Aprendizado Profundo

Caio Lucas da Silva Chacon
Deivison Rodrigues Jordão
Luiz Fernando Costa dos Santos

Pytorch: Como funciona

- Tensors
- Datasets
- DataLoader
- Processo de Treinamento
- Processo de avaliação

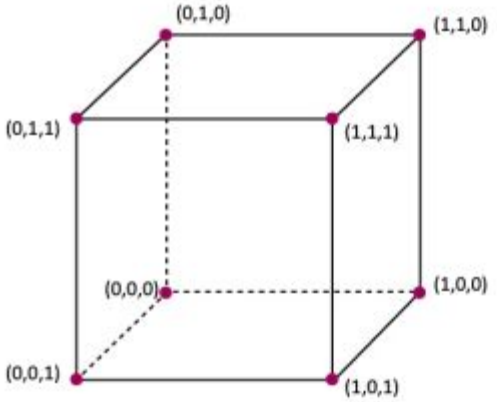
Dataset

```
class CustomDataset(Dataset):
    def __init__(self, …):
        …
    def __len__(self):
        …
    def __getitem__(self, idx):
        …
```

Dataloader

```
DataLoader(
    training_data or test_data,
    batch_size,
    …
)
```

Questão 1



- Dataset

  número de exemplos: 1000000

  divisão: 70 treino, 20 validação, 10 teste

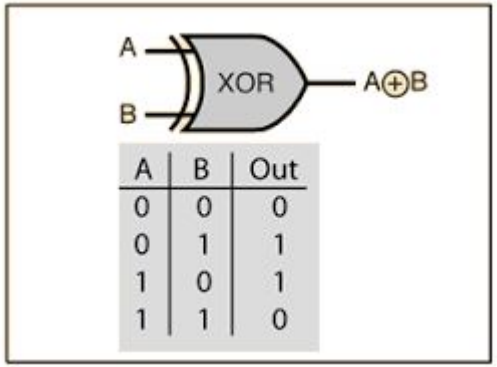
- Arquitetura

```
RosemblattPerceptron(
  (perceptron): Linear(in_features=3, out_features=8, bias=True)
)
```

- Acurácia: 100%

```
Test Error:
 Accuracy: 100.0%, Avg loss: 1.321413
```

Questão 2
a) XOR



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

- Dataset

  número de exemplos: 4

  divisão: 4 treino, 4 teste

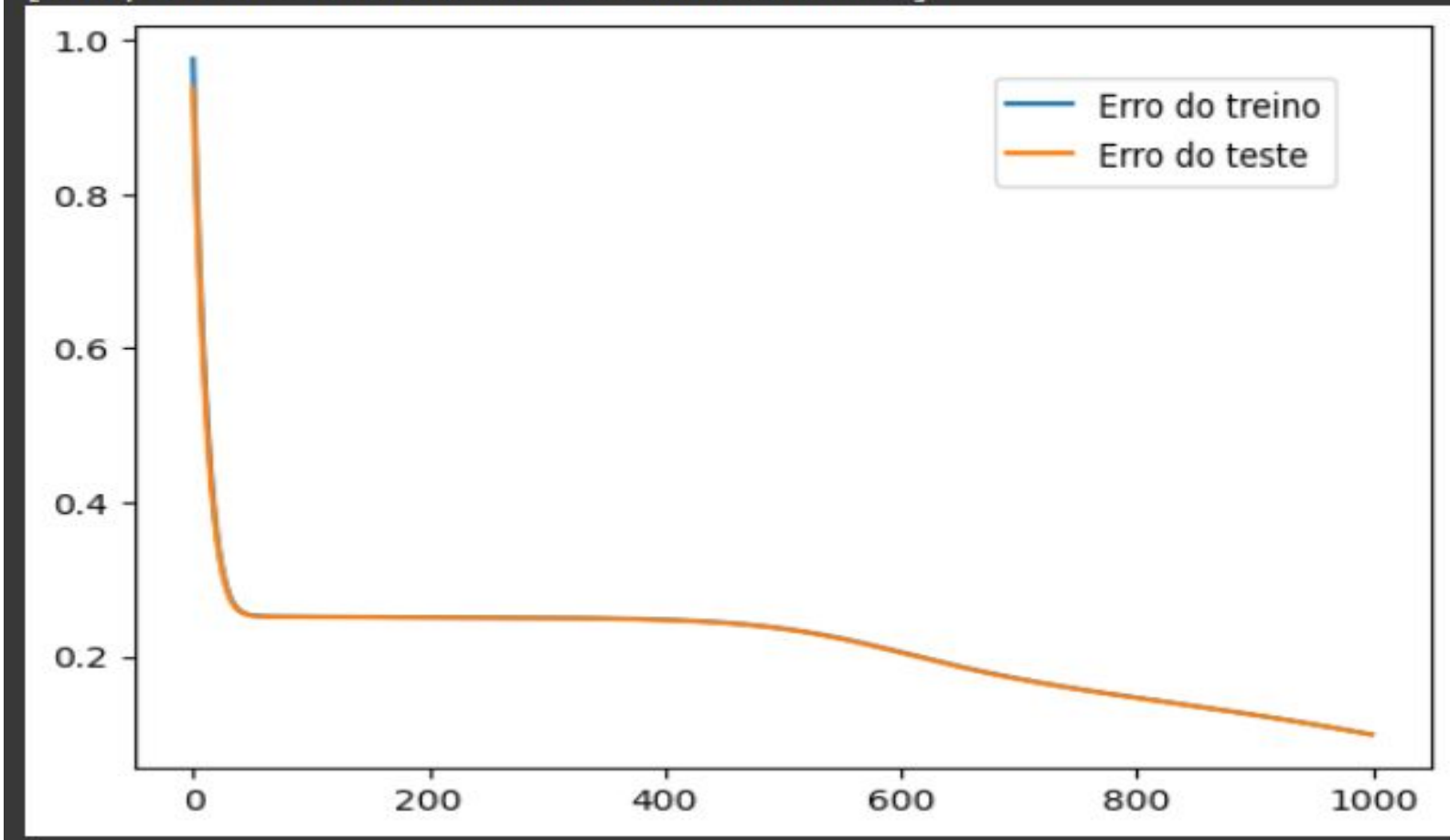
- Arquitetura: Adam, lr=0.005, MSE

```
MultiLayerPerceptron(
  (linear): Linear(in_features=2, out_features=2, bias=True)
  (Sigmoid): Sigmoid()
  (linear2): Linear(in_features=2, out_features=1, bias=True)
)
```

