fase3_2

May 21, 2025

[1]:
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Dense, Input
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt
```

```
2025-05-21 06:18:46.172976: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32]
Could not find cuda drivers on your machine, GPU will not be used.
2025-05-21 06:18:46.242654: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32]
Could not find cuda drivers on your machine, GPU will not be used.
2025-05-21 06:18:46.314646: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:467] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
E0000 00:00:1747822726.375491   71837 cuda_dnn.cc:8579] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1747822726.388153   71837 cuda_blas.cc:1407] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
W0000 00:00:1747822726.444700   71837 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747822726.444732   71837 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747822726.444735   71837 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747822726.444737   71837 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
2025-05-21 06:18:46.452968: I tensorflow/core/platform/cpu_feature_guard.cc:210]
```

```
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild
TensorFlow with the appropriate compiler flags.
```

[2]:
```
# ================================
# 1. Carregar dados reais e gerados
# ================================
df_all = pd.read_csv("/home/darkcover/Documentos/Gan/Data/df_all.csv")
df_generated = pd.read_csv("/home/darkcover/Documentos/Gan/Data/df_generated.
 ↪csv")

df_real = df_all[df_all["source"] == "real"].copy()
X_real = df_real.iloc[:, :10].values.astype(np.float32)
y_real = df_real[["X", "Y"]].values.astype(np.float32)
```

[3]:
```
# ================================
# 2. Treinar rede DNN para pseudo-rotulação
#   (2 camadas: 30 e 20 neurônios, 250 epochs, batch_size=100)
# ================================
n_features = X_real.shape[1]
inp = Input(shape=(n_features,))
x = Dense(30, activation='relu')(inp)
x = Dense(20, activation='relu')(x)
out = Dense(2, activation='linear')(x) # Saida linear para as coordernadas X e Y
model_dnn = Model(inputs=inp, outputs=out, name="PseudoLabelModel")

model_dnn.compile(optimizer=Adam(learning_rate=0.01), loss='mse')
X_train, X_val, y_train, y_val = train_test_split(X_real, y_real, test_size=0.
 ↪2, random_state=42)

model_dnn.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=250,␣
 ↪batch_size=100, verbose=1)
# Salvar o modelo
model_dnn.save("/home/darkcover/Documentos/Gan/Models/pseudo_label_model.keras")
```

```
Epoch 1/250
```

```
2025-05-21 06:18:52.478399: E
external/local_xla/xla/stream_executor/cuda/cuda_platform.cc:51] failed call to
cuInit: INTERNAL: CUDA error: Failed call to cuInit: UNKNOWN ERROR (303)
```

```
8/8              2s 39ms/step - loss:
133.8811 - val_loss: 52.1939
Epoch 2/250
8/8              0s 17ms/step - loss:
45.0885 - val_loss: 30.1341
Epoch 3/250
8/8              0s 14ms/step - loss:
```

```
29.1097 - val_loss: 25.1584
Epoch 4/250
8/8              0s 14ms/step - loss:
24.7247 - val_loss: 21.5194
Epoch 5/250
8/8              0s 14ms/step - loss:
20.3516 - val_loss: 19.4714
Epoch 6/250
8/8              0s 16ms/step - loss:
19.9057 - val_loss: 19.5458
Epoch 7/250
8/8              0s 14ms/step - loss:
18.6641 - val_loss: 18.2087
Epoch 8/250
8/8              0s 15ms/step - loss:
17.1468 - val_loss: 16.7954
Epoch 9/250
8/8              0s 15ms/step - loss:
16.3092 - val_loss: 15.2981
Epoch 10/250
8/8              0s 17ms/step - loss:
14.1954 - val_loss: 13.5237
Epoch 11/250
8/8              0s 25ms/step - loss:
12.5333 - val_loss: 11.4291
Epoch 12/250
8/8              0s 15ms/step - loss:
10.3772 - val_loss: 8.3082
Epoch 13/250
8/8              0s 15ms/step - loss:
7.4332 - val_loss: 7.0585
Epoch 14/250
8/8              0s 14ms/step - loss:
6.2966 - val_loss: 5.4895
Epoch 15/250
8/8              0s 18ms/step - loss:
5.2360 - val_loss: 4.7415
Epoch 16/250
8/8              0s 18ms/step - loss:
4.6527 - val_loss: 4.1346
Epoch 17/250
8/8              0s 18ms/step - loss:
4.3864 - val_loss: 4.1962
Epoch 18/250
8/8              0s 15ms/step - loss:
4.1448 - val_loss: 4.1320
Epoch 19/250
8/8              0s 15ms/step - loss:
```

```
4.0501 - val_loss: 3.7729
Epoch 20/250
8/8           0s 14ms/step - loss:
3.7609 - val_loss: 3.8127
Epoch 21/250
8/8           0s 14ms/step - loss:
3.9557 - val_loss: 4.3563
Epoch 22/250
8/8           0s 15ms/step - loss:
4.1789 - val_loss: 3.5662
Epoch 23/250
8/8           0s 15ms/step - loss:
3.9194 - val_loss: 3.8500
Epoch 24/250
8/8           0s 15ms/step - loss:
3.8097 - val_loss: 3.6200
Epoch 25/250
8/8           0s 14ms/step - loss:
3.6054 - val_loss: 3.3886
Epoch 26/250
8/8           0s 15ms/step - loss:
3.4488 - val_loss: 3.4352
Epoch 27/250
8/8           0s 15ms/step - loss:
3.7341 - val_loss: 3.2434
Epoch 28/250
8/8           0s 15ms/step - loss:
3.9627 - val_loss: 3.4234
Epoch 29/250
8/8           0s 16ms/step - loss:
3.6423 - val_loss: 3.6637
Epoch 30/250
8/8           0s 19ms/step - loss:
4.0640 - val_loss: 3.4194
Epoch 31/250
8/8           0s 15ms/step - loss:
3.8133 - val_loss: 3.7245
Epoch 32/250
8/8           0s 15ms/step - loss:
3.5660 - val_loss: 3.5398
Epoch 33/250
8/8           0s 15ms/step - loss:
3.6852 - val_loss: 3.8603
Epoch 34/250
8/8           0s 15ms/step - loss:
3.5386 - val_loss: 2.9968
Epoch 35/250
8/8           0s 16ms/step - loss:
```

