fase3_2

May 21, 2025

[1]:
```
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 06:09:45.077568: 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 06:09:45.191875: 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 06:09:45.324867: 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 00:00:1747822185.441995   65371 cuda_dnn.cc:8579] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1747822185.477283   65371 cuda_blas.cc:1407] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
W0000 00:00:1747822185.642571   65371 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747822185.642612   65371 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747822185.642617   65371 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747822185.642621   65371 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
2025-05-21 06:09:45.670346: 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]:
```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)
```

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

```
Epoch 1/250

2025-05-21 06:09:51.667334: 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 36ms/step - loss:
225.2290 - val_loss: 48.9505
Epoch 2/250
8/8             0s 14ms/step - loss:
48.0811 - val_loss: 38.8521
Epoch 3/250
8/8             0s 13ms/step - loss:
```

```
33.1836 - val_loss: 24.7351
Epoch 4/250
8/8            0s 27ms/step - loss:
23.1559 - val_loss: 19.1107
Epoch 5/250
8/8            0s 13ms/step - loss:
17.4890 - val_loss: 15.9609
Epoch 6/250
8/8            0s 13ms/step - loss:
15.8472 - val_loss: 12.8586
Epoch 7/250
8/8            0s 13ms/step - loss:
12.4958 - val_loss: 11.4062
Epoch 8/250
8/8            0s 13ms/step - loss:
10.7754 - val_loss: 9.6301
Epoch 9/250
8/8            0s 13ms/step - loss:
8.8141 - val_loss: 7.8427
Epoch 10/250
8/8            0s 13ms/step - loss:
7.2449 - val_loss: 7.3323
Epoch 11/250
8/8            0s 15ms/step - loss:
6.8941 - val_loss: 6.7287
Epoch 12/250
8/8            0s 13ms/step - loss:
6.8390 - val_loss: 6.1989
Epoch 13/250
8/8            0s 13ms/step - loss:
5.6920 - val_loss: 5.8750
Epoch 14/250
8/8            0s 13ms/step - loss:
5.6877 - val_loss: 5.7013
Epoch 15/250
8/8            0s 13ms/step - loss:
5.8598 - val_loss: 5.8856
Epoch 16/250
8/8            0s 14ms/step - loss:
5.5990 - val_loss: 5.4417
Epoch 17/250
8/8            0s 15ms/step - loss:
5.9827 - val_loss: 6.2285
Epoch 18/250
8/8            0s 16ms/step - loss:
5.9945 - val_loss: 5.0193
Epoch 19/250
8/8            0s 15ms/step - loss:
```

```
5.1531 - val_loss: 5.0263
Epoch 20/250
8/8             0s 18ms/step - loss:
5.2115 - val_loss: 5.2593
Epoch 21/250
8/8             0s 25ms/step - loss:
5.0357 - val_loss: 5.1957
Epoch 22/250
8/8             0s 19ms/step - loss:
4.9906 - val_loss: 4.7834
Epoch 23/250
8/8             0s 15ms/step - loss:
4.9282 - val_loss: 4.6189
Epoch 24/250
8/8             0s 14ms/step - loss:
4.6593 - val_loss: 4.9295
Epoch 25/250
8/8             0s 14ms/step - loss:
4.6845 - val_loss: 4.4611
Epoch 26/250
8/8             0s 16ms/step - loss:
4.5100 - val_loss: 4.8161
Epoch 27/250
8/8             0s 17ms/step - loss:
4.6263 - val_loss: 4.4389
Epoch 28/250
8/8             0s 15ms/step - loss:
4.3312 - val_loss: 4.3615
Epoch 29/250
8/8             0s 15ms/step - loss:
4.4796 - val_loss: 4.3241
Epoch 30/250
8/8             0s 15ms/step - loss:
4.6129 - val_loss: 5.4834
Epoch 31/250
8/8             0s 15ms/step - loss:
4.8614 - val_loss: 4.8015
Epoch 32/250
8/8             0s 14ms/step - loss:
5.0735 - val_loss: 4.5783
Epoch 33/250
8/8             0s 19ms/step - loss:
4.5324 - val_loss: 4.2041
Epoch 34/250
8/8             0s 16ms/step - loss:
4.2973 - val_loss: 4.2100
Epoch 35/250
8/8             0s 16ms/step - loss:
```