- Acurácia:

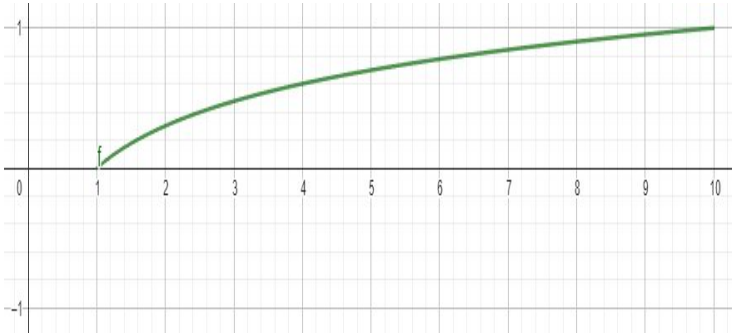
```
Epoch 1000
-------------------------------
loss: 0.001433  [    1/    4]
Test Error:
 Accuracy: 75.0%, Avg loss: 0.098248
```

```
[<matplotlib.lines.Line2D at 0x7916fe8308b0>]
```



Questão 2

b) $f(x) = \log_{10}(x)$, onde $1 \leq x \leq 10$

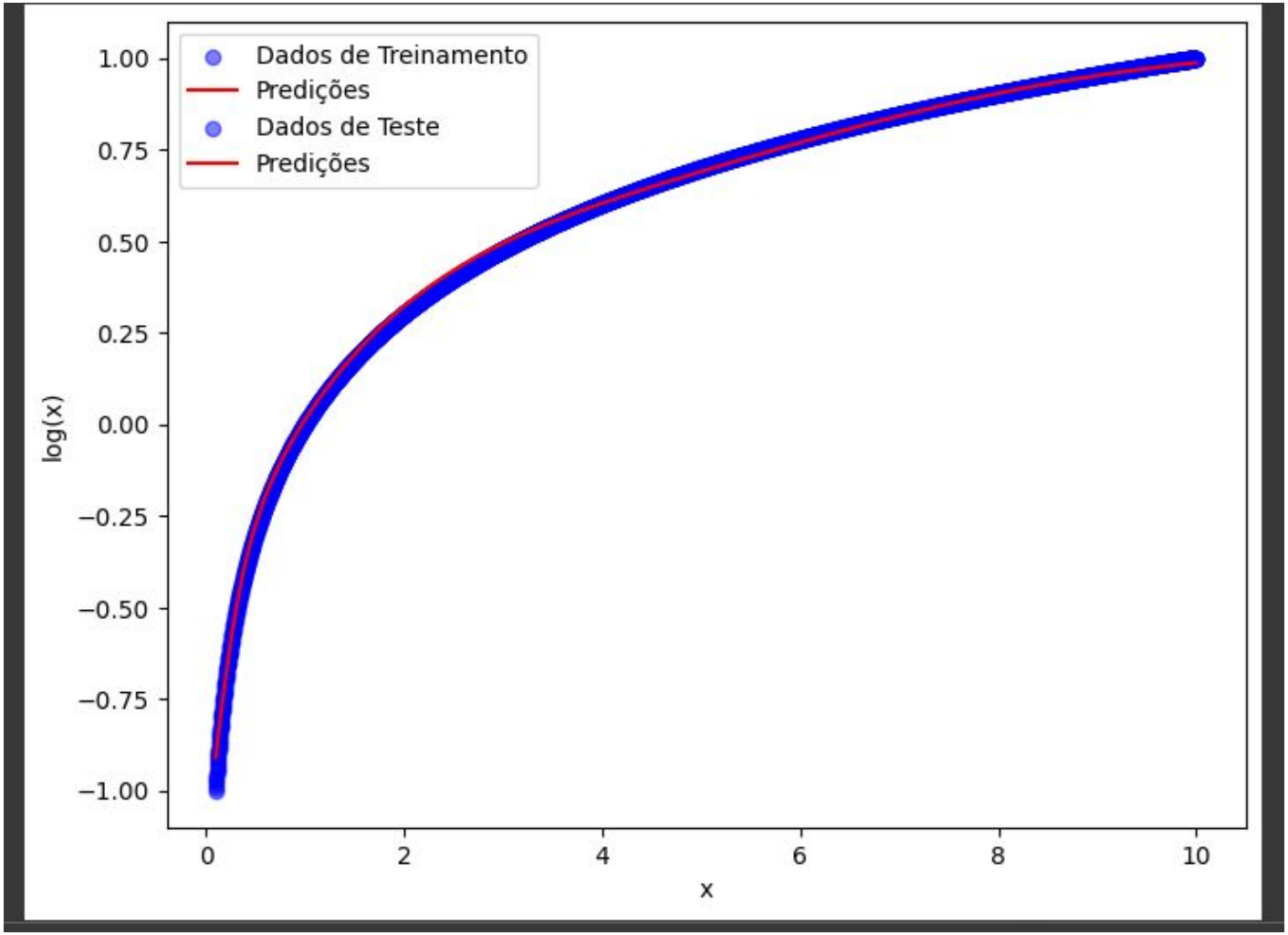


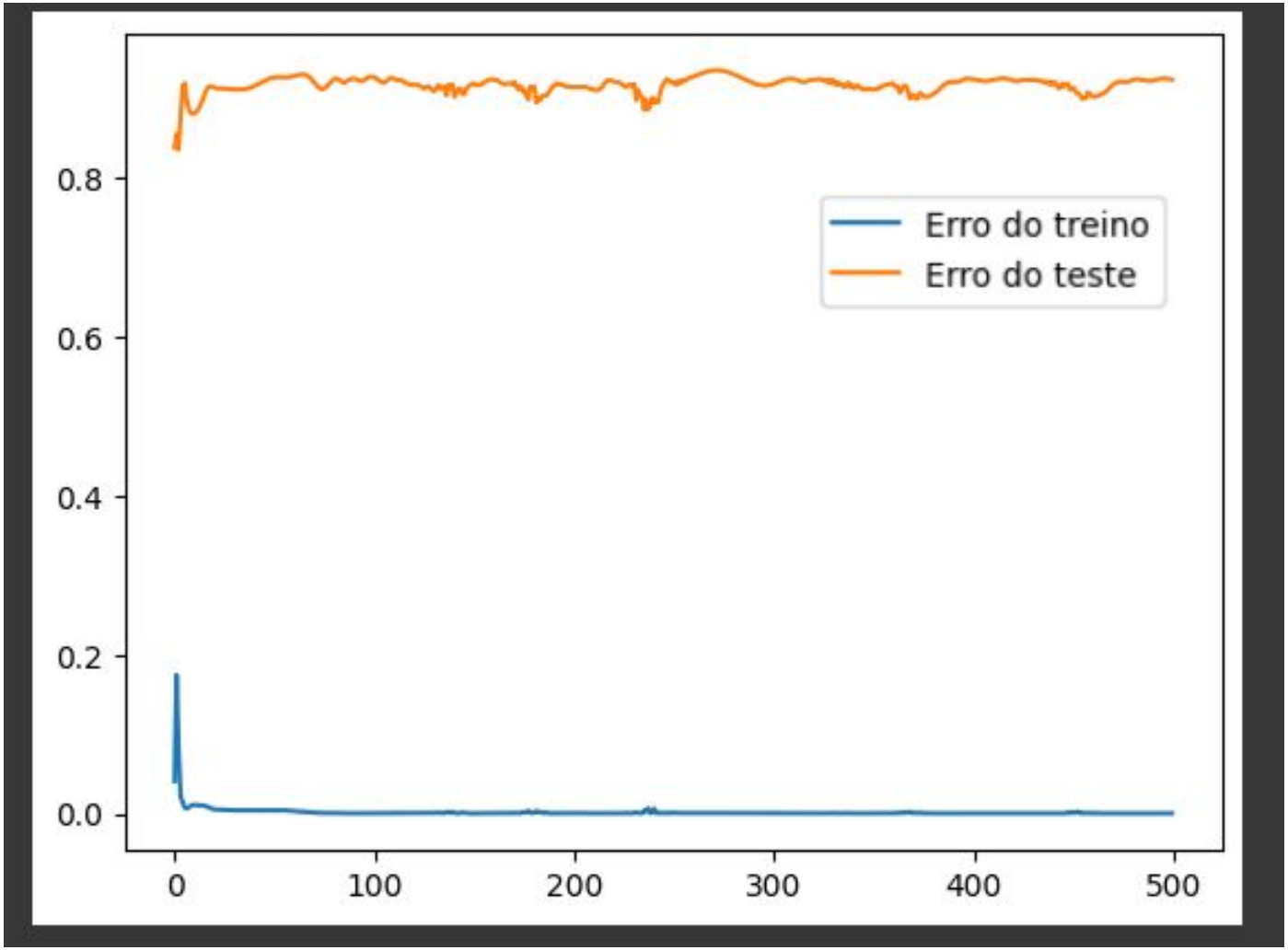
- Dataset:
  - 5000 exemplos, divisão 85/15
  - batch_size = 64
- Arquitetura: Adam, lr=0.01, MSE

```
LogMultiLayerPerceptron(
  (linear): Linear(in_features=1, out_features=128, bias=True)
  (Sigmoid): Sigmoid()
  (linear2): Linear(in_features=128, out_features=1, bias=True)
)
```

- Loss

```
Test Error:
 Avg loss: 0.923831
```


Dados de Treinamento
Predições
Dados de Teste
Predições
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
log(x)
0
2
4
6
8
10
x


0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
Erro do treino
Erro do teste