```
3.2490 - val_loss: 3.0006
Epoch 36/250
8/8              0s 14ms/step - loss:
3.1100 - val_loss: 2.9563
Epoch 37/250
8/8              0s 14ms/step - loss:
3.1023 - val_loss: 3.0147
Epoch 38/250
8/8              0s 15ms/step - loss:
3.2604 - val_loss: 2.9437
Epoch 39/250
8/8              0s 16ms/step - loss:
3.0402 - val_loss: 3.1983
Epoch 40/250
8/8              0s 15ms/step - loss:
3.3901 - val_loss: 3.2182
Epoch 41/250
8/8              0s 14ms/step - loss:
3.4834 - val_loss: 3.3229
Epoch 42/250
8/8              0s 14ms/step - loss:
3.6670 - val_loss: 3.3890
Epoch 43/250
8/8              0s 17ms/step - loss:
3.1728 - val_loss: 3.0115
Epoch 44/250
8/8              0s 14ms/step - loss:
3.1695 - val_loss: 2.8246
Epoch 45/250
8/8              0s 15ms/step - loss:
3.1020 - val_loss: 3.5199
Epoch 46/250
8/8              0s 16ms/step - loss:
3.4422 - val_loss: 2.8131
Epoch 47/250
8/8              0s 15ms/step - loss:
3.1246 - val_loss: 2.9963
Epoch 48/250
8/8              0s 15ms/step - loss:
3.2543 - val_loss: 2.8990
Epoch 49/250
8/8              0s 15ms/step - loss:
2.8502 - val_loss: 2.7519
Epoch 50/250
8/8              0s 15ms/step - loss:
3.0070 - val_loss: 2.9049
Epoch 51/250
8/8              0s 15ms/step - loss:
```

```
3.0415 - val_loss: 2.8176
Epoch 52/250
8/8             0s 16ms/step - loss:
3.2398 - val_loss: 2.7138
Epoch 53/250
8/8             0s 15ms/step - loss:
3.0084 - val_loss: 2.7137
Epoch 54/250
8/8             0s 15ms/step - loss:
2.9122 - val_loss: 2.6769
Epoch 55/250
8/8             0s 25ms/step - loss:
2.8480 - val_loss: 2.6250
Epoch 56/250
8/8             0s 15ms/step - loss:
2.8643 - val_loss: 2.9124
Epoch 57/250
8/8             0s 14ms/step - loss:
3.1533 - val_loss: 2.6753
Epoch 58/250
8/8             0s 15ms/step - loss:
2.7935 - val_loss: 2.7164
Epoch 59/250
8/8             0s 15ms/step - loss:
2.8378 - val_loss: 2.7678
Epoch 60/250
8/8             0s 15ms/step - loss:
2.8387 - val_loss: 2.7057
Epoch 61/250
8/8             0s 15ms/step - loss:
2.9000 - val_loss: 2.7820
Epoch 62/250
8/8             0s 15ms/step - loss:
2.8145 - val_loss: 2.7769
Epoch 63/250
8/8             0s 15ms/step - loss:
2.9153 - val_loss: 2.5503
Epoch 64/250
8/8             0s 15ms/step - loss:
2.6939 - val_loss: 2.7886
Epoch 65/250
8/8             0s 15ms/step - loss:
3.0388 - val_loss: 2.8880
Epoch 66/250
8/8             0s 16ms/step - loss:
3.3053 - val_loss: 3.1659
Epoch 67/250
8/8             0s 17ms/step - loss:
```

```
3.1692 - val_loss: 2.7650
Epoch 68/250
8/8             0s 16ms/step - loss:
3.2202 - val_loss: 3.4634
Epoch 69/250
8/8             0s 16ms/step - loss:
3.1496 - val_loss: 2.9759
Epoch 70/250
8/8             0s 15ms/step - loss:
2.9801 - val_loss: 2.9781
Epoch 71/250
8/8             0s 15ms/step - loss:
3.4709 - val_loss: 3.2751
Epoch 72/250
8/8             0s 20ms/step - loss:
3.5249 - val_loss: 3.1491
Epoch 73/250
8/8             0s 17ms/step - loss:
3.3729 - val_loss: 2.8517
Epoch 74/250
8/8             0s 15ms/step - loss:
2.8752 - val_loss: 2.6271
Epoch 75/250
8/8             0s 15ms/step - loss:
2.9935 - val_loss: 3.0860
Epoch 76/250
8/8             0s 16ms/step - loss:
3.3299 - val_loss: 2.7026
Epoch 77/250
8/8             0s 16ms/step - loss:
2.8216 - val_loss: 2.6080
Epoch 78/250
8/8             0s 16ms/step - loss:
2.6123 - val_loss: 2.3140
Epoch 79/250
8/8             0s 15ms/step - loss:
2.4313 - val_loss: 2.4474
Epoch 80/250
8/8             0s 17ms/step - loss:
2.7892 - val_loss: 2.9582
Epoch 81/250
8/8             0s 15ms/step - loss:
2.7081 - val_loss: 2.2777
Epoch 82/250
8/8             0s 15ms/step - loss:
2.4439 - val_loss: 2.6390
Epoch 83/250
8/8             0s 15ms/step - loss:
```