```
4.4238 - val_loss: 4.2128
Epoch 36/250
8/8            0s 15ms/step - loss:
4.3144 - val_loss: 4.2439
Epoch 37/250
8/8            0s 16ms/step - loss:
4.0161 - val_loss: 4.4072
Epoch 38/250
8/8            0s 20ms/step - loss:
4.5975 - val_loss: 5.4604
Epoch 39/250
8/8            0s 16ms/step - loss:
5.1419 - val_loss: 4.2823
Epoch 40/250
8/8            0s 15ms/step - loss:
4.5289 - val_loss: 4.7382
Epoch 41/250
8/8            0s 15ms/step - loss:
4.5610 - val_loss: 4.1976
Epoch 42/250
8/8            0s 18ms/step - loss:
4.2357 - val_loss: 4.4281
Epoch 43/250
8/8            0s 17ms/step - loss:
4.1101 - val_loss: 4.0004
Epoch 44/250
8/8            0s 16ms/step - loss:
4.3171 - val_loss: 3.9503
Epoch 45/250
8/8            0s 14ms/step - loss:
4.0939 - val_loss: 4.4447
Epoch 46/250
8/8            0s 23ms/step - loss:
4.3339 - val_loss: 4.6227
Epoch 47/250
8/8            0s 15ms/step - loss:
3.7886 - val_loss: 4.2211
Epoch 48/250
8/8            0s 16ms/step - loss:
4.0820 - val_loss: 4.5420
Epoch 49/250
8/8            0s 15ms/step - loss:
4.5593 - val_loss: 4.4349
Epoch 50/250
8/8            0s 14ms/step - loss:
4.3148 - val_loss: 4.2845
Epoch 51/250
8/8            0s 14ms/step - loss:
```

```
3.8308 - val_loss: 3.7452
Epoch 52/250
8/8            0s 16ms/step - loss:
3.8542 - val_loss: 3.8048
Epoch 53/250
8/8            0s 15ms/step - loss:
3.9126 - val_loss: 3.9600
Epoch 54/250
8/8            0s 14ms/step - loss:
4.3832 - val_loss: 4.0146
Epoch 55/250
8/8            0s 14ms/step - loss:
4.3315 - val_loss: 4.0181
Epoch 56/250
8/8            0s 15ms/step - loss:
4.1917 - val_loss: 4.1740
Epoch 57/250
8/8            0s 16ms/step - loss:
4.2072 - val_loss: 4.0058
Epoch 58/250
8/8            0s 15ms/step - loss:
4.1314 - val_loss: 4.0397
Epoch 59/250
8/8            0s 15ms/step - loss:
3.6320 - val_loss: 3.7346
Epoch 60/250
8/8            0s 16ms/step - loss:
3.7324 - val_loss: 3.6166
Epoch 61/250
8/8            0s 15ms/step - loss:
3.6848 - val_loss: 3.8937
Epoch 62/250
8/8            0s 15ms/step - loss:
3.7771 - val_loss: 3.9210
Epoch 63/250
8/8            0s 15ms/step - loss:
3.7499 - val_loss: 3.9125
Epoch 64/250
8/8            0s 15ms/step - loss:
3.7816 - val_loss: 3.6178
Epoch 65/250
8/8            0s 16ms/step - loss:
4.0502 - val_loss: 3.8862
Epoch 66/250
8/8            0s 26ms/step - loss:
4.0172 - val_loss: 3.6463
Epoch 67/250
8/8            0s 14ms/step - loss:
```

```
3.8670 - val_loss: 3.8792
Epoch 68/250
8/8            0s 14ms/step - loss:
3.9916 - val_loss: 3.7919
Epoch 69/250
8/8            0s 15ms/step - loss:
4.1001 - val_loss: 4.4588
Epoch 70/250
8/8            0s 15ms/step - loss:
4.5469 - val_loss: 4.9861
Epoch 71/250
8/8            0s 15ms/step - loss:
4.6404 - val_loss: 4.1167
Epoch 72/250
8/8            0s 16ms/step - loss:
4.1575 - val_loss: 4.1079
Epoch 73/250
8/8            0s 14ms/step - loss:
4.3289 - val_loss: 3.8350
Epoch 74/250
8/8            0s 15ms/step - loss:
3.7022 - val_loss: 3.4135
Epoch 75/250
8/8            0s 16ms/step - loss:
3.3698 - val_loss: 3.6056
Epoch 76/250
8/8            0s 15ms/step - loss:
3.7491 - val_loss: 4.4831
Epoch 77/250
8/8            0s 15ms/step - loss:
4.4797 - val_loss: 4.7889
Epoch 78/250
8/8            0s 25ms/step - loss:
4.1253 - val_loss: 3.8260
Epoch 79/250
8/8            0s 16ms/step - loss:
3.7549 - val_loss: 4.0267
Epoch 80/250
8/8            0s 15ms/step - loss:
3.6783 - val_loss: 3.3645
Epoch 81/250
8/8            0s 16ms/step - loss:
3.1779 - val_loss: 3.4905
Epoch 82/250
8/8            0s 15ms/step - loss:
3.3743 - val_loss: 3.5248
Epoch 83/250
8/8            0s 15ms/step - loss:
```