Questão 2

c) $f(x) = 10x^5 + 5x^4 + 2x^3 - 0.5x^2 + 3x + 2$

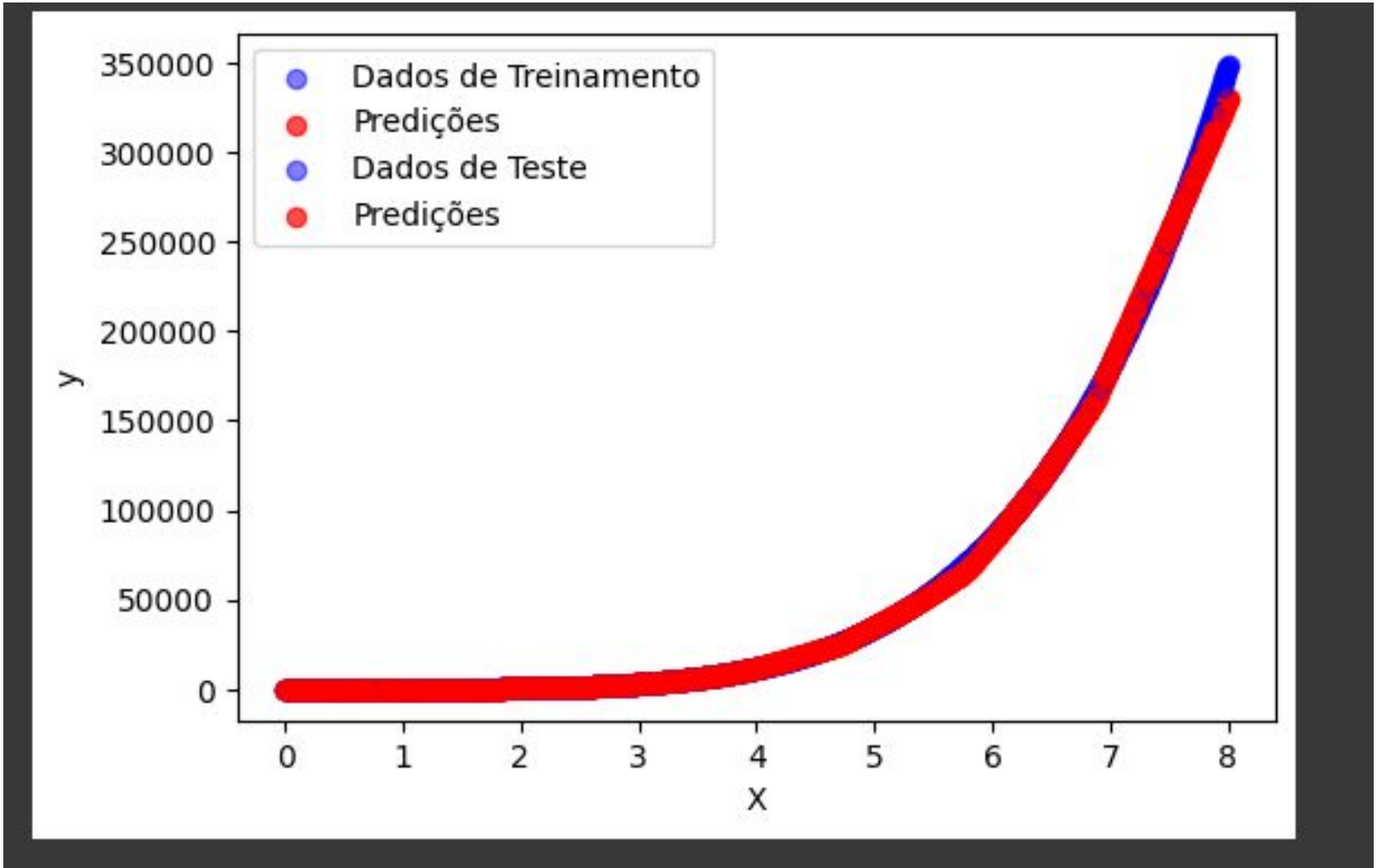
onde 0 ≤ x ≤ 8

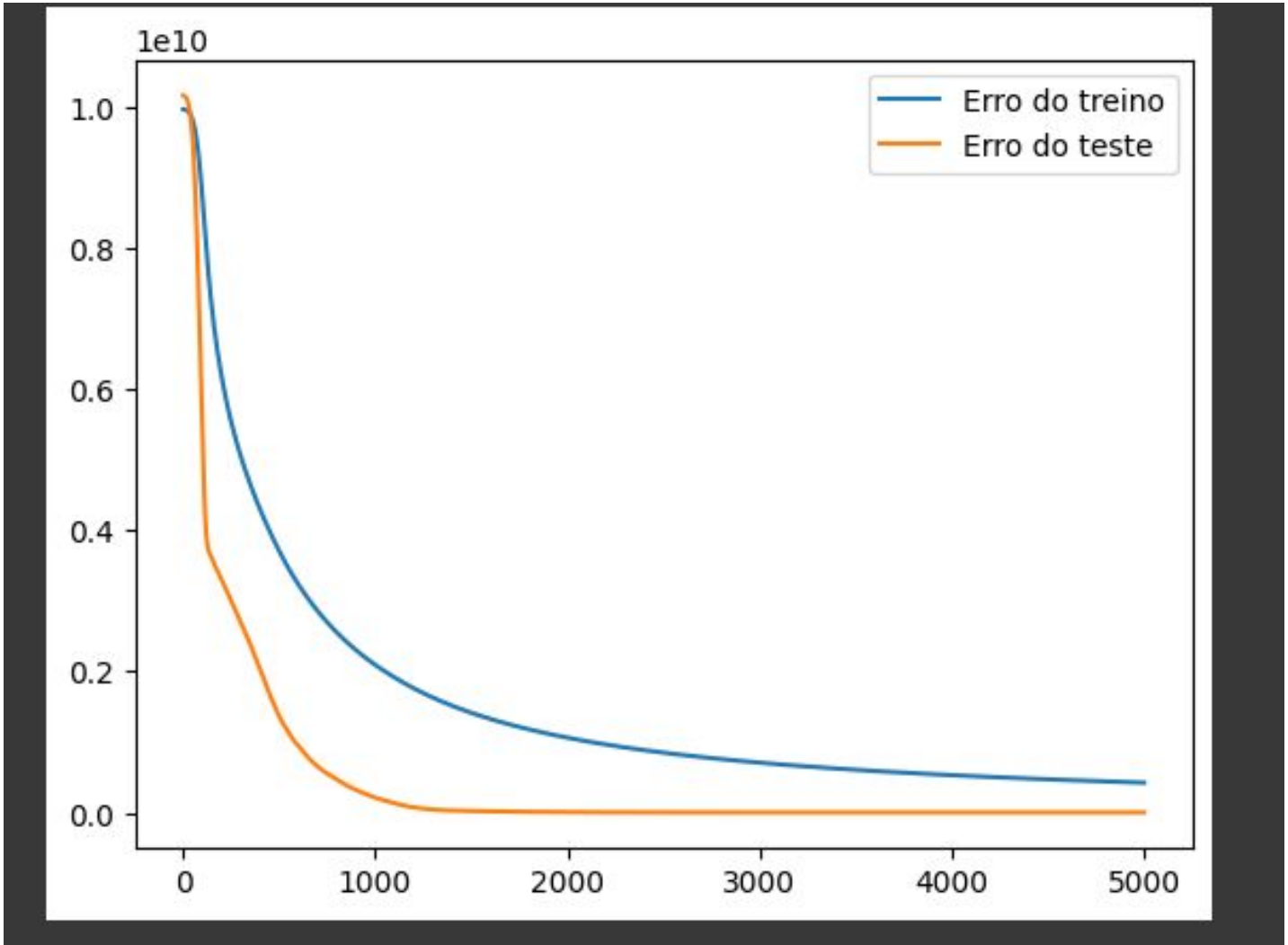
- Dataset:
  - 2000 exemplos, divisão aleatória 80/20
  - batch_size = 32
- Arquitetura: Adam, lr=0.01, MSE

```
PoynomialModel(
  (linear): Linear(in_features=1, out_features=128, bias=True)
  (ReLU): ReLU()
  (linear2): Linear(in_features=128, out_features=64, bias=True)
  (ReLU2): ReLU()
  (linear3): Linear(in_features=64, out_features=1, bias=True)
)
```

