fase3_2

May 21, 2025

[33]:
```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
```

[34]:
```python
# =================================
# 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)
```

[35]:
```python
# =================================
# 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)
```

```python
# Salvar o modelo
model_dnn.save("/home/darkcover/Documentos/Gan/Models/pseudo_label_model.keras")
```

```
Epoch 1/250

8/8              2s 38ms/step - loss:
1918.4399 - val_loss: 123.3480
Epoch 2/250
8/8              0s 13ms/step - loss:
91.9337 - val_loss: 71.7335
Epoch 3/250
8/8              0s 13ms/step - loss:
63.3985 - val_loss: 44.2677
Epoch 4/250
8/8              0s 14ms/step - loss:
48.3444 - val_loss: 33.4482
Epoch 5/250
8/8              0s 13ms/step - loss:
33.9562 - val_loss: 29.2731
Epoch 6/250
8/8              0s 13ms/step - loss:
28.1700 - val_loss: 23.6993
Epoch 7/250
8/8              0s 13ms/step - loss:
22.5565 - val_loss: 19.9321
Epoch 8/250
8/8              0s 14ms/step - loss:
18.1816 - val_loss: 17.7275
Epoch 9/250
8/8              0s 14ms/step - loss:
17.7881 - val_loss: 16.2061
Epoch 10/250
8/8              0s 13ms/step - loss:
16.6534 - val_loss: 14.8085
Epoch 11/250
8/8              0s 13ms/step - loss:
14.8596 - val_loss: 13.8147
Epoch 12/250
8/8              0s 13ms/step - loss:
13.3045 - val_loss: 11.8507
Epoch 13/250
8/8              0s 13ms/step - loss:
11.6568 - val_loss: 10.4727
Epoch 14/250
8/8              0s 13ms/step - loss:
9.7378 - val_loss: 9.2843
Epoch 15/250
8/8              0s 13ms/step - loss:
```

```
8.5150 - val_loss: 7.8157
Epoch 16/250
8/8            0s 13ms/step - loss:
7.2352 - val_loss: 6.6290
Epoch 17/250
8/8            0s 13ms/step - loss:
6.2295 - val_loss: 5.9885
Epoch 18/250
8/8            0s 13ms/step - loss:
5.7207 - val_loss: 5.1383
Epoch 19/250
8/8            0s 13ms/step - loss:
4.9185 - val_loss: 4.9630
Epoch 20/250
8/8            0s 13ms/step - loss:
4.7254 - val_loss: 5.2709
Epoch 21/250
8/8            0s 13ms/step - loss:
4.7733 - val_loss: 5.1562
Epoch 22/250
8/8            0s 13ms/step - loss:
4.9529 - val_loss: 5.3197
Epoch 23/250
8/8            0s 13ms/step - loss:
4.9335 - val_loss: 5.1622
Epoch 24/250
8/8            0s 13ms/step - loss:
4.3694 - val_loss: 4.5321
Epoch 25/250
8/8            0s 14ms/step - loss:
4.2866 - val_loss: 4.8205
Epoch 26/250
8/8            0s 19ms/step - loss:
4.3365 - val_loss: 4.5162
Epoch 27/250
8/8            0s 16ms/step - loss:
4.3279 - val_loss: 4.4671
Epoch 28/250
8/8            0s 19ms/step - loss:
3.9940 - val_loss: 4.4875
Epoch 29/250
8/8            0s 16ms/step - loss:
4.1959 - val_loss: 4.4012
Epoch 30/250
8/8            0s 15ms/step - loss:
4.1950 - val_loss: 4.2256
Epoch 31/250
8/8            0s 15ms/step - loss:
```

```
4.1101 - val_loss: 4.2619