```
2.4496 - val_loss: 2.2799
Epoch 84/250
8/8           0s 15ms/step - loss:
2.5781 - val_loss: 2.3142
Epoch 85/250
8/8           0s 15ms/step - loss:
2.4632 - val_loss: 2.3451
Epoch 86/250
8/8           0s 15ms/step - loss:
2.3574 - val_loss: 2.3041
Epoch 87/250
8/8           0s 15ms/step - loss:
2.3513 - val_loss: 2.4002
Epoch 88/250
8/8           0s 15ms/step - loss:
2.2942 - val_loss: 2.4951
Epoch 89/250
8/8           0s 15ms/step - loss:
2.4508 - val_loss: 3.1408
Epoch 90/250
8/8           0s 15ms/step - loss:
2.8635 - val_loss: 2.7328
Epoch 91/250
8/8           0s 15ms/step - loss:
2.7765 - val_loss: 3.0366
Epoch 92/250
8/8           0s 15ms/step - loss:
2.7186 - val_loss: 2.2757
Epoch 93/250
8/8           0s 16ms/step - loss:
2.4527 - val_loss: 2.2182
Epoch 94/250
8/8           0s 36ms/step - loss:
2.3859 - val_loss: 2.5397
Epoch 95/250
8/8           0s 15ms/step - loss:
2.3482 - val_loss: 2.3849
Epoch 96/250
8/8           0s 14ms/step - loss:
2.3636 - val_loss: 2.8372
Epoch 97/250
8/8           0s 15ms/step - loss:
2.5472 - val_loss: 2.7859
Epoch 98/250
8/8           0s 15ms/step - loss:
2.5865 - val_loss: 2.2550
Epoch 99/250
8/8           0s 15ms/step - loss:
```

```
2.4725 - val_loss: 2.8665
Epoch 100/250
8/8             0s 15ms/step - loss:
2.7647 - val_loss: 3.2878
Epoch 101/250
8/8             0s 15ms/step - loss:
3.1143 - val_loss: 2.6184
Epoch 102/250
8/8             0s 15ms/step - loss:
2.6167 - val_loss: 2.7686
Epoch 103/250
8/8             0s 15ms/step - loss:
2.4154 - val_loss: 2.4003
Epoch 104/250
8/8             0s 15ms/step - loss:
2.8086 - val_loss: 3.0125
Epoch 105/250
8/8             0s 17ms/step - loss:
3.1747 - val_loss: 2.5525
Epoch 106/250
8/8             0s 24ms/step - loss:
2.7303 - val_loss: 2.3423
Epoch 107/250
8/8             0s 15ms/step - loss:
2.5287 - val_loss: 2.3611
Epoch 108/250
8/8             0s 15ms/step - loss:
2.1934 - val_loss: 2.2023
Epoch 109/250
8/8             0s 15ms/step - loss:
2.1344 - val_loss: 2.3846
Epoch 110/250
8/8             0s 15ms/step - loss:
2.1862 - val_loss: 2.1273
Epoch 111/250
8/8             0s 15ms/step - loss:
2.3021 - val_loss: 2.3445
Epoch 112/250
8/8             0s 15ms/step - loss:
2.2276 - val_loss: 2.3902
Epoch 113/250
8/8             0s 16ms/step - loss:
2.4564 - val_loss: 2.4122
Epoch 114/250
8/8             0s 15ms/step - loss:
2.2914 - val_loss: 2.2155
Epoch 115/250
8/8             0s 15ms/step - loss:
```

```
2.5639 - val_loss: 2.6665
Epoch 116/250
8/8            0s 17ms/step - loss:
2.3967 - val_loss: 2.2370
Epoch 117/250
8/8            0s 16ms/step - loss:
2.4671 - val_loss: 2.6120
Epoch 118/250
8/8            0s 15ms/step - loss:
2.5114 - val_loss: 2.8911
Epoch 119/250
8/8            0s 21ms/step - loss:
2.4806 - val_loss: 2.1733
Epoch 120/250
8/8            0s 16ms/step - loss:
2.1490 - val_loss: 2.2773
Epoch 121/250
8/8            0s 16ms/step - loss:
2.1459 - val_loss: 2.1175
Epoch 122/250
8/8            0s 17ms/step - loss:
2.2354 - val_loss: 2.1822
Epoch 123/250
8/8            0s 19ms/step - loss:
2.2716 - val_loss: 2.3165
Epoch 124/250
8/8            0s 21ms/step - loss:
2.2585 - val_loss: 2.1405
Epoch 125/250
8/8            0s 18ms/step - loss:
2.2006 - val_loss: 2.0549
Epoch 126/250
8/8            0s 21ms/step - loss:
2.1724 - val_loss: 2.0546
Epoch 127/250
8/8            0s 19ms/step - loss:
2.1177 - val_loss: 2.0977
Epoch 128/250
8/8            0s 16ms/step - loss:
2.2522 - val_loss: 2.2346
Epoch 129/250
8/8            0s 16ms/step - loss:
2.0559 - val_loss: 2.2596
Epoch 130/250
8/8            0s 16ms/step - loss:
2.2754 - val_loss: 1.9973
Epoch 131/250
8/8            0s 16ms/step - loss:
```