```
3.3968 - val_loss: 3.3479
Epoch 84/250
8/8            0s 15ms/step - loss:
3.4818 - val_loss: 3.6155
Epoch 85/250
8/8            0s 17ms/step - loss:
3.5587 - val_loss: 3.5779
Epoch 86/250
8/8            0s 15ms/step - loss:
3.6455 - val_loss: 3.9540
Epoch 87/250
8/8            0s 15ms/step - loss:
3.5046 - val_loss: 3.7799
Epoch 88/250
8/8            0s 16ms/step - loss:
3.3664 - val_loss: 3.8674
Epoch 89/250
8/8            0s 15ms/step - loss:
3.6069 - val_loss: 3.9458
Epoch 90/250
8/8            0s 15ms/step - loss:
3.5297 - val_loss: 3.2649
Epoch 91/250
8/8            0s 15ms/step - loss:
3.4220 - val_loss: 3.7874
Epoch 92/250
8/8            0s 15ms/step - loss:
3.2555 - val_loss: 3.2582
Epoch 93/250
8/8            0s 15ms/step - loss:
3.3738 - val_loss: 3.2487
Epoch 94/250
8/8            0s 15ms/step - loss:
3.0997 - val_loss: 3.5620
Epoch 95/250
8/8            0s 15ms/step - loss:
3.2847 - val_loss: 3.4976
Epoch 96/250
8/8            0s 24ms/step - loss:
3.3554 - val_loss: 3.1870
Epoch 97/250
8/8            0s 22ms/step - loss:
3.2682 - val_loss: 3.7821
Epoch 98/250
8/8            0s 16ms/step - loss:
3.6648 - val_loss: 3.3343
Epoch 99/250
8/8            0s 14ms/step - loss:
```

```
3.3806 - val_loss: 3.4415
Epoch 100/250
8/8            0s 15ms/step - loss:
3.3375 - val_loss: 3.3251
Epoch 101/250
8/8            0s 15ms/step - loss:
3.1834 - val_loss: 3.1491
Epoch 102/250
8/8            0s 15ms/step - loss:
2.9890 - val_loss: 3.2246
Epoch 103/250
8/8            0s 14ms/step - loss:
3.1030 - val_loss: 3.1875
Epoch 104/250
8/8            0s 15ms/step - loss:
3.1880 - val_loss: 3.1368
Epoch 105/250
8/8            0s 14ms/step - loss:
3.0560 - val_loss: 3.2332
Epoch 106/250
8/8            0s 13ms/step - loss:
2.9840 - val_loss: 3.4493
Epoch 107/250
8/8            0s 15ms/step - loss:
3.4091 - val_loss: 3.3539
Epoch 108/250
8/8            0s 15ms/step - loss:
3.3986 - val_loss: 3.6104
Epoch 109/250
8/8            0s 19ms/step - loss:
3.0446 - val_loss: 3.1120
Epoch 110/250
8/8            0s 16ms/step - loss:
3.2354 - val_loss: 3.1098
Epoch 111/250
8/8            0s 17ms/step - loss:
3.0711 - val_loss: 3.1889
Epoch 112/250
8/8            0s 16ms/step - loss:
2.9922 - val_loss: 3.2166
Epoch 113/250
8/8            0s 23ms/step - loss:
3.0885 - val_loss: 3.1175
Epoch 114/250
8/8            0s 15ms/step - loss:
3.0562 - val_loss: 3.2100
Epoch 115/250
8/8            0s 16ms/step - loss:
```

```
3.2446 - val_loss: 3.5411
Epoch 116/250
8/8            0s 16ms/step - loss:
3.6000 - val_loss: 3.3373
Epoch 117/250
8/8            0s 17ms/step - loss:
3.1815 - val_loss: 3.3000
Epoch 118/250
8/8            0s 15ms/step - loss:
3.3869 - val_loss: 3.3674
Epoch 119/250
8/8            0s 15ms/step - loss:
3.2048 - val_loss: 3.0711
Epoch 120/250
8/8            0s 15ms/step - loss:
3.2712 - val_loss: 3.3854
Epoch 121/250
8/8            0s 15ms/step - loss:
3.3963 - val_loss: 3.4105
Epoch 122/250
8/8            0s 17ms/step - loss:
3.1527 - val_loss: 3.1056
Epoch 123/250
8/8            0s 15ms/step - loss:
3.2696 - val_loss: 3.4088
Epoch 124/250
8/8            0s 16ms/step - loss:
3.1162 - val_loss: 3.0132
Epoch 125/250
8/8            0s 15ms/step - loss:
3.1078 - val_loss: 3.1898
Epoch 126/250
8/8            0s 15ms/step - loss:
2.8116 - val_loss: 3.0593
Epoch 127/250
8/8            0s 15ms/step - loss:
2.9025 - val_loss: 3.1661
Epoch 128/250
8/8            0s 16ms/step - loss:
3.0102 - val_loss: 3.5520
Epoch 129/250
8/8            0s 16ms/step - loss:
3.4107 - val_loss: 3.2521
Epoch 130/250
8/8            0s 16ms/step - loss:
3.1430 - val_loss: 3.0559
Epoch 131/250
8/8            0s 14ms/step - loss:
```