Epoch 32/250
8/8            0s 16ms/step - loss:
3.9472 - val_loss: 4.7233
Epoch 33/250
8/8            0s 17ms/step - loss:
3.8310 - val_loss: 4.1380
Epoch 34/250
8/8            0s 17ms/step - loss:
3.9136 - val_loss: 4.0744
Epoch 35/250
8/8            0s 15ms/step - loss:
4.0219 - val_loss: 4.1986
Epoch 36/250
8/8            0s 15ms/step - loss:
3.4935 - val_loss: 4.0508
Epoch 37/250
8/8            0s 13ms/step - loss:
3.7200 - val_loss: 4.0263
Epoch 38/250
8/8            0s 14ms/step - loss:
3.7237 - val_loss: 3.9542
Epoch 39/250
8/8            0s 17ms/step - loss:
4.2211 - val_loss: 4.8645
Epoch 40/250
8/8            0s 16ms/step - loss:
4.1602 - val_loss: 5.0247
Epoch 41/250
8/8            0s 14ms/step - loss:
4.3937 - val_loss: 5.0287
Epoch 42/250
8/8            0s 15ms/step - loss:
4.1487 - val_loss: 4.1744
Epoch 43/250
8/8            0s 28ms/step - loss:
3.5144 - val_loss: 4.1166
Epoch 44/250
8/8            0s 19ms/step - loss:
4.2125 - val_loss: 5.2371
Epoch 45/250
8/8            0s 13ms/step - loss:
4.4151 - val_loss: 3.8922
Epoch 46/250
8/8            0s 14ms/step - loss:
3.6710 - val_loss: 3.7805
Epoch 47/250
8/8            0s 13ms/step - loss:
```

```
3.9533 - val_loss: 3.8797
Epoch 48/250
8/8            0s 13ms/step - loss:
3.5574 - val_loss: 3.9474
Epoch 49/250
8/8            0s 13ms/step - loss:
3.7332 - val_loss: 4.0632
Epoch 50/250
8/8            0s 13ms/step - loss:
3.7063 - val_loss: 4.7731
Epoch 51/250
8/8            0s 13ms/step - loss:
3.8886 - val_loss: 3.9884
Epoch 52/250
8/8            0s 13ms/step - loss:
3.4109 - val_loss: 3.7116
Epoch 53/250
8/8            0s 13ms/step - loss:
3.4204 - val_loss: 3.7883
Epoch 54/250
8/8            0s 14ms/step - loss:
3.3371 - val_loss: 3.9241
Epoch 55/250
8/8            0s 13ms/step - loss:
3.5332 - val_loss: 3.9396
Epoch 56/250
8/8            0s 14ms/step - loss:
3.6738 - val_loss: 3.7927
Epoch 57/250
8/8            0s 15ms/step - loss:
3.7892 - val_loss: 3.8183
Epoch 58/250
8/8            0s 16ms/step - loss:
4.0052 - val_loss: 4.3349
Epoch 59/250
8/8            0s 16ms/step - loss:
3.4092 - val_loss: 3.7952
Epoch 60/250
8/8            0s 15ms/step - loss:
3.4528 - val_loss: 3.7430
Epoch 61/250
8/8            0s 27ms/step - loss:
3.2570 - val_loss: 3.9642
Epoch 62/250
8/8            0s 14ms/step - loss:
3.3435 - val_loss: 3.8988
Epoch 63/250
8/8            0s 16ms/step - loss:
```

```
3.2473 - val_loss: 3.9612
Epoch 64/250
8/8            0s 15ms/step - loss:
3.6429 - val_loss: 3.8029
Epoch 65/250
8/8            0s 13ms/step - loss:
3.2844 - val_loss: 3.6122
Epoch 66/250
8/8            0s 13ms/step - loss:
3.2897 - val_loss: 3.7547
Epoch 67/250
8/8            0s 13ms/step - loss:
3.3917 - val_loss: 4.2650
Epoch 68/250
8/8            0s 16ms/step - loss:
3.4222 - val_loss: 4.0333
Epoch 69/250