```
2.0061 - val_loss: 2.0208
Epoch 132/250
8/8              0s 16ms/step - loss:
2.2095 - val_loss: 2.1245
Epoch 133/250
8/8              0s 15ms/step - loss:
2.3452 - val_loss: 2.5489
Epoch 134/250
8/8              0s 22ms/step - loss:
2.4218 - val_loss: 2.8693
Epoch 135/250
8/8              0s 34ms/step - loss:
2.3475 - val_loss: 2.3560
Epoch 136/250
8/8              0s 20ms/step - loss:
2.0606 - val_loss: 2.1651
Epoch 137/250
8/8              0s 20ms/step - loss:
2.2297 - val_loss: 2.3877
Epoch 138/250
8/8              0s 19ms/step - loss:
2.5390 - val_loss: 2.0791
Epoch 139/250
8/8              0s 17ms/step - loss:
2.0193 - val_loss: 2.0333
Epoch 140/250
8/8              0s 19ms/step - loss:
2.0360 - val_loss: 2.0258
Epoch 141/250
8/8              0s 16ms/step - loss:
2.0237 - val_loss: 2.2923
Epoch 142/250
8/8              0s 16ms/step - loss:
2.0782 - val_loss: 2.0150
Epoch 143/250
8/8              0s 18ms/step - loss:
2.0200 - val_loss: 2.0762
Epoch 144/250
8/8              0s 20ms/step - loss:
2.3052 - val_loss: 1.9834
Epoch 145/250
8/8              0s 17ms/step - loss:
2.1333 - val_loss: 2.2920
Epoch 146/250
8/8              0s 16ms/step - loss:
2.1528 - val_loss: 2.1038
Epoch 147/250
8/8              0s 17ms/step - loss:
```

```
2.3095 - val_loss: 3.6688
Epoch 148/250
8/8            0s 15ms/step - loss:
3.4582 - val_loss: 2.6040
Epoch 149/250
8/8            0s 16ms/step - loss:
2.6092 - val_loss: 2.0808
Epoch 150/250
8/8            0s 16ms/step - loss:
2.2795 - val_loss: 2.5557
Epoch 151/250
8/8            0s 23ms/step - loss:
2.4458 - val_loss: 3.1971
Epoch 152/250
8/8            0s 15ms/step - loss:
2.5088 - val_loss: 2.0173
Epoch 153/250
8/8            0s 16ms/step - loss:
2.1214 - val_loss: 2.3774
Epoch 154/250
8/8            0s 17ms/step - loss:
2.2463 - val_loss: 1.9903
Epoch 155/250
8/8            0s 16ms/step - loss:
2.1886 - val_loss: 2.6922
Epoch 156/250
8/8            0s 16ms/step - loss:
2.5490 - val_loss: 3.3028
Epoch 157/250
8/8            0s 23ms/step - loss:
2.6140 - val_loss: 2.1544
Epoch 158/250
8/8            0s 16ms/step - loss:
2.1250 - val_loss: 2.2972
Epoch 159/250
8/8            0s 19ms/step - loss:
2.2399 - val_loss: 1.9217
Epoch 160/250
8/8            0s 14ms/step - loss:
1.9731 - val_loss: 2.0488
Epoch 161/250
8/8            0s 13ms/step - loss:
1.9508 - val_loss: 2.1416
Epoch 162/250
8/8            0s 22ms/step - loss:
1.9807 - val_loss: 1.9648
Epoch 163/250
8/8            0s 18ms/step - loss:
```

```
1.9059 - val_loss: 1.8629
Epoch 164/250
8/8             0s 16ms/step - loss:
1.9185 - val_loss: 2.5535
Epoch 165/250
8/8             0s 15ms/step - loss:
2.0807 - val_loss: 2.1633
Epoch 166/250
8/8             0s 21ms/step - loss:
2.0340 - val_loss: 2.1930
Epoch 167/250
8/8             0s 17ms/step - loss:
2.3838 - val_loss: 2.7657
Epoch 168/250
8/8             0s 22ms/step - loss:
2.7083 - val_loss: 2.8500
Epoch 169/250
8/8             0s 37ms/step - loss:
2.6081 - val_loss: 2.0968
Epoch 170/250
8/8             0s 27ms/step - loss:
2.5757 - val_loss: 2.6697
Epoch 171/250
8/8             0s 36ms/step - loss:
2.5744 - val_loss: 2.2359
Epoch 172/250
8/8             1s 22ms/step - loss:
2.2776 - val_loss: 1.8830
Epoch 173/250
8/8             0s 24ms/step - loss:
2.0733 - val_loss: 1.9517
Epoch 174/250
8/8             0s 19ms/step - loss:
2.0446 - val_loss: 3.0145
Epoch 175/250
8/8             0s 21ms/step - loss:
2.3974 - val_loss: 1.8714
Epoch 176/250
8/8             0s 23ms/step - loss:
2.0932 - val_loss: 2.4061
Epoch 177/250
8/8             0s 19ms/step - loss:
2.7380 - val_loss: 2.1415
Epoch 178/250
8/8             0s 17ms/step - loss:
2.4226 - val_loss: 2.2198
Epoch 179/250
8/8             0s 19ms/step - loss:
```