```
2.9547 - val_loss: 3.0736
Epoch 132/250
8/8            0s 15ms/step - loss:
2.8891 - val_loss: 3.1722
Epoch 133/250
8/8            0s 15ms/step - loss:
2.8756 - val_loss: 3.1165
Epoch 134/250
8/8            0s 15ms/step - loss:
3.0503 - val_loss: 3.4193
Epoch 135/250
8/8            0s 30ms/step - loss:
3.2509 - val_loss: 3.0149
Epoch 136/250
8/8            0s 17ms/step - loss:
3.0833 - val_loss: 3.2408
Epoch 137/250
8/8            0s 14ms/step - loss:
3.0824 - val_loss: 3.0025
Epoch 138/250
8/8            0s 15ms/step - loss:
3.0581 - val_loss: 3.2466
Epoch 139/250
8/8            0s 15ms/step - loss:
3.0376 - val_loss: 2.9359
Epoch 140/250
8/8            0s 15ms/step - loss:
3.0345 - val_loss: 2.9833
Epoch 141/250
8/8            0s 15ms/step - loss:
4.7942 - val_loss: 3.9384
Epoch 142/250
8/8            0s 15ms/step - loss:
5.0845 - val_loss: 3.7631
Epoch 143/250
8/8            0s 16ms/step - loss:
3.5571 - val_loss: 3.5936
Epoch 144/250
8/8            0s 15ms/step - loss:
3.5092 - val_loss: 3.2621
Epoch 145/250
8/8            0s 16ms/step - loss:
3.4099 - val_loss: 3.4115
Epoch 146/250
8/8            0s 15ms/step - loss:
3.2097 - val_loss: 3.2733
Epoch 147/250
8/8            0s 17ms/step - loss:
```

```
3.4026 - val_loss: 3.0059
Epoch 148/250
8/8            0s 15ms/step - loss:
3.4358 - val_loss: 3.5528
Epoch 149/250
8/8            0s 15ms/step - loss:
3.6286 - val_loss: 3.0983
Epoch 150/250
8/8            0s 15ms/step - loss:
3.3349 - val_loss: 2.9492
Epoch 151/250
8/8            0s 15ms/step - loss:
2.9537 - val_loss: 2.9727
Epoch 152/250
8/8            0s 15ms/step - loss:
2.8586 - val_loss: 2.9235
Epoch 153/250
8/8            0s 16ms/step - loss:
2.8897 - val_loss: 3.0769
Epoch 154/250
8/8            0s 15ms/step - loss:
2.8074 - val_loss: 3.1005
Epoch 155/250
8/8            0s 15ms/step - loss:
3.1719 - val_loss: 2.9261
Epoch 156/250
8/8            0s 15ms/step - loss:
3.0852 - val_loss: 3.2114
Epoch 157/250
8/8            0s 15ms/step - loss:
3.1005 - val_loss: 2.8361
Epoch 158/250
8/8            0s 15ms/step - loss:
3.0428 - val_loss: 2.9086
Epoch 159/250
8/8            0s 16ms/step - loss:
2.8439 - val_loss: 2.9515
Epoch 160/250
8/8            0s 16ms/step - loss:
2.8407 - val_loss: 2.9237
Epoch 161/250
8/8            0s 15ms/step - loss:
2.7900 - val_loss: 2.9818
Epoch 162/250
8/8            0s 15ms/step - loss:
2.9607 - val_loss: 3.1891
Epoch 163/250
8/8            0s 16ms/step - loss:
```