8/8            0s 17ms/step - loss:
3.5339 - val_loss: 3.8665
Epoch 70/250
8/8            0s 15ms/step - loss:
3.2375 - val_loss: 3.5190
Epoch 71/250
8/8            0s 14ms/step - loss:
3.1870 - val_loss: 3.6121
Epoch 72/250
8/8            0s 14ms/step - loss:
3.3674 - val_loss: 3.7019
Epoch 73/250
8/8            0s 14ms/step - loss:
3.2982 - val_loss: 3.7362
Epoch 74/250
8/8            0s 13ms/step - loss:
3.3845 - val_loss: 4.4913
Epoch 75/250
8/8            0s 19ms/step - loss:
3.5884 - val_loss: 3.9656
Epoch 76/250
8/8            0s 17ms/step - loss:
3.5034 - val_loss: 3.7725
Epoch 77/250
8/8            0s 14ms/step - loss:
3.3811 - val_loss: 3.8900
Epoch 78/250
8/8            0s 13ms/step - loss:
3.3924 - val_loss: 3.8306
Epoch 79/250
8/8            0s 16ms/step - loss:
```

```
3.3399 - val_loss: 3.7815
Epoch 80/250
8/8              0s 18ms/step - loss:
3.1395 - val_loss: 3.7856
Epoch 81/250
8/8              0s 17ms/step - loss:
3.7090 - val_loss: 3.6853
Epoch 82/250
8/8              0s 17ms/step - loss:
3.0607 - val_loss: 3.6571
Epoch 83/250
8/8              0s 17ms/step - loss:
3.3859 - val_loss: 3.8379
Epoch 84/250
8/8              0s 16ms/step - loss:
3.1451 - val_loss: 3.5412
Epoch 85/250
8/8              0s 14ms/step - loss:
3.2023 - val_loss: 3.6028
Epoch 86/250
8/8              0s 13ms/step - loss:
3.3016 - val_loss: 3.8572
Epoch 87/250
8/8              0s 13ms/step - loss:
3.5570 - val_loss: 3.5096
Epoch 88/250
8/8              0s 14ms/step - loss:
3.2471 - val_loss: 3.5162
Epoch 89/250
8/8              0s 18ms/step - loss:
3.0058 - val_loss: 3.9020
Epoch 90/250
8/8              0s 20ms/step - loss:
3.0403 - val_loss: 3.9628
Epoch 91/250
8/8              0s 18ms/step - loss:
3.0173 - val_loss: 3.3913
Epoch 92/250
8/8              0s 14ms/step - loss:
3.3396 - val_loss: 3.8356
Epoch 93/250
8/8              0s 14ms/step - loss:
3.3955 - val_loss: 4.0431
Epoch 94/250
8/8              0s 16ms/step - loss:
3.2940 - val_loss: 3.5857
Epoch 95/250
8/8              0s 16ms/step - loss:
```

```
3.5791 - val_loss: 4.3708
Epoch 96/250
8/8              0s 13ms/step - loss:
3.4592 - val_loss: 3.9275
Epoch 97/250
8/8              0s 15ms/step - loss:
3.0427 - val_loss: 3.5305
Epoch 98/250
8/8              0s 21ms/step - loss:
3.0898 - val_loss: 3.4564
Epoch 99/250
8/8              0s 19ms/step - loss:
3.0763 - val_loss: 4.0141
Epoch 100/250
8/8              0s 15ms/step - loss:
3.5653 - val_loss: 3.7900
Epoch 101/250
8/8              0s 16ms/step - loss:
3.2799 - val_loss: 3.6945
Epoch 102/250
8/8              0s 14ms/step - loss:
3.5751 - val_loss: 3.3681
Epoch 103/250
8/8              0s 14ms/step - loss:
3.3871 - val_loss: 3.4569
Epoch 104/250
8/8              0s 16ms/step - loss:
3.0218 - val_loss: 3.5148
Epoch 105/250
8/8              0s 16ms/step - loss:
3.0626 - val_loss: 3.7389