- Loss

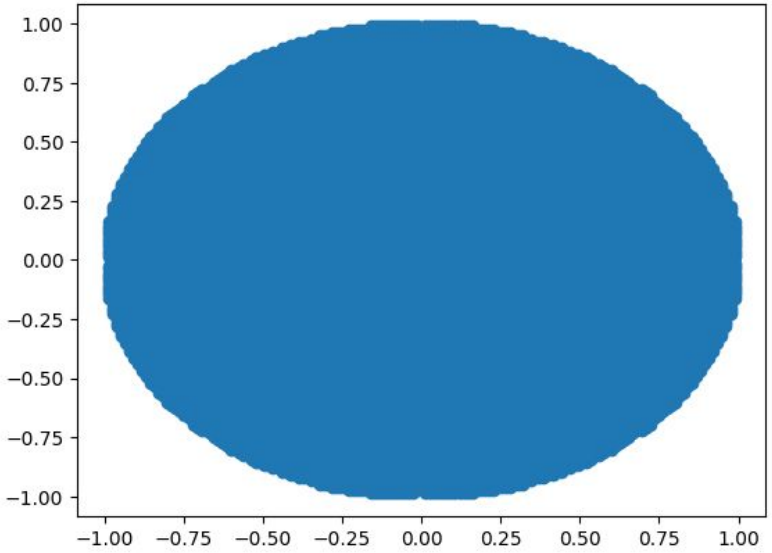
```
Epoch 5000
-------------------------------
Test Error:
 Avg loss: 24364401.406250
```

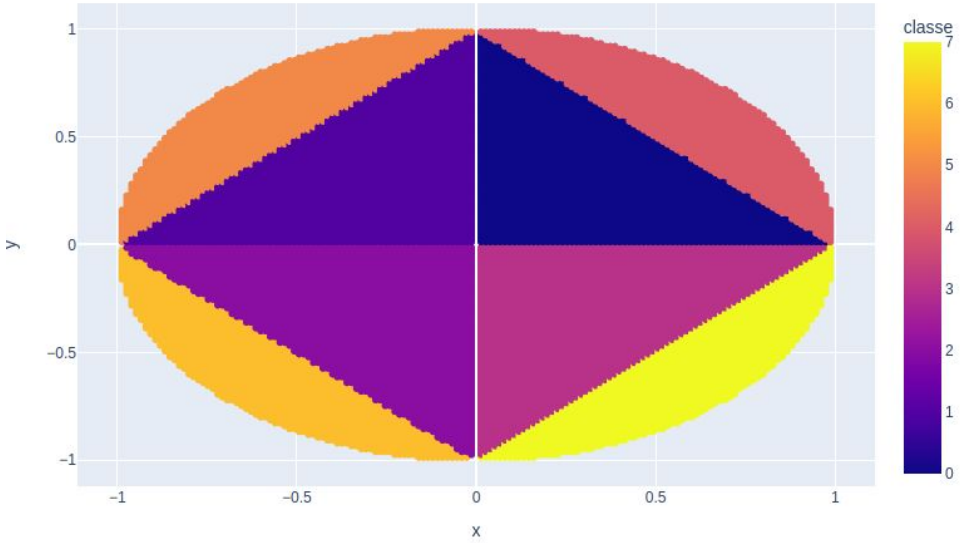

Dados de Treinamento
Predições
Dados de Teste
Predições
350000
300000
250000
200000
150000
100000
50000
0
y
0
1
2
3
4
5
6
7
8
X


1e10
1.0
0.8
0.6
0.4
0.2
0.0
0
1000
2000
3000
4000
5000
Erro do treino
Erro do teste