```
3.0279 - val_loss: 3.0650
Epoch 164/250
8/8            0s 15ms/step - loss:
3.0599 - val_loss: 3.2494
Epoch 165/250
8/8            0s 15ms/step - loss:
2.9604 - val_loss: 3.0460
Epoch 166/250
8/8            0s 15ms/step - loss:
2.9486 - val_loss: 2.9496
Epoch 167/250
8/8            0s 15ms/step - loss:
2.9382 - val_loss: 2.7240
Epoch 168/250
8/8            0s 16ms/step - loss:
2.8286 - val_loss: 3.3075
Epoch 169/250
8/8            0s 15ms/step - loss:
2.9233 - val_loss: 2.8125
Epoch 170/250
8/8            0s 25ms/step - loss:
2.8798 - val_loss: 2.8717
Epoch 171/250
8/8            0s 15ms/step - loss:
2.7877 - val_loss: 2.9914
Epoch 172/250
8/8            0s 29ms/step - loss:
2.6135 - val_loss: 2.7967
Epoch 173/250
8/8            0s 15ms/step - loss:
2.7139 - val_loss: 2.8421
Epoch 174/250
8/8            0s 15ms/step - loss:
2.9863 - val_loss: 2.7652
Epoch 175/250
8/8            0s 16ms/step - loss:
2.9558 - val_loss: 3.3884
Epoch 176/250
8/8            0s 16ms/step - loss:
3.1170 - val_loss: 2.9864
Epoch 177/250
8/8            0s 15ms/step - loss:
2.9461 - val_loss: 2.9386
Epoch 178/250
8/8            0s 15ms/step - loss:
2.7909 - val_loss: 2.6372
Epoch 179/250
8/8            0s 16ms/step - loss:
```

```
2.6983 - val_loss: 2.7370
Epoch 180/250
8/8            0s 16ms/step - loss:
2.7791 - val_loss: 3.4099
Epoch 181/250
8/8            0s 15ms/step - loss:
2.9247 - val_loss: 2.6719
Epoch 182/250
8/8            0s 17ms/step - loss:
2.7237 - val_loss: 2.9372
Epoch 183/250
8/8            0s 15ms/step - loss:
2.9845 - val_loss: 4.0785
Epoch 184/250
8/8            0s 14ms/step - loss:
3.1276 - val_loss: 2.6753
Epoch 185/250
8/8            0s 15ms/step - loss:
2.8063 - val_loss: 2.7358
Epoch 186/250
8/8            0s 15ms/step - loss:
2.5269 - val_loss: 2.5702
Epoch 187/250
8/8            0s 16ms/step - loss:
2.6094 - val_loss: 3.0553
Epoch 188/250
8/8            0s 15ms/step - loss:
2.6906 - val_loss: 2.5969
Epoch 189/250
8/8            0s 24ms/step - loss:
2.7352 - val_loss: 3.8584
Epoch 190/250
8/8            0s 18ms/step - loss:
3.4634 - val_loss: 3.1650
Epoch 191/250
8/8            0s 14ms/step - loss:
2.9326 - val_loss: 3.0471
Epoch 192/250
8/8            0s 15ms/step - loss:
2.6020 - val_loss: 2.5491
Epoch 193/250
8/8            0s 16ms/step - loss:
2.6587 - val_loss: 2.5072
Epoch 194/250
8/8            0s 19ms/step - loss:
2.5715 - val_loss: 2.7773
Epoch 195/250
8/8            0s 16ms/step - loss:
```

```
2.5824 - val_loss: 2.4678
Epoch 196/250
8/8            0s 15ms/step - loss:
2.7100 - val_loss: 2.9512
Epoch 197/250
8/8            0s 22ms/step - loss:
2.7270 - val_loss: 2.7320
Epoch 198/250
8/8            0s 22ms/step - loss:
2.8113 - val_loss: 2.5093
Epoch 199/250
8/8            0s 22ms/step - loss:
2.6001 - val_loss: 2.4332
Epoch 200/250
8/8            0s 22ms/step - loss:
2.5585 - val_loss: 3.1570
Epoch 201/250
8/8            0s 19ms/step - loss:
2.7650 - val_loss: 3.8658
Epoch 202/250
8/8            0s 18ms/step - loss:
3.5460 - val_loss: 3.4019
Epoch 203/250
8/8            0s 19ms/step - loss:
2.9419 - val_loss: 3.0953
Epoch 204/250
8/8            0s 15ms/step - loss:
2.8190 - val_loss: 2.4361
Epoch 205/250
8/8            0s 25ms/step - loss:
2.6013 - val_loss: 2.5263
Epoch 206/250
8/8            0s 20ms/step - loss:
2.5209 - val_loss: 2.4802
Epoch 207/250
8/8            0s 15ms/step - loss:
2.4443 - val_loss: 2.4547
Epoch 208/250
8/8            0s 15ms/step - loss:
2.4868 - val_loss: 2.4380
Epoch 209/250
8/8            0s 15ms/step - loss:
2.4755 - val_loss: 2.3213
Epoch 210/250
8/8            0s 15ms/step - loss:
2.4209 - val_loss: 2.6253
Epoch 211/250
8/8            0s 15ms/step - loss:
```