Epoch 106/250
8/8              0s 39ms/step - loss:
3.1320 - val_loss: 3.2999
Epoch 107/250
8/8              0s 17ms/step - loss:
3.0195 - val_loss: 3.8026
Epoch 108/250
8/8              0s 14ms/step - loss:
3.2995 - val_loss: 3.4903
Epoch 109/250
8/8              0s 17ms/step - loss:
2.9554 - val_loss: 3.3274
Epoch 110/250
8/8              0s 16ms/step - loss:
2.9474 - val_loss: 3.2734
Epoch 111/250
8/8              0s 14ms/step - loss:
```

```
2.8492 - val_loss: 3.4922
Epoch 112/250
8/8            0s 15ms/step - loss:
3.1676 - val_loss: 3.4310
Epoch 113/250
8/8            0s 16ms/step - loss:
2.9392 - val_loss: 3.4707
Epoch 114/250
8/8            0s 13ms/step - loss:
3.0215 - val_loss: 3.6275
Epoch 115/250
8/8            0s 13ms/step - loss:
3.0582 - val_loss: 3.4924
Epoch 116/250
8/8            0s 13ms/step - loss:
3.1304 - val_loss: 3.3033
Epoch 117/250
8/8            0s 13ms/step - loss:
2.9306 - val_loss: 3.2812
Epoch 118/250
8/8            0s 13ms/step - loss:
2.9623 - val_loss: 3.6463
Epoch 119/250
8/8            0s 12ms/step - loss:
2.9771 - val_loss: 3.2923
Epoch 120/250
8/8            0s 13ms/step - loss:
3.1767 - val_loss: 3.9721
Epoch 121/250
8/8            0s 12ms/step - loss:
3.3571 - val_loss: 3.2481
Epoch 122/250
8/8            0s 13ms/step - loss:
2.9300 - val_loss: 3.2632
Epoch 123/250
8/8            0s 13ms/step - loss:
3.3300 - val_loss: 3.3451
Epoch 124/250
8/8            0s 13ms/step - loss:
2.8625 - val_loss: 3.6540
Epoch 125/250
8/8            0s 13ms/step - loss:
3.3044 - val_loss: 3.3976
Epoch 126/250
8/8            0s 12ms/step - loss:
3.6335 - val_loss: 3.4135
Epoch 127/250
8/8            0s 13ms/step - loss:
```

```
3.2759 - val_loss: 3.3847
Epoch 128/250
8/8            0s 17ms/step - loss:
3.3028 - val_loss: 3.3820
Epoch 129/250
8/8            0s 14ms/step - loss:
3.0881 - val_loss: 3.2860
Epoch 130/250
8/8            0s 15ms/step - loss:
3.1732 - val_loss: 3.6789
Epoch 131/250
8/8            0s 13ms/step - loss:
3.0261 - val_loss: 3.4222
Epoch 132/250
8/8            0s 13ms/step - loss:
3.0202 - val_loss: 3.1877
Epoch 133/250
8/8            0s 13ms/step - loss:
2.8390 - val_loss: 3.4471
Epoch 134/250
8/8            0s 14ms/step - loss:
3.3808 - val_loss: 3.8040
Epoch 135/250
8/8            0s 18ms/step - loss:
3.5450 - val_loss: 4.1029
Epoch 136/250
8/8            0s 16ms/step - loss:
3.3996 - val_loss: 3.5228
Epoch 137/250
8/8            0s 14ms/step - loss:
3.0014 - val_loss: 4.1702
Epoch 138/250
8/8            0s 15ms/step - loss:
3.3622 - val_loss: 4.2507
Epoch 139/250
8/8            0s 13ms/step - loss:
3.5239 - val_loss: 4.0857
Epoch 140/250
8/8            0s 13ms/step - loss:
3.3365 - val_loss: 4.0007
Epoch 141/250
8/8            0s 13ms/step - loss:
3.6556 - val_loss: 4.3725
Epoch 142/250
8/8            0s 13ms/step - loss:
3.4249 - val_loss: 3.4605
Epoch 143/250
8/8            0s 13ms/step - loss:
```

```
3.3095 - val_loss: 3.6842
Epoch 144/250
8/8             0s 15ms/step - loss:
3.5377 - val_loss: 4.0976
Epoch 145/250
8/8             0s 13ms/step - loss:
3.3792 - val_loss: 3.5307
Epoch 146/250
8/8             0s 21ms/step - loss:
3.2307 - val_loss: 3.3244
Epoch 147/250
8/8             0s 15ms/step - loss:
3.0717 - val_loss: 3.1948
Epoch 148/250
8/8             0s 19ms/step - loss:
2.8372 - val_loss: 3.1924
Epoch 149/250
8/8             0s 17ms/step - loss:
2.9449 - val_loss: 3.4436
Epoch 150/250
8/8             0s 16ms/step - loss:
3.1724 - val_loss: 3.2725
Epoch 151/250
8/8             0s 16ms/step - loss:
2.9288 - val_loss: 3.3891
Epoch 152/250
8/8             0s 17ms/step - loss:
3.1180 - val_loss: 3.2663
Epoch 153/250
8/8             0s 16ms/step - loss:
3.0802 - val_loss: 3.0891
Epoch 154/250
8/8             0s 15ms/step - loss:
3.1676 - val_loss: 3.3667
Epoch 155/250
8/8             0s 14ms/step - loss:
2.8837 - val_loss: 3.4482
Epoch 156/250
8/8             0s 15ms/step - loss:
2.7958 - val_loss: 2.9868
Epoch 157/250
8/8             0s 13ms/step - loss:
2.8710 - val_loss: 3.0894
Epoch 158/250
8/8             0s 13ms/step - loss:
2.8029 - val_loss: 3.2512
Epoch 159/250
8/8             0s 13ms/step - loss:
```

```
2.8113 - val_loss: 2.9710
Epoch 160/250
8/8            0s 13ms/step - loss:
2.5236 - val_loss: 3.0272
Epoch 161/250
8/8            0s 14ms/step - loss:
2.8521 - val_loss: 3.2415
Epoch 162/250
8/8            0s 13ms/step - loss:
2.8121 - val_loss: 2.9064
Epoch 163/250
8/8            0s 13ms/step - loss:
2.6733 - val_loss: 2.8839
Epoch 164/250
8/8            0s 13ms/step - loss:
2.9209 - val_loss: 3.3569
Epoch 165/250
8/8            0s 13ms/step - loss:
2.9056 - val_loss: 2.8741
Epoch 166/250
8/8            0s 15ms/step - loss:
2.5774 - val_loss: 2.9144
Epoch 167/250
8/8            0s 14ms/step - loss:
2.6709 - val_loss: 3.0261
Epoch 168/250
8/8            0s 13ms/step - loss:
2.8163 - val_loss: 2.9801
Epoch 169/250
8/8            0s 13ms/step - loss:
2.6176 - val_loss: 2.8335
Epoch 170/250
8/8            0s 17ms/step - loss:
2.7054 - val_loss: 3.0623
Epoch 171/250
8/8            0s 15ms/step - loss:
3.0994 - val_loss: 3.1713
Epoch 172/250
8/8            0s 14ms/step - loss:
2.9912 - val_loss: 3.1087
Epoch 173/250
8/8            0s 16ms/step - loss:
2.7611 - val_loss: 2.8760
Epoch 174/250
8/8            0s 18ms/step - loss:
2.9130 - val_loss: 3.0043
Epoch 175/250
8/8            0s 16ms/step - loss:
```

```
2.5414 - val_loss: 2.9790
Epoch 176/250
8/8             0s 15ms/step - loss:
2.5630 - val_loss: 2.8930
Epoch 177/250
8/8             0s 14ms/step - loss:
2.4459 - val_loss: 2.8482
Epoch 178/250
8/8             0s 13ms/step - loss:
2.7471 - val_loss: 2.8158
Epoch 179/250
