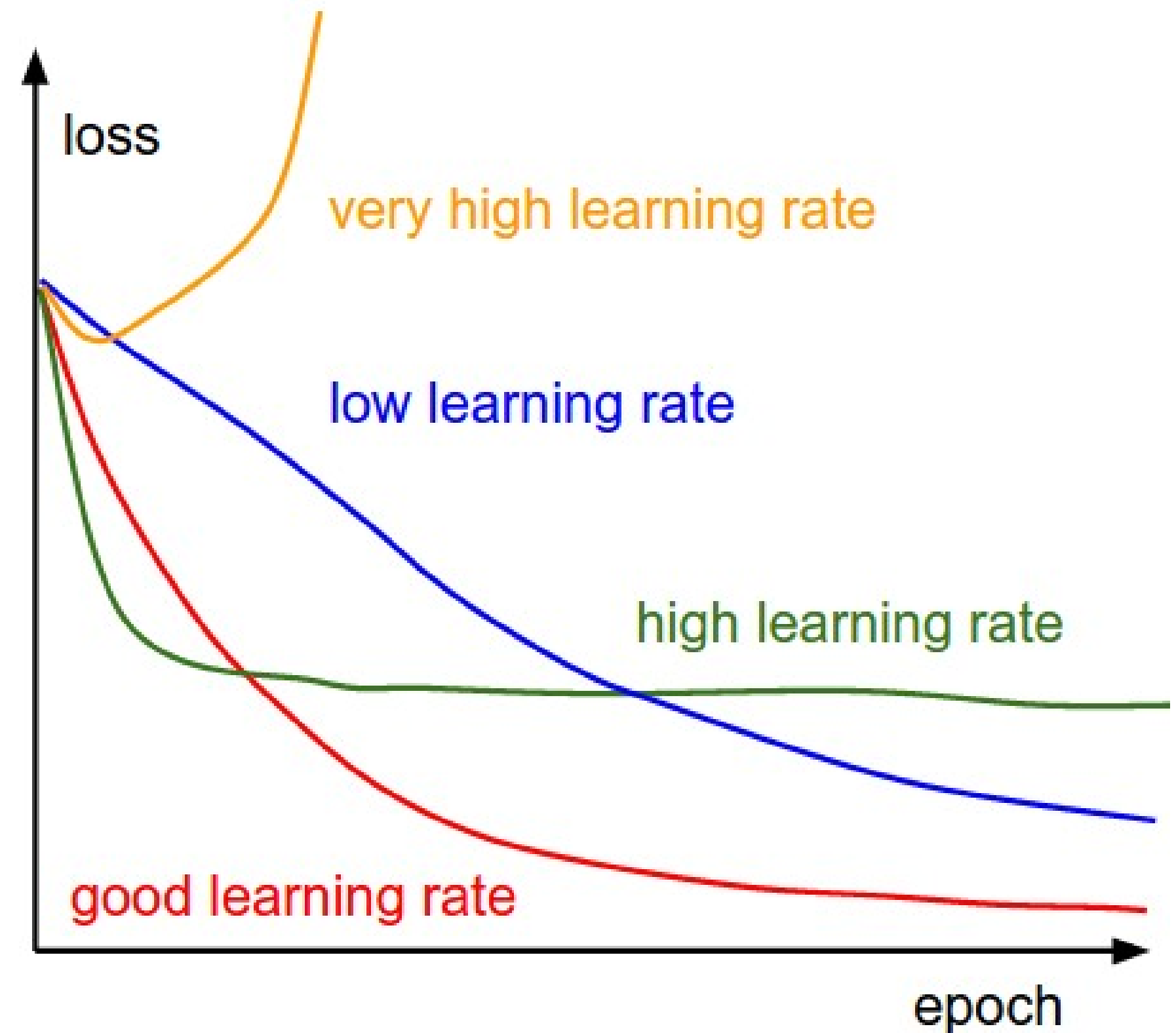
- **SGD: o clássico**

`optimizer = torch.optim.SGD(net.parameters(), lr=0.001)`

- `lr`: *learning rate* / taxa de aprendizado

TREINANDO UMA REDE NEURAL NO PYTORCH

- **lr:** *learning rate*,
taxa de aprendizado



TREINANDO UMA REDE NEURAL NO PYTORCH

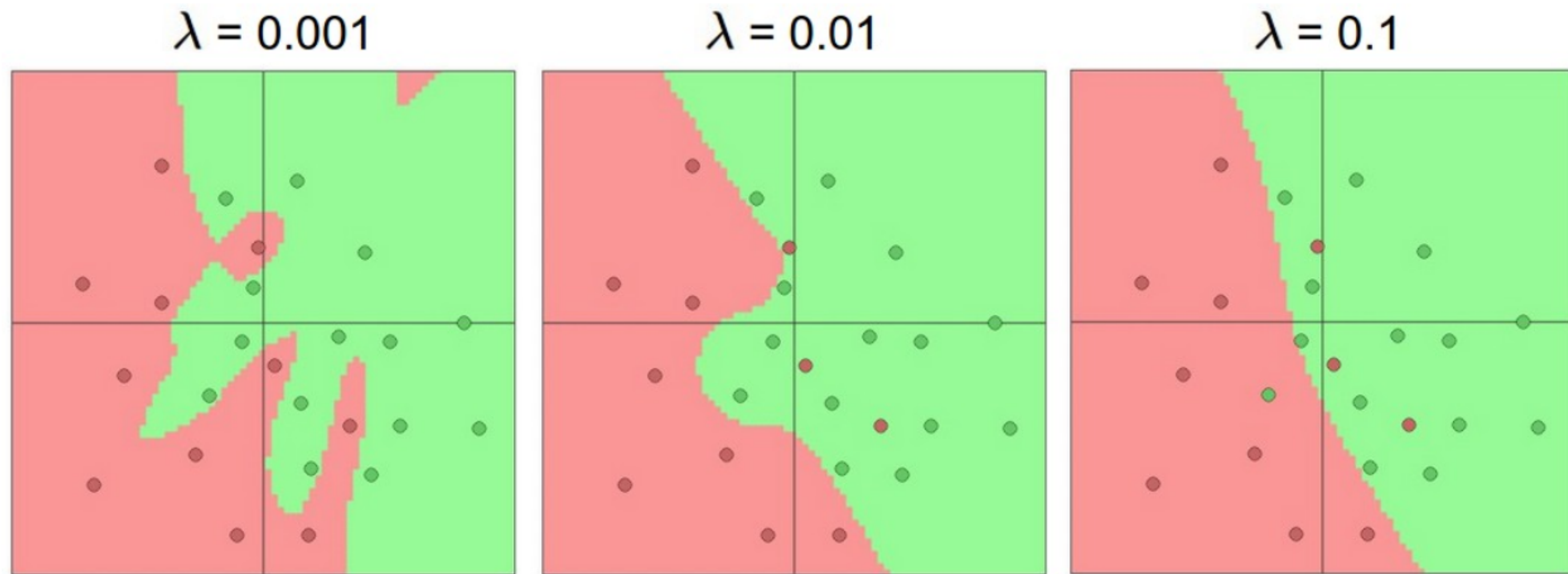
- Adam: um dos melhores

```
optimizer = torch.optim.Adam(net.parameters(),  
lr=0.001, weight_decay=0.0005)
```

- weight_decay: regularização

TREINANDO UMA REDE NEURAL NO PYTORCH

- `weight_decay`: regularização



TREINANDO UMA REDE NEURAL NO PYTORCH



FLUXO DE OTIMIZAÇÃO

```
# Define o otimizador
optimizer = torch.optim.Adam(net.parameters(),
lr=0.001, weight_decay=0.0005)
# Zera o gradiente do otimizador
optimizer.zero_grad()
# Calcula o novo gradiente
loss.backward()
# Atualiza os pesos da rede neural
optimizer.step()
```