```
2.5898 - val_loss: 2.6312
Epoch 212/250
8/8            0s 16ms/step - loss:
2.5716 - val_loss: 2.5715
Epoch 213/250
8/8            0s 15ms/step - loss:
2.4552 - val_loss: 2.3571
Epoch 214/250
8/8            0s 14ms/step - loss:
2.4012 - val_loss: 2.2782
Epoch 215/250
8/8            0s 15ms/step - loss:
2.2899 - val_loss: 2.3827
Epoch 216/250
8/8            0s 15ms/step - loss:
2.2734 - val_loss: 2.4058
Epoch 217/250
8/8            0s 15ms/step - loss:
2.3307 - val_loss: 2.3985
Epoch 218/250
8/8            0s 16ms/step - loss:
2.3093 - val_loss: 2.2692
Epoch 219/250
8/8            0s 15ms/step - loss:
2.4100 - val_loss: 2.2661
Epoch 220/250
8/8            0s 16ms/step - loss:
2.3534 - val_loss: 2.5791
Epoch 221/250
8/8            0s 15ms/step - loss:
2.2932 - val_loss: 2.4603
Epoch 222/250
8/8            0s 15ms/step - loss:
2.3096 - val_loss: 2.2076
Epoch 223/250
8/8            0s 15ms/step - loss:
2.1404 - val_loss: 2.1823
Epoch 224/250
8/8            0s 15ms/step - loss:
2.2498 - val_loss: 2.4179
Epoch 225/250
8/8            0s 16ms/step - loss:
2.3597 - val_loss: 2.2973
Epoch 226/250
8/8            0s 16ms/step - loss:
2.1776 - val_loss: 2.3105
Epoch 227/250
8/8            0s 15ms/step - loss:
```

```
2.2629 - val_loss: 2.3691
Epoch 228/250
8/8            0s 16ms/step - loss:
2.3955 - val_loss: 2.3406
Epoch 229/250
8/8            0s 15ms/step - loss:
2.1555 - val_loss: 2.1432
Epoch 230/250
8/8            0s 17ms/step - loss:
2.2205 - val_loss: 2.1663
Epoch 231/250
8/8            0s 16ms/step - loss:
2.3479 - val_loss: 2.4674
Epoch 232/250
8/8            0s 16ms/step - loss:
2.4346 - val_loss: 2.4957
Epoch 233/250
8/8            0s 17ms/step - loss:
2.3167 - val_loss: 2.2012
Epoch 234/250
8/8            0s 16ms/step - loss:
2.2350 - val_loss: 2.0767
Epoch 235/250
8/8            0s 33ms/step - loss:
2.1318 - val_loss: 2.3862
Epoch 236/250
8/8            0s 19ms/step - loss:
2.2365 - val_loss: 2.1454
Epoch 237/250
8/8            0s 15ms/step - loss:
2.1377 - val_loss: 2.5488
Epoch 238/250
8/8            0s 15ms/step - loss:
2.2770 - val_loss: 2.4009
Epoch 239/250
8/8            0s 15ms/step - loss:
2.2837 - val_loss: 2.2970
Epoch 240/250
8/8            0s 15ms/step - loss:
2.2123 - val_loss: 2.1593
Epoch 241/250
8/8            0s 15ms/step - loss:
2.1554 - val_loss: 2.0149
Epoch 242/250
8/8            0s 16ms/step - loss:
2.1401 - val_loss: 3.0589
Epoch 243/250
8/8            0s 16ms/step - loss:
```

```
2.4515 - val_loss: 2.1788
Epoch 244/250
8/8             0s 15ms/step - loss:
2.0785 - val_loss: 2.1511
Epoch 245/250
8/8             0s 15ms/step - loss:
2.0779 - val_loss: 1.9913
Epoch 246/250
8/8             0s 15ms/step - loss:
1.9923 - val_loss: 2.2492
Epoch 247/250
8/8             0s 16ms/step - loss:
2.2093 - val_loss: 2.2675
Epoch 248/250
8/8             0s 16ms/step - loss:
2.1973 - val_loss: 2.1014
Epoch 249/250
8/8             0s 15ms/step - loss:
2.2451 - val_loss: 2.3353
Epoch 250/250
8/8             0s 15ms/step - loss:
2.2593 - val_loss: 2.6945
```

[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
```