```
2.2357 - val_loss: 2.1870
Epoch 180/250
8/8              0s 24ms/step - loss:
1.8853 - val_loss: 1.9866
Epoch 181/250
8/8              0s 17ms/step - loss:
2.0391 - val_loss: 1.9108
Epoch 182/250
8/8              0s 25ms/step - loss:
1.9971 - val_loss: 2.6271
Epoch 183/250
8/8              0s 22ms/step - loss:
2.5568 - val_loss: 2.1946
Epoch 184/250
8/8              0s 23ms/step - loss:
2.2946 - val_loss: 2.0206
Epoch 185/250
8/8              0s 18ms/step - loss:
2.0192 - val_loss: 2.7380
Epoch 186/250
8/8              0s 12ms/step - loss:
2.0354 - val_loss: 1.9474
Epoch 187/250
8/8              0s 22ms/step - loss:
1.9561 - val_loss: 2.0841
Epoch 188/250
8/8              0s 12ms/step - loss:
2.0167 - val_loss: 1.8449
Epoch 189/250
8/8              0s 17ms/step - loss:
1.9350 - val_loss: 1.9616
Epoch 190/250
8/8              0s 14ms/step - loss:
1.9390 - val_loss: 2.1793
Epoch 191/250
8/8              0s 11ms/step - loss:
2.0854 - val_loss: 2.5630
Epoch 192/250
8/8              0s 17ms/step - loss:
2.2857 - val_loss: 2.0940
Epoch 193/250
8/8              0s 17ms/step - loss:
2.1172 - val_loss: 2.3004
Epoch 194/250
8/8              0s 11ms/step - loss:
2.4419 - val_loss: 3.0075
Epoch 195/250
8/8              0s 20ms/step - loss:
```

```
2.1687 - val_loss: 1.8772
Epoch 196/250
8/8             0s 21ms/step - loss:
1.9090 - val_loss: 1.9455
Epoch 197/250
8/8             0s 28ms/step - loss:
1.8642 - val_loss: 1.7880
Epoch 198/250
8/8             0s 12ms/step - loss:
1.9249 - val_loss: 1.8295
Epoch 199/250
8/8             0s 20ms/step - loss:
1.6971 - val_loss: 2.2537
Epoch 200/250
8/8             0s 12ms/step - loss:
1.8009 - val_loss: 1.8748
Epoch 201/250
8/8             0s 22ms/step - loss:
1.9030 - val_loss: 1.8565
Epoch 202/250
8/8             0s 19ms/step - loss:
1.7173 - val_loss: 1.8081
Epoch 203/250
8/8             0s 17ms/step - loss:
1.7861 - val_loss: 1.8802
Epoch 204/250
8/8             0s 16ms/step - loss:
1.8111 - val_loss: 2.1462
Epoch 205/250
8/8             0s 12ms/step - loss:
1.9516 - val_loss: 2.2679
Epoch 206/250
8/8             0s 11ms/step - loss:
2.0555 - val_loss: 1.8252
Epoch 207/250
8/8             0s 11ms/step - loss:
1.8050 - val_loss: 1.9498
Epoch 208/250
8/8             0s 12ms/step - loss:
1.7644 - val_loss: 2.0946
Epoch 209/250
8/8             0s 13ms/step - loss:
1.7379 - val_loss: 1.7218
Epoch 210/250
8/8             0s 15ms/step - loss:
1.6998 - val_loss: 2.3824
Epoch 211/250
8/8             0s 14ms/step - loss:
```

```
2.1594 - val_loss: 1.7795
Epoch 212/250
8/8              0s 14ms/step - loss:
1.8142 - val_loss: 1.9619
Epoch 213/250
8/8              0s 15ms/step - loss:
1.8491 - val_loss: 2.0030
Epoch 214/250
8/8              0s 15ms/step - loss:
1.8298 - val_loss: 1.9685
Epoch 215/250
8/8              0s 15ms/step - loss:
1.9983 - val_loss: 2.7583
Epoch 216/250
8/8              0s 13ms/step - loss:
2.3298 - val_loss: 2.2541
Epoch 217/250
8/8              0s 13ms/step - loss:
2.0355 - val_loss: 1.9085
Epoch 218/250
8/8              0s 15ms/step - loss:
1.8873 - val_loss: 1.8727
Epoch 219/250
8/8              0s 15ms/step - loss:
2.0614 - val_loss: 2.0371
Epoch 220/250
8/8              0s 16ms/step - loss:
2.3375 - val_loss: 1.9363
Epoch 221/250
8/8              0s 35ms/step - loss:
1.9390 - val_loss: 1.8687
Epoch 222/250
8/8              0s 25ms/step - loss:
1.8369 - val_loss: 2.1309
Epoch 223/250
8/8              0s 22ms/step - loss:
1.9380 - val_loss: 1.8923
Epoch 224/250
8/8              0s 20ms/step - loss:
1.7441 - val_loss: 1.8196
Epoch 225/250
8/8              0s 19ms/step - loss:
1.7984 - val_loss: 1.9269
Epoch 226/250
8/8              0s 20ms/step - loss:
1.8829 - val_loss: 1.6906
Epoch 227/250
8/8              0s 20ms/step - loss:
```

```
1.9155 - val_loss: 1.8462
Epoch 228/250
8/8             0s 20ms/step - loss:
1.8636 - val_loss: 2.0192
Epoch 229/250
8/8             0s 20ms/step - loss:
1.7751 - val_loss: 1.9468
Epoch 230/250
8/8             0s 24ms/step - loss:
1.8483 - val_loss: 1.8115
Epoch 231/250
8/8             0s 20ms/step - loss:
1.7196 - val_loss: 2.0340
Epoch 232/250
8/8             0s 17ms/step - loss:
1.8220 - val_loss: 1.6937
Epoch 233/250
8/8             0s 18ms/step - loss:
2.1151 - val_loss: 1.6636
Epoch 234/250
8/8             0s 22ms/step - loss:
1.8440 - val_loss: 1.8543
Epoch 235/250
8/8             0s 20ms/step - loss:
1.7871 - val_loss: 1.8088
Epoch 236/250
8/8             0s 20ms/step - loss:
1.8090 - val_loss: 1.6655
Epoch 237/250
8/8             0s 20ms/step - loss:
1.7918 - val_loss: 2.0376
Epoch 238/250
8/8             0s 20ms/step - loss:
1.8039 - val_loss: 1.7293
Epoch 239/250
8/8             0s 21ms/step - loss:
1.6135 - val_loss: 1.6974
Epoch 240/250
8/8             0s 20ms/step - loss:
1.6669 - val_loss: 1.7514
Epoch 241/250
8/8             0s 19ms/step - loss:
1.6508 - val_loss: 1.6159
Epoch 242/250
8/8             0s 20ms/step - loss:
1.7076 - val_loss: 1.6625
Epoch 243/250
8/8             0s 20ms/step - loss:
```