8/8             0s 13ms/step - loss:
2.6262 - val_loss: 2.9421
Epoch 180/250
8/8             0s 14ms/step - loss:
2.7541 - val_loss: 3.1860
Epoch 181/250
8/8             0s 13ms/step - loss:
2.8702 - val_loss: 3.2193
Epoch 182/250
8/8             0s 13ms/step - loss:
3.0620 - val_loss: 2.8906
Epoch 183/250
8/8             0s 13ms/step - loss:
2.6153 - val_loss: 3.0230
Epoch 184/250
8/8             0s 22ms/step - loss:
2.9780 - val_loss: 2.8207
Epoch 185/250
8/8             0s 18ms/step - loss:
2.6279 - val_loss: 3.1171
Epoch 186/250
8/8             0s 13ms/step - loss:
2.8123 - val_loss: 2.7742
Epoch 187/250
8/8             0s 13ms/step - loss:
2.5581 - val_loss: 2.9438
Epoch 188/250
8/8             0s 13ms/step - loss:
2.6428 - val_loss: 3.0153
Epoch 189/250
8/8             0s 13ms/step - loss:
2.6929 - val_loss: 2.7569
Epoch 190/250
8/8             0s 13ms/step - loss:
2.6233 - val_loss: 2.9072
Epoch 191/250
8/8             0s 13ms/step - loss:
```

```
2.6010 - val_loss: 3.0313
Epoch 192/250
8/8            0s 13ms/step - loss:
2.9142 - val_loss: 2.6704
Epoch 193/250
8/8            0s 12ms/step - loss:
2.6024 - val_loss: 2.9002
Epoch 194/250
8/8            0s 13ms/step - loss:
2.8371 - val_loss: 3.1703
Epoch 195/250
8/8            0s 15ms/step - loss:
2.5916 - val_loss: 2.8851
Epoch 196/250
8/8            0s 13ms/step - loss:
2.5158 - val_loss: 2.8282
Epoch 197/250
8/8            0s 13ms/step - loss:
2.4862 - val_loss: 2.6464
Epoch 198/250
8/8            0s 13ms/step - loss:
2.5178 - val_loss: 2.7205
Epoch 199/250
8/8            0s 13ms/step - loss:
2.6780 - val_loss: 2.6854
Epoch 200/250
8/8            0s 13ms/step - loss:
2.7847 - val_loss: 2.7456
Epoch 201/250
8/8            0s 14ms/step - loss:
2.6973 - val_loss: 3.0340
Epoch 202/250
8/8            0s 13ms/step - loss:
2.6448 - val_loss: 3.3541
Epoch 203/250
8/8            0s 13ms/step - loss:
2.9174 - val_loss: 2.6517
Epoch 204/250
8/8            0s 18ms/step - loss:
2.5360 - val_loss: 2.8544
Epoch 205/250
8/8            0s 17ms/step - loss:
2.5844 - val_loss: 2.7052
Epoch 206/250
8/8            0s 18ms/step - loss:
2.4476 - val_loss: 2.6490
Epoch 207/250
8/8            0s 18ms/step - loss:
```

```
2.5809 - val_loss: 2.9596
Epoch 208/250
8/8              0s 17ms/step - loss:
2.5066 - val_loss: 2.9539
Epoch 209/250
8/8              0s 19ms/step - loss:
2.7879 - val_loss: 2.7316
Epoch 210/250
8/8              0s 16ms/step - loss:
2.4063 - val_loss: 2.7973
Epoch 211/250
8/8              0s 13ms/step - loss:
2.5988 - val_loss: 2.6656
Epoch 212/250
8/8              0s 13ms/step - loss:
2.3476 - val_loss: 2.6060
Epoch 213/250
8/8              0s 13ms/step - loss:
2.5817 - val_loss: 2.5777
Epoch 214/250
8/8              0s 17ms/step - loss:
2.4121 - val_loss: 2.6798
Epoch 215/250
8/8              0s 22ms/step - loss:
2.4011 - val_loss: 2.6544