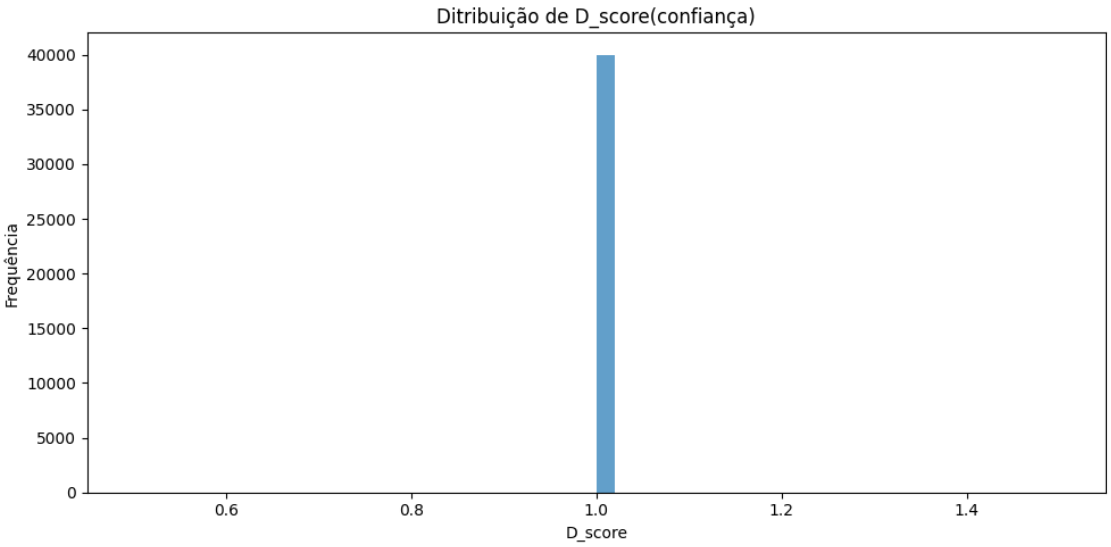
[6]:
```
# ================================
# 5. Dividir ambiente em zonas de 4m^2 (2x2m) e selecionar 1000 amostras
#    (X, Y) para cada zona
```