```
2.1536 - val_loss: 3.1298
Epoch 244/250
8/8              0s 19ms/step - loss:
2.9045 - val_loss: 2.4336
Epoch 245/250
8/8              0s 19ms/step - loss:
2.3632 - val_loss: 2.2731
Epoch 246/250
8/8              0s 21ms/step - loss:
2.2189 - val_loss: 2.7511
Epoch 247/250
8/8              0s 24ms/step - loss:
2.3456 - val_loss: 2.1147
Epoch 248/250
8/8              0s 20ms/step - loss:
1.8764 - val_loss: 1.7843
Epoch 249/250
8/8              0s 19ms/step - loss:
1.7238 - val_loss: 1.8330
Epoch 250/250
8/8              0s 20ms/step - loss:
2.0318 - val_loss: 2.0749
```

[4]:
```
# ================================
# 3. Predição de pseudo-rotulos nos vetores sinteticos
#    (X, Y) para os dados gerados
# ================================
X_gen = df_generated.iloc[:, :10].values.astype(np.float32)
pseudo = model_dnn.predict(X_gen, verbose=1)
df_generated[['X', 'Y']] = pseudo
```

```
1250/1250            2s 2ms/step
```

[5]:
```
# ================================
# 4. Avaliar realismo via Discriminador (critério 2)
#   (Modelo DNN com 2 camadas: 30 e 20 neurônios, 250 epochs, batch_size=100)
# ================================
# Carregar o modelo do discriminador
discriminator = load_model("/home/darkcover/Documentos/Gan/Models/
 ↪Modelsdiscriminator.keras")
d_score = discriminator.predict(X_gen, verbose=1)
df_generated['D_score'] = d_score.flatten()
```

```
1250/1250            2s 2ms/step
```

[6]:
```
# ================================
# 5. Dividir ambiente em zonas de 4m^2 (2x2m) e selecionar 1000 amostras
#    (X, Y) para cada zona
# ================================
L, W = 20, 20
zone_size = 2
nx, ny = int(L / zone_size), int(W / zone_size)
num_zones = nx * ny
ms = 1000
nj = ms // num_zones

# Calcular rotulos de zona
df_generated['zone_x'] = np.minimum((df_generated['X'] // zone_size).
 ↪astype(int), nx - 1)
df_generated['zone_y'] = np.minimum((df_generated['Y'] // zone_size).
 ↪astype(int), ny - 1)
df_generated['zone_id'] = df_generated['zone_x'] + nx * df_generated['zone_y']

# Selecionar top-nj por zona segundo D_score
selected_blocks, zone_logs = [], []
for zone, group in df_generated.groupby('zone_id'):
    topk = group.nlargest(nj, 'D_score')
    selected_blocks.append(topk)
    zone_logs.append((zone, len(group), topk['D_score'].mean()))

df_selected = pd.concat(selected_blocks, ignore_index=True)
df_selected.to_csv("/home/darkcover/Documentos/Gan/Data/df_selected_synthetic.
 ↪csv", index=False)
```

[7]:
```
# ================================
# 6. Logs e Visualizações
# ================================
print("Total sintéticas selecionadas:", len(df_selected))
for zid, total, avg in zone_logs:
    print(f"Zona {zid:03d}: {total} geradas, D_score médio selecionadas {avg:.
 ↪3f}")

L, W        = 20, 20
zone_size   = 2.0

fig, ax = plt.subplots(figsize=(6,6))
# Pontos reais: círculos vazados
ax.scatter(df_real['X'], df_real['Y'],
           facecolors='lightblue', edgecolors='blue',
           s=40, label='Real')