Questão 3

- Dataset
  - número de exemplos: 17246, divisão aleatória 80/20



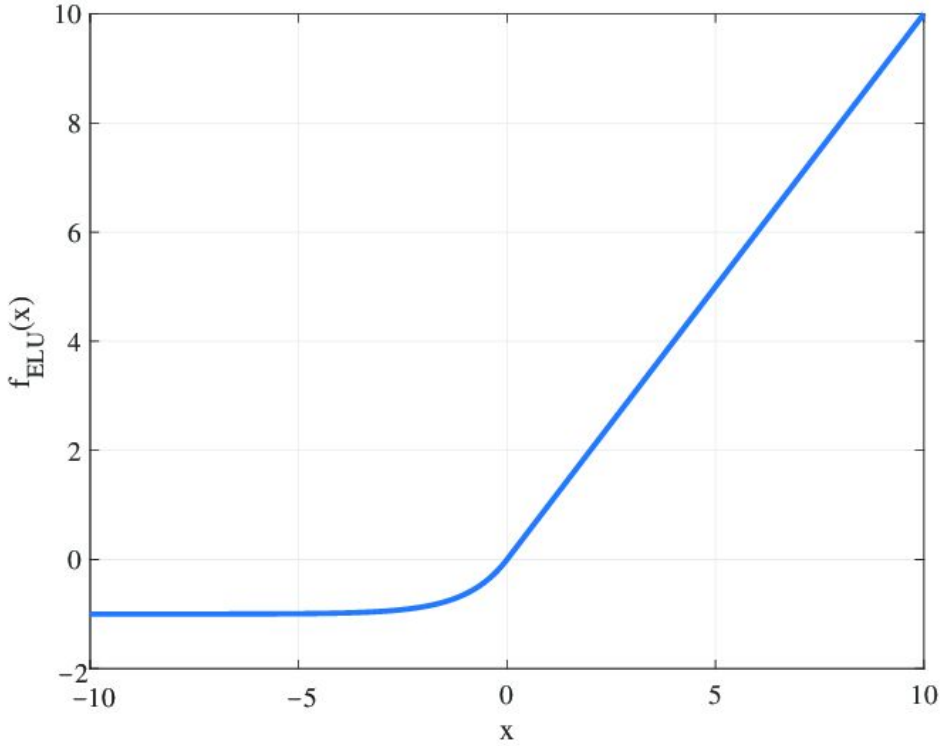


Questão 3

- Arquitetura

```
ClassesNeuralNet(
  (elu): ELU(alpha=1.0)
  (linear1): Linear(in_features=2, out_features=256, bias=True)
  (linear2): Linear(in_features=256, out_features=256, bias=True)
  (linear3): Linear(in_features=256, out_features=64, bias=True)
  (linear4): Linear(in_features=64, out_features=16, bias=True)
  (linear5): Linear(in_features=16, out_features=8, bias=True)
)
```

Questão 3



Questão 3

a) **Sem momentum**

loss: Cross Entropy / **Optimizer:** SGD / **Learning rate:** 0.005

Batch size: 128 / **N_Epochs:** 1000

Acurácia:

```
-------------------------------
loss: 1.434185  [  128/13796]
loss: 1.413840  [12928/13796]
Test Error:
 Accuracy: 83.7%, Avg loss: 1.440140
```

Questão 3

a) **Com momentum (momentum = 0.5)**

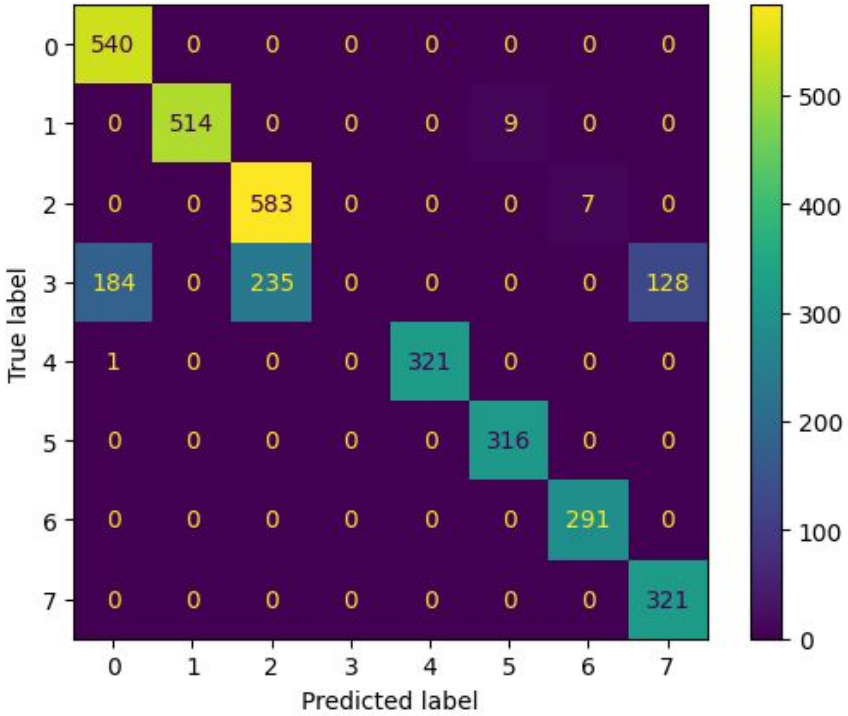
loss: Cross Entropy / **Optimizer:** SGD / **Learning rate:** 0.005