Epoch 216/250
8/8              0s 16ms/step - loss:
2.3469 - val_loss: 2.6284
Epoch 217/250
8/8              0s 13ms/step - loss:
2.1841 - val_loss: 2.7264
Epoch 218/250
8/8              0s 13ms/step - loss:
2.3985 - val_loss: 2.5384
Epoch 219/250
8/8              0s 13ms/step - loss:
2.2590 - val_loss: 2.5911
Epoch 220/250
8/8              0s 13ms/step - loss:
2.6058 - val_loss: 2.5409
Epoch 221/250
8/8              0s 13ms/step - loss:
2.5033 - val_loss: 2.6867
Epoch 222/250
8/8              0s 13ms/step - loss:
2.4019 - val_loss: 2.8562
Epoch 223/250
8/8              0s 13ms/step - loss:
```

```
2.5830 - val_loss: 2.6484
Epoch 224/250
8/8             0s 13ms/step - loss:
2.4115 - val_loss: 2.5228
Epoch 225/250
8/8             0s 13ms/step - loss:
2.3907 - val_loss: 2.8402
Epoch 226/250
8/8             0s 13ms/step - loss:
2.4103 - val_loss: 2.8649
Epoch 227/250
8/8             0s 13ms/step - loss:
2.4914 - val_loss: 2.6461
Epoch 228/250
8/8             0s 13ms/step - loss:
2.5274 - val_loss: 2.5128
Epoch 229/250
8/8             0s 13ms/step - loss:
2.3912 - val_loss: 3.0698
Epoch 230/250
8/8             0s 13ms/step - loss:
2.6779 - val_loss: 2.6600
Epoch 231/250
8/8             0s 13ms/step - loss:
2.4622 - val_loss: 2.7183
Epoch 232/250
8/8             0s 13ms/step - loss:
2.7125 - val_loss: 3.2702
Epoch 233/250
8/8             0s 13ms/step - loss:
2.5869 - val_loss: 2.5520
Epoch 234/250
8/8             0s 12ms/step - loss:
2.3138 - val_loss: 2.7724
Epoch 235/250
8/8             0s 13ms/step - loss:
2.3066 - val_loss: 2.6678
Epoch 236/250
8/8             0s 13ms/step - loss:
2.3575 - val_loss: 2.9221
Epoch 237/250
8/8             0s 13ms/step - loss:
2.6503 - val_loss: 2.5473
Epoch 238/250
8/8             0s 13ms/step - loss:
2.3392 - val_loss: 2.5012
Epoch 239/250
8/8             0s 13ms/step - loss:
```

```
2.4223 - val_loss: 2.5203
Epoch 240/250
8/8              0s 13ms/step - loss:
2.2452 - val_loss: 2.5098
Epoch 241/250
8/8              0s 13ms/step - loss:
2.2928 - val_loss: 2.6741
Epoch 242/250
8/8              0s 13ms/step - loss:
2.2902 - val_loss: 2.4388
Epoch 243/250
8/8              0s 14ms/step - loss:
2.4358 - val_loss: 2.6542
Epoch 244/250
8/8              0s 24ms/step - loss:
2.3562 - val_loss: 2.8325
Epoch 245/250
8/8              0s 21ms/step - loss:
2.5471 - val_loss: 2.5571
Epoch 246/250
8/8              0s 13ms/step - loss:
2.4250 - val_loss: 2.4898
Epoch 247/250
8/8              0s 13ms/step - loss:
2.2759 - val_loss: 2.4635
Epoch 248/250
8/8              0s 13ms/step - loss:
2.2975 - val_loss: 2.5253
Epoch 249/250
8/8              0s 13ms/step - loss:
2.2699 - val_loss: 2.4112
Epoch 250/250
8/8              0s 13ms/step - loss:
2.4405 - val_loss: 2.5085
```

[36]:

```
# ================================
# 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 1ms/step
```