# Pontos sintéticos: xis laranja
ax.scatter(df_selected['X'], df_selected['Y'],
           marker='x', c='#FF6600',
           s=30, label='Sintéticas Selecionadas')
```

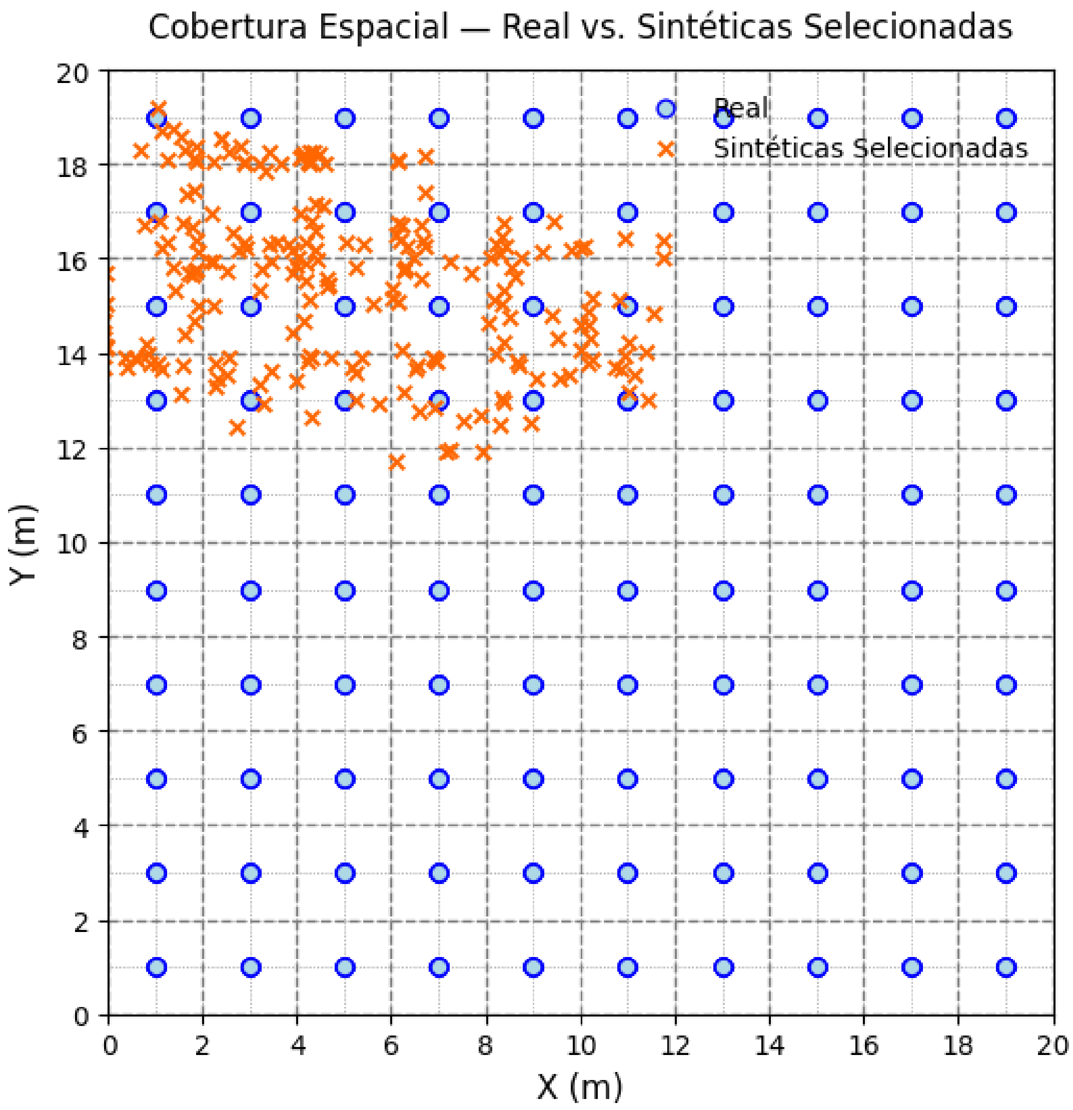
```python
# Desenha grid fino tracejado a cada 1 m
for coord in np.arange(0, L+1, 1):
    ax.axvline(coord, linestyle=':', linewidth=0.5, color='gray', zorder=0)
    ax.axhline(coord, linestyle=':', linewidth=0.5, color='gray', zorder=0)

# Delimitações das zonas (a cada 2 m)
for coord in np.arange(0, L+zone_size, zone_size):
    ax.axvline(coord, linestyle='--', linewidth=1, color='gray', zorder=0)
    ax.axhline(coord, linestyle='--', linewidth=1, color='gray', zorder=0)

ax.set_xlim(0, L)
ax.set_ylim(0, W)
ax.set_aspect('equal', 'box')
ax.set_xticks(np.arange(0, L+1, 2))
ax.set_yticks(np.arange(0, W+1, 2))
ax.set_xlabel("X (m)", fontsize=12)
ax.set_ylabel("Y (m)", fontsize=12)
ax.legend(frameon=False, loc='upper right', fontsize=10)
ax.set_title("Cobertura Espacial - Real vs. Sintéticas Selecionadas", pad=12)

plt.tight_layout()
plt.show()
```

```
Total sintéticas selecionadas: 221
Zona 053: 4 geradas, D_score médio selecionadas 1.000
Zona 059: 1 geradas, D_score médio selecionadas 1.000
Zona 060: 192 geradas, D_score médio selecionadas 1.000
Zona 061: 364 geradas, D_score médio selecionadas 1.000
Zona 062: 609 geradas, D_score médio selecionadas 1.000
Zona 063: 443 geradas, D_score médio selecionadas 1.000
Zona 064: 107 geradas, D_score médio selecionadas 1.000
Zona 065: 8 geradas, D_score médio selecionadas 1.000
Zona 069: 17 geradas, D_score médio selecionadas 1.000
Zona 070: 3106 geradas, D_score médio selecionadas 1.000
Zona 071: 8942 geradas, D_score médio selecionadas 1.000
Zona 072: 4592 geradas, D_score médio selecionadas 1.000
Zona 073: 1692 geradas, D_score médio selecionadas 1.000
Zona 074: 246 geradas, D_score médio selecionadas 1.000
Zona 075: 24 geradas, D_score médio selecionadas 1.000
Zona 080: 2932 geradas, D_score médio selecionadas 1.000
Zona 081: 13035 geradas, D_score médio selecionadas 1.000
Zona 082: 2819 geradas, D_score médio selecionadas 1.000
Zona 083: 473 geradas, D_score médio selecionadas 1.000
Zona 084: 43 geradas, D_score médio selecionadas 1.000
Zona 085: 5 geradas, D_score médio selecionadas 1.000
Zona 090: 93 geradas, D_score médio selecionadas 1.000
Zona 091: 239 geradas, D_score médio selecionadas 1.000
Zona 092: 11 geradas, D_score médio selecionadas 1.000
Zona 093: 3 geradas, D_score médio selecionadas 1.000
```



[9]:
```python
L, W        = 20, 20
zone_size   = 2.0

fig, ax = plt.subplots(figsize=(6,6))
# Pontos reais: círculos vazados
ax.scatter(df_real['X'], df_real['Y'],
           facecolors='lightblue', edgecolors='blue',
           s=40, label='Real')