Batch size: 128 / **N_Epochs:** 1000

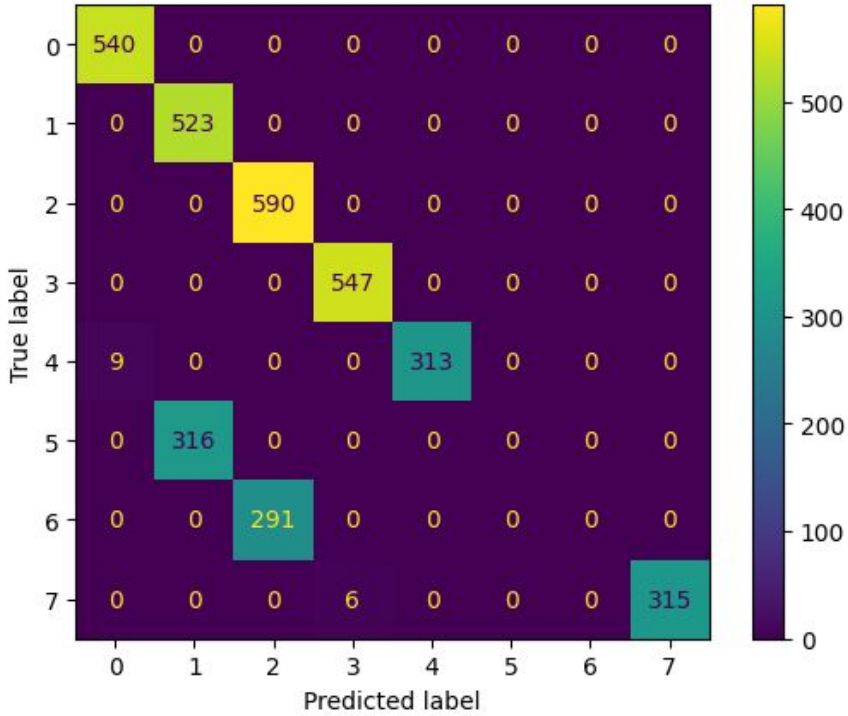
Acurácia:

```
--------------------------------
loss: 1.446448  [  128/13796]
loss: 1.425820  [12928/13796]
Test Error:
 Accuracy: 82.0%, Avg loss: 1.454916
```

Questão 3



Sem momentum



Com momentum

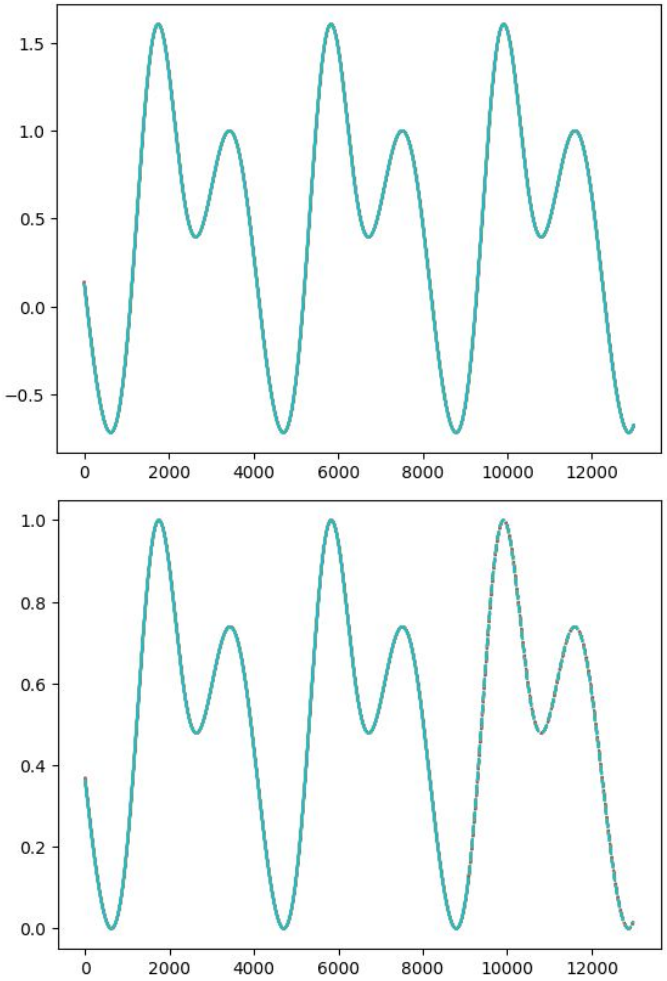
Questão 4

- Dataset
  - número de exemplos: 12988, divisão 70/30

x(n) = sen^2(n) + cos(n+cos(n)).