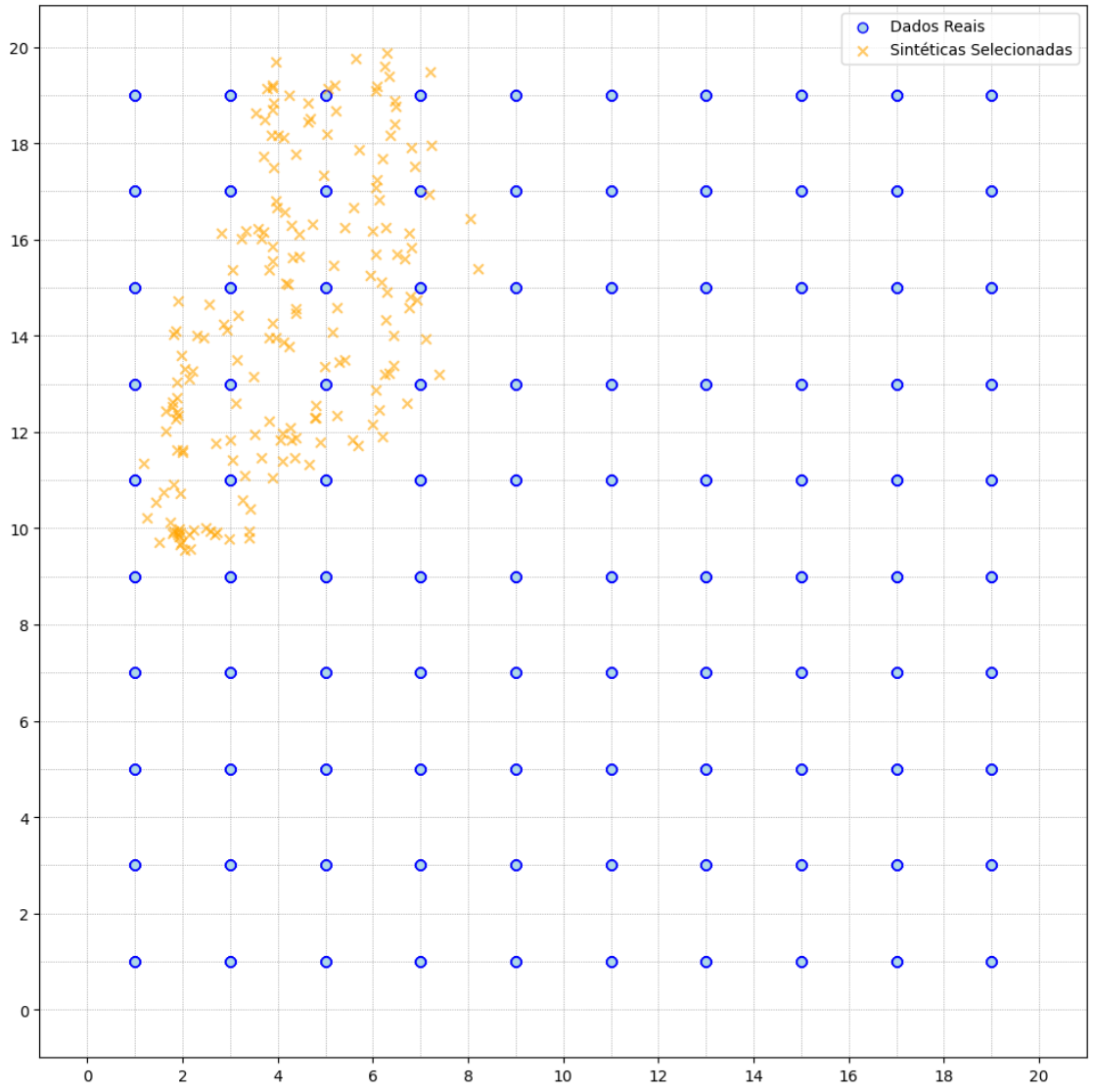
```
# ================================
L, W = 20, 20
zone_size = 2
nx, ny = int(L / zone_size), int(W / zone_size)
num_zones = nx * ny
ms = 1000
nj = ms // num_zones

# 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}")

# Histograma de confiança (D_score)
plt.figure(figsize=(10, 5))
plt.hist(df_generated['D_score'], bins=50, alpha=0.7, label='Geradas')
plt.title("Ditribuição de D_score(confiança)")
plt.xlabel("D_score")
plt.ylabel("Frequência")
plt.tight_layout()
plt.show()

# Mapa de Cobertura Espacial
plt.figure(figsize=(10, 10))
plt.scatter(df_real['X'], df_real['Y'],facecolors='lightblue',␣
 ↪edgecolors='blue', s=40, label='Dados Reais')
```

```
plt.scatter(df_selected['X'], df_selected['Y'], c='orange', marker='x', s=40,␣
 ↪alpha=0.6, label='Sintéticas Selecionadas')
# Sublinhas tracejadas de 1 em 1 metro (por cima dos pontos)
for x in np.arange(0, 21, 1):
    plt.axvline(x, color='gray', linestyle=':', linewidth=0.5, zorder=0)

for y in np.arange(0, 21, 1):
    plt.axhline(y, color='gray', linestyle=':', linewidth=0.5, zorder=0)
plt.xticks(np.arange(0, 21, 2))
plt.yticks(np.arange(0, 21, 2))
plt.legend(); plt.tight_layout(); plt.show()
```

```
Total sintéticas selecionadas: 186
Zona 040: 15 geradas, D_score médio selecionadas 1.000
Zona 041: 44 geradas, D_score médio selecionadas 1.000
Zona 050: 215 geradas, D_score médio selecionadas 1.000
Zona 051: 4007 geradas, D_score médio selecionadas 1.000
Zona 052: 1624 geradas, D_score médio selecionadas 1.000
Zona 053: 1 geradas, D_score médio selecionadas 1.000
Zona 060: 141 geradas, D_score médio selecionadas 1.000
Zona 061: 6654 geradas, D_score médio selecionadas 1.000
Zona 062: 7858 geradas, D_score médio selecionadas 1.000
Zona 063: 519 geradas, D_score médio selecionadas 1.000
Zona 070: 3 geradas, D_score médio selecionadas 1.000
Zona 071: 3803 geradas, D_score médio selecionadas 1.000
Zona 072: 7494 geradas, D_score médio selecionadas 1.000
Zona 073: 1060 geradas, D_score médio selecionadas 1.000
Zona 074: 1 geradas, D_score médio selecionadas 1.000
Zona 081: 983 geradas, D_score médio selecionadas 1.000
Zona 082: 3823 geradas, D_score médio selecionadas 1.000
Zona 083: 561 geradas, D_score médio selecionadas 1.000
Zona 084: 1 geradas, D_score médio selecionadas 1.000
Zona 091: 76 geradas, D_score médio selecionadas 1.000
Zona 092: 975 geradas, D_score médio selecionadas 1.000
Zona 093: 142 geradas, D_score médio selecionadas 1.000
```


Ditribuição de D_score(confiança)
Frequência
D_score
0
5000
10000
15000
20000
25000
30000
35000
40000
0.6
0.8
1.0
1.2
1.4



[8]:
```
import pandas as pd
import numpy as np
import tensorflow as tf

# 1) Carrega as amostras geradas
df_generated = pd.read_csv('/home/darkcover/Documentos/Gan/Data/df_generated.
 ↪csv')
wap_columns = [c for c in df_generated.columns if c.startswith('WAP')]

X_gen = df_generated[wap_columns].values.astype(np.float32)

# 2) Carrega o modelo regressor treinado nos dados reais
#    (ex: treinado em X_real,y_real na própria Fase 