# Pontos sintéticos: xis laranja
```

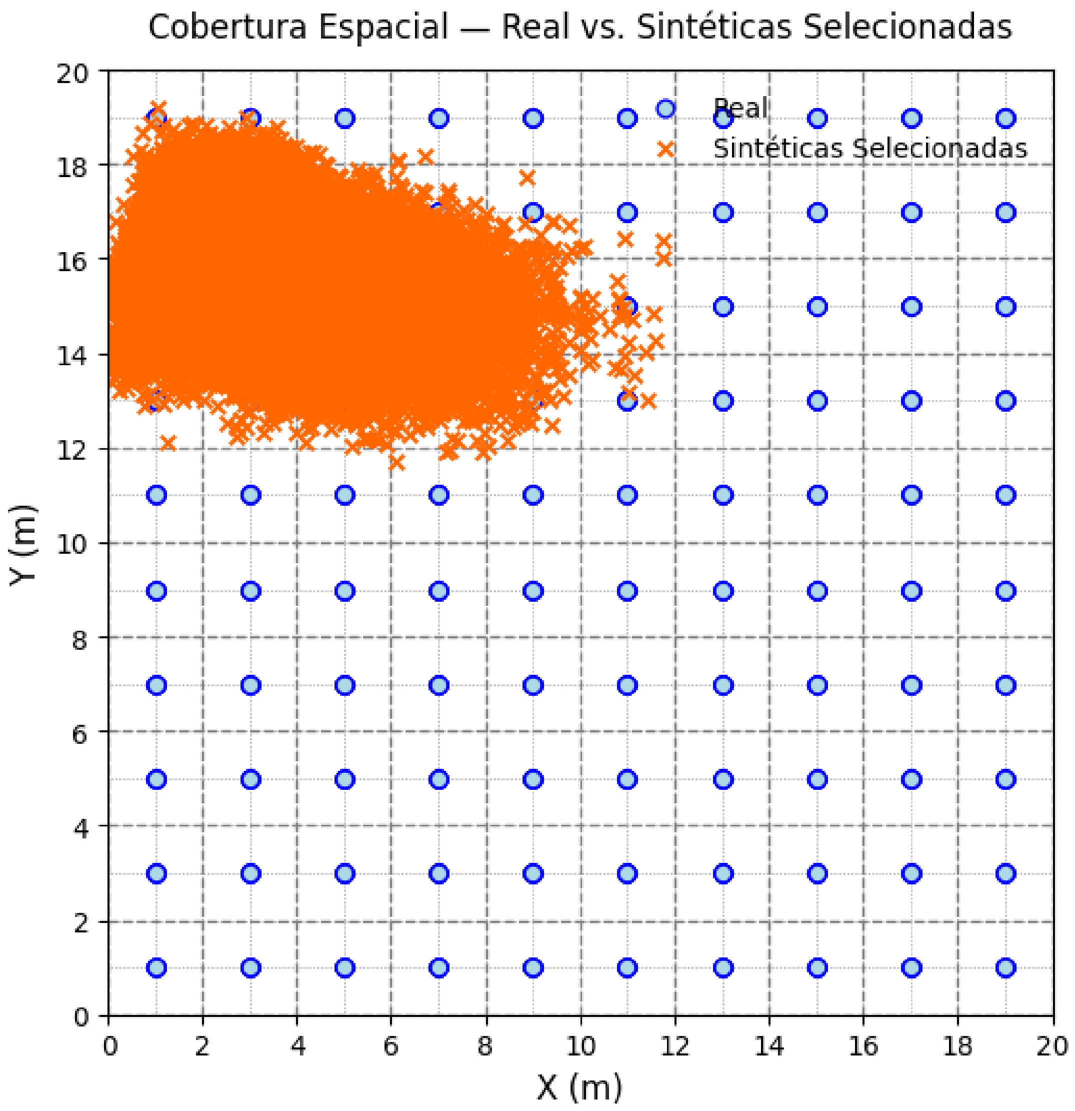
```
ax.scatter(df_generated['X'], df_generated['Y'],
           marker='x', c='#FF6600',
           s=30, label='Sintéticas Selecionadas')

# Desenha grid fino tracejado a cada 1 m
for coord in np.arange(0, L+1, 1):
    ax.axvline(coord, linestyle=':', linewidth=0.5, color='gray', zorder=0)
    ax.axhline(coord, linestyle=':', linewidth=0.5, color='gray', zorder=0)

# Delimitações das zonas (a cada 2 m)
for coord in np.arange(0, L+zone_size, zone_size):
    ax.axvline(coord, linestyle='--', linewidth=1, color='gray', zorder=0)
    ax.axhline(coord, linestyle='--', linewidth=1, color='gray', zorder=0)

ax.set_xlim(0, L)
ax.set_ylim(0, W)
ax.set_aspect('equal', 'box')
ax.set_xticks(np.arange(0, L+1, 2))
ax.set_yticks(np.arange(0, W+1, 2))
ax.set_xlabel("X (m)", fontsize=12)
ax.set_ylabel("Y (m)", fontsize=12)
ax.legend(frameon=False, loc='upper right', fontsize=10)
ax.set_title("Cobertura Espacial - Real vs. Sintéticas Selecionadas", pad=12)

plt.tight_layout()
plt.show()
```


Cobertura Espacial — Real vs. Sintéticas Selecionadas
Real
Sintéticas Selecionadas
Y (m)
X (m)
0
2
4
6
8
10
12
14
16
18
20