[37]:

```
# ================================
# 4. Avaliar realismo via Discriminador (critério 2)
#   (Modelo DNN com 2 camadas: 30 e 20 neurônios, 250 epochs, batch_size=100)
```

```python
# =================================
# 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()

# 4.1. Salvar df_generated com coordenadas e D_score para uso na Fase 4
df_generated.to_csv("/home/darkcover/Documentos/Gan/Data/
↪df_generated_with_coords.csv", index=False)
print(">>> df_generated_with_coords.csv gravado com X, Y e D_score")
```

```
1250/1250            2s 1ms/step
>>> df_generated_with_coords.csv gravado com X, Y e D_score
```

[38]:

```python
# =================================
# 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)
```

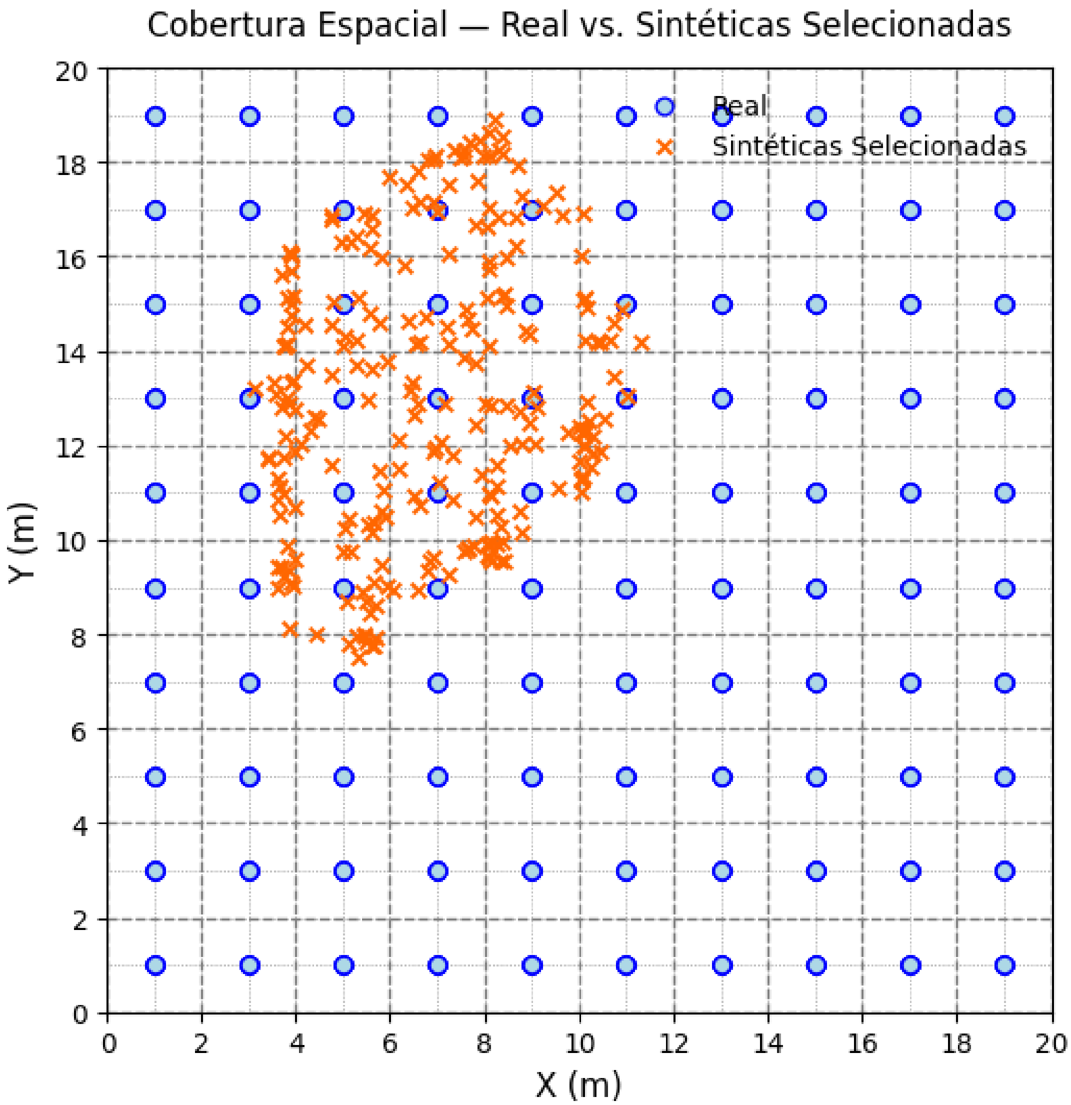
[39]:

```python
# =================================
# 6. Logs e Visualizações
# =================================
```

```python
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')

# 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()
```

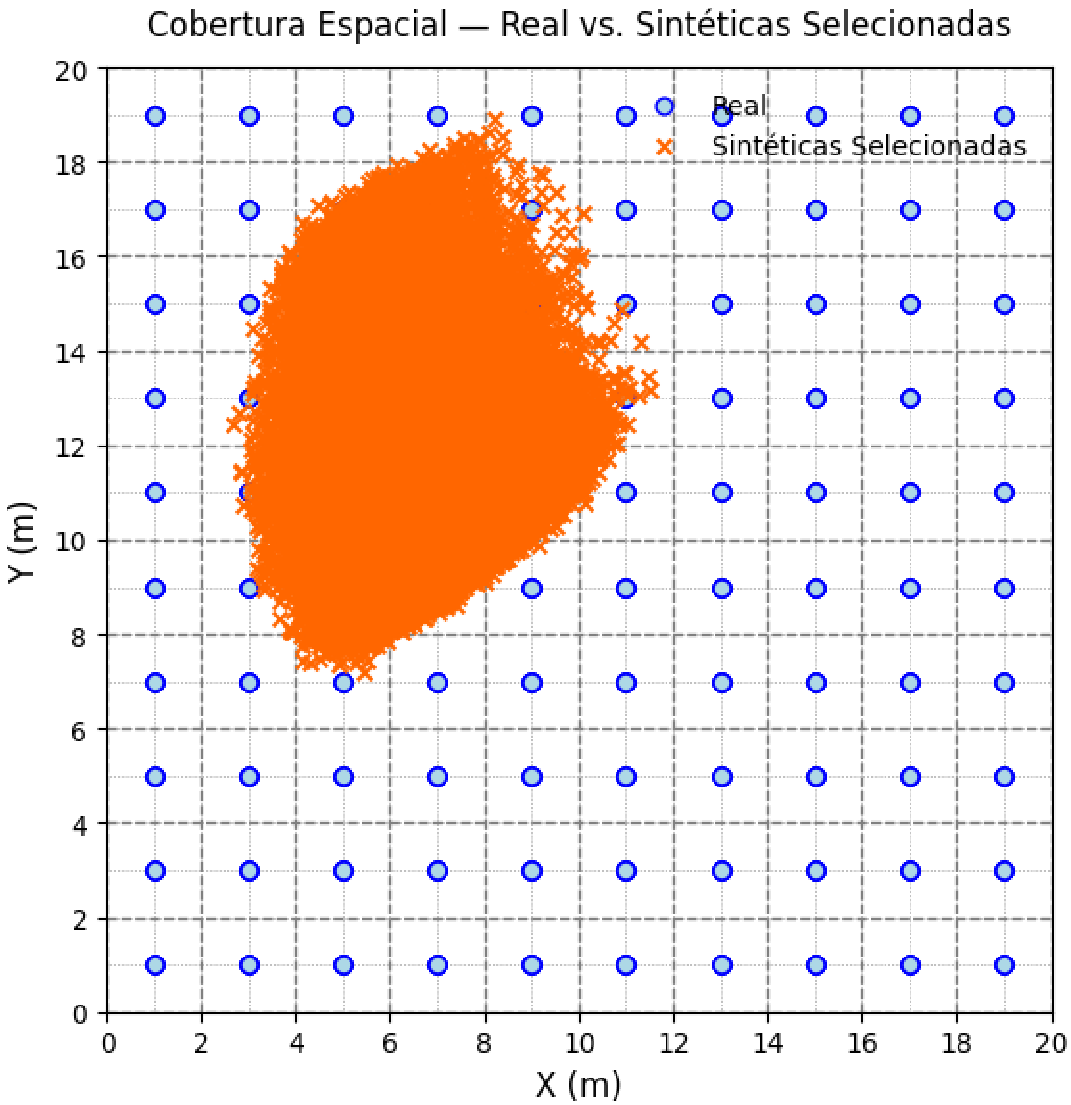
```
Total sintéticas selecionadas: 251
Zona 032: 116 geradas, D_score médio selecionadas 1.000
Zona 041: 90 geradas, D_score médio selecionadas 1.000
Zona 042: 2338 geradas, D_score médio selecionadas 1.000
Zona 043: 5467 geradas, D_score médio selecionadas 1.000
Zona 044: 415 geradas, D_score médio selecionadas 1.000
```

```
Zona 051: 279 geradas, D_score médio selecionadas 1.000
Zona 052: 3993 geradas, D_score médio selecionadas 1.000
Zona 053: 6294 geradas, D_score médio selecionadas 1.000
Zona 054: 4382 geradas, D_score médio selecionadas 1.000
Zona 055: 66 geradas, D_score médio selecionadas 1.000
Zona 061: 215 geradas, D_score médio selecionadas 1.000
Zona 062: 3721 geradas, D_score médio selecionadas 1.000
Zona 063: 4515 geradas, D_score médio selecionadas 1.000
Zona 064: 1370 geradas, D_score médio selecionadas 1.000
Zona 065: 165 geradas, D_score médio selecionadas 1.000
Zona 071: 88 geradas, D_score médio selecionadas 1.000
Zona 072: 2159 geradas, D_score médio selecionadas 1.000
Zona 073: 2201 geradas, D_score médio selecionadas 1.000
Zona 074: 445 geradas, D_score médio selecionadas 1.000
Zona 075: 11 geradas, D_score médio selecionadas 1.000
Zona 081: 3 geradas, D_score médio selecionadas 1.000
Zona 082: 810 geradas, D_score médio selecionadas 1.000
Zona 083: 742 geradas, D_score médio selecionadas 1.000
Zona 084: 82 geradas, D_score médio selecionadas 1.000
Zona 085: 2 geradas, D_score médio selecionadas 1.000
Zona 093: 25 geradas, D_score médio selecionadas 1.000
Zona 094: 6 geradas, D_score médio selecionadas 1.000
```



[40]:
```
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',
```

```python
          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