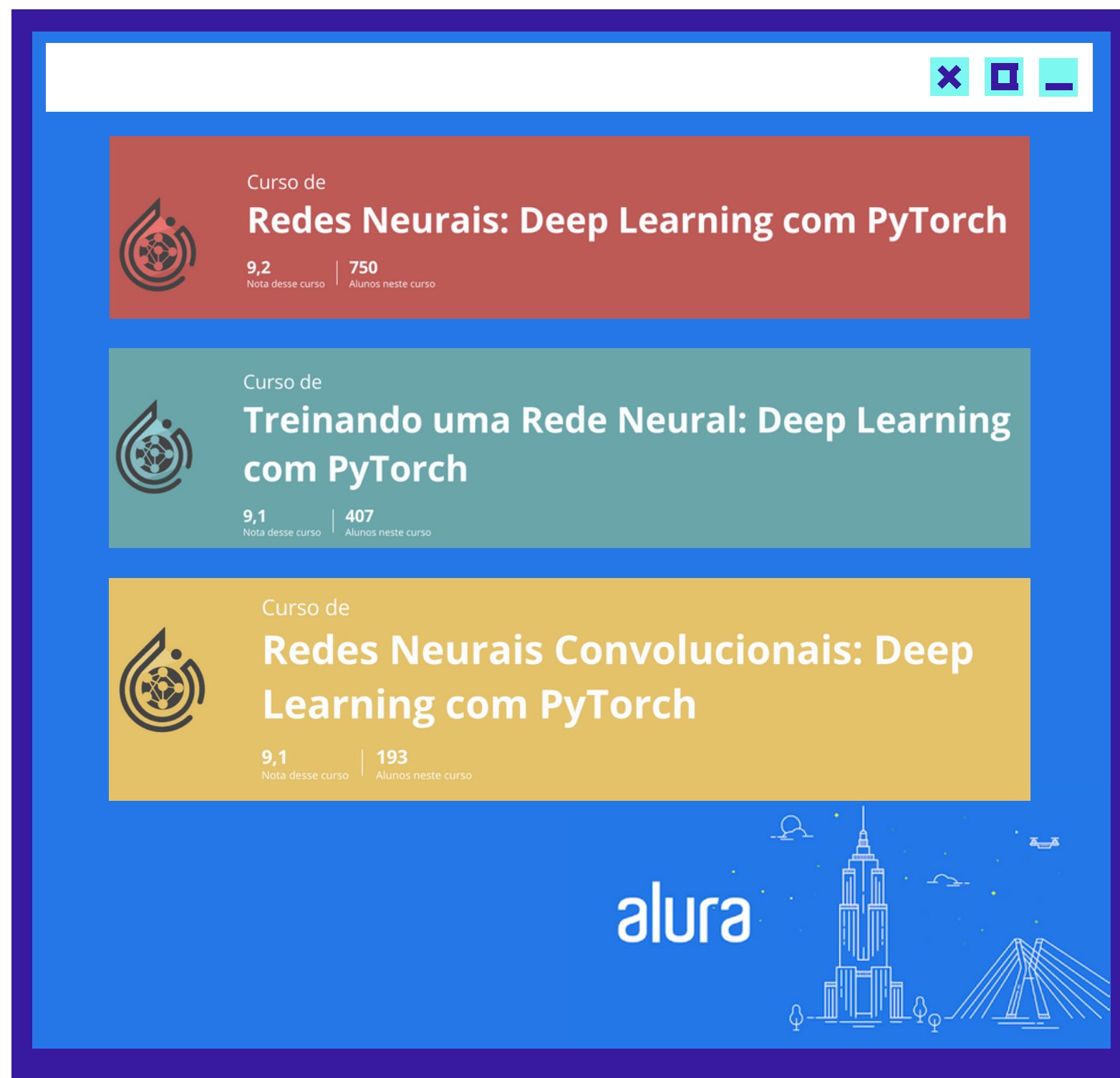

TREINANDO UMA REDE NEURAL NO PYTORCH

- Antes de tudo
 - Define o **critério**
 - Define o **otimizador**
- Forward (Fluxo do dado)
 - Alimentar os dados para a rede
`pred = net(dado)`
 - Calcular a função de custo
`loss = critério(pred, rotulo)`
- Backpropagation (Otimização)
 - Zerar o gradiente do otimizador
`optimizer.zero_grad()`
 - Calcular o novo gradiente
`loss.backward()`
 - Atualizar os pesos
`optimizer.step()`

PRÁTICA: PARTE 2



MLP_CLASSIFICATION_MNIST.IPYNB



Curso de
Redes Neurais: Deep Learning com PyTorch
9,2 Nota desse curso | 750 Alunos neste curso

Curso de
Treinando uma Rede Neural: Deep Learning com PyTorch
9,1 Nota desse curso | 407 Alunos neste curso

Curso de
Redes Neurais Convolucionais: Deep Learning com PyTorch
9,1 Nota desse curso | 193 Alunos neste curso

alura



@MILALARANJEIRA
@CANALPEIXEBABEL

CANAL PEIXE BABEL

OBRIGADA!

<https://www.alura.com.br/promocao/peixebabel>

(4)

7
1
-2
13

(4, 5)

7	-1	3	11	0
1	22	2	4	9
-2	3	7	12	11
13	1	0	0	2

(4, 5, 3)

7	-1	3	11	0
1	22	2	4	9
-2	3	7	12	11
13	1	0	0	2



(4, 5, 3, 4)



(4, 5, 3, 4, 5)