k = 10

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$



Questão 4

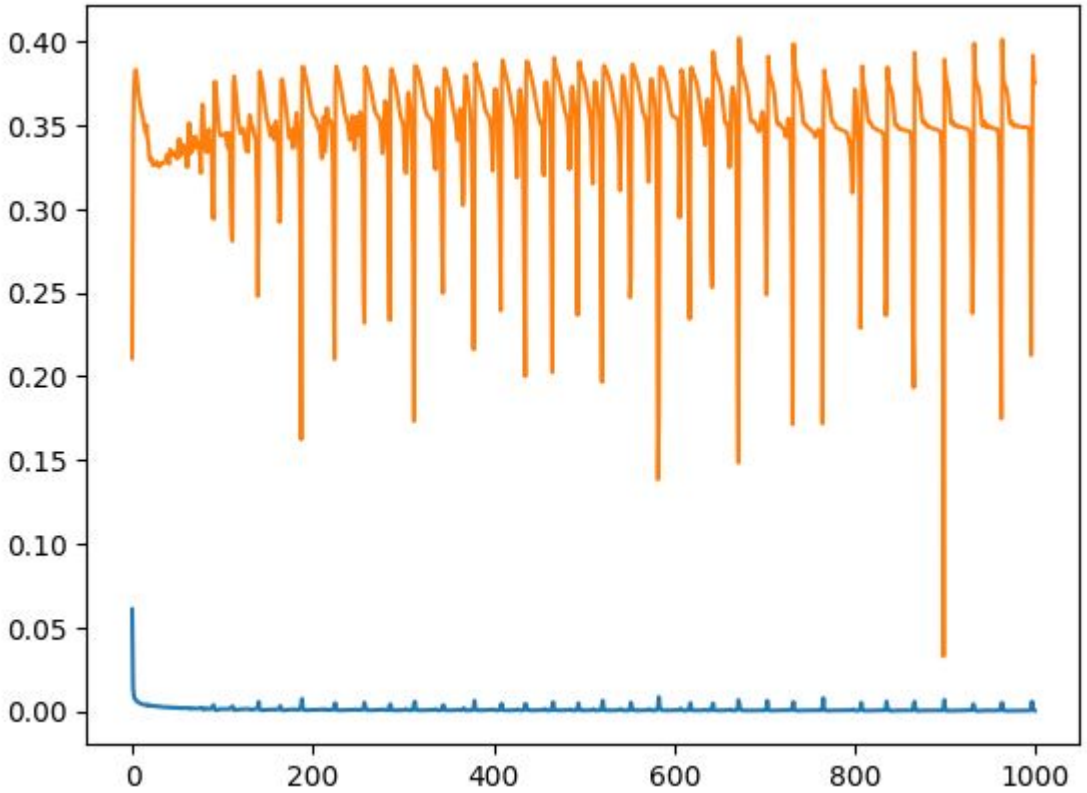
- Arquitetura

  loss: MSELoss / **Optimizer:** Adam / **Learning rate:** 0.001

  Batch size: 64 / **N_Epochs:** 1000

```
TemporalSeriesNeuralNet(
  (linear_layers): Sequential(
    (0): Linear(in_features=10, out_features=32, bias=True)
    (1): Tanh()
    (2): Linear(in_features=32, out_features=64, bias=True)
    (3): Tanh()
    (4): Linear(in_features=64, out_features=64, bias=True)
    (5): Tanh()
    (6): Linear(in_features=64, out_features=32, bias=True)
    (7): Tanh()
    (8): Linear(in_features=32, out_features=3, bias=True)
  )
)
```

Questão 4



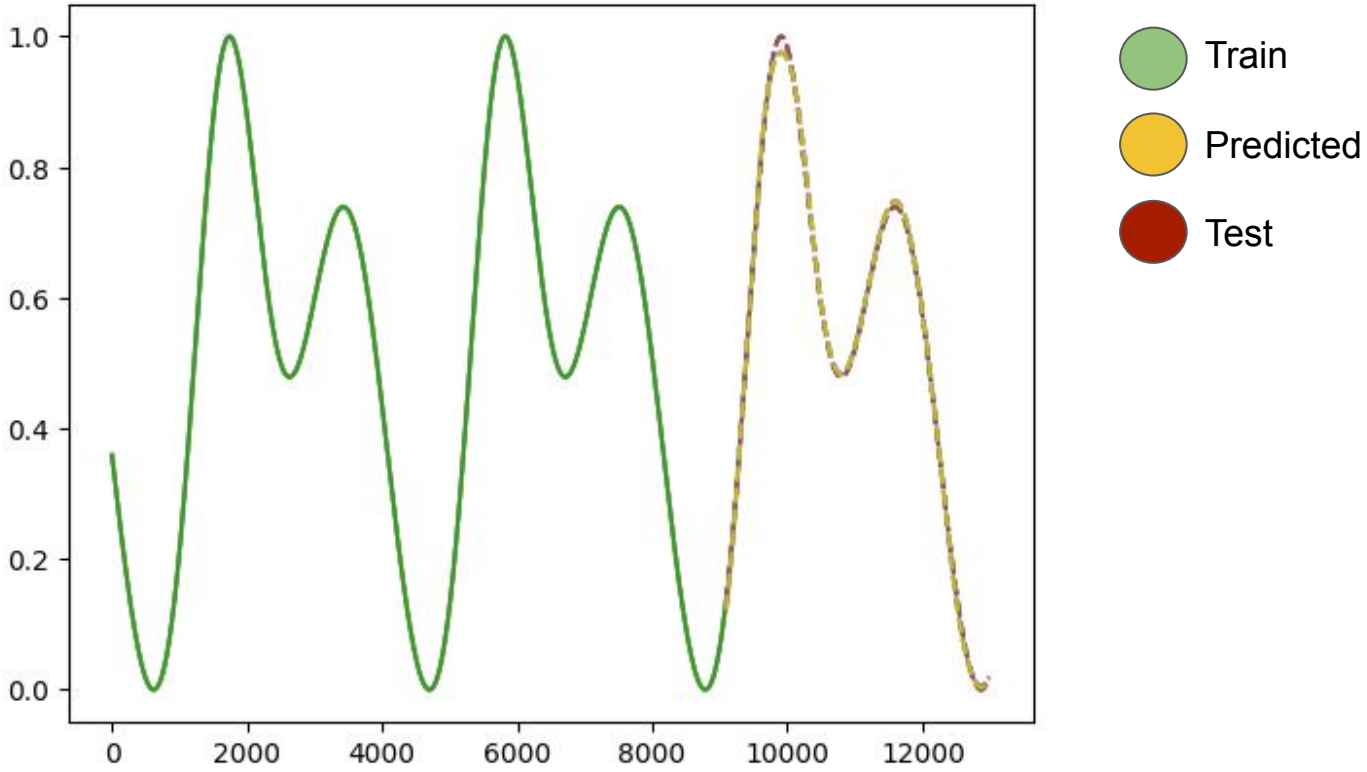
Train

Test

```
--------------------------------
loss: 0.000015  [   64/ 9091]
loss: 0.000017  [ 6464/ 9091]
Train Error:
 Avg loss: 0.000121

loss: 0.023524  [   64/ 3897]
Test Error:
 Avg loss: 0.375266
```

Questão 4


1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000
12000
Train
Predicted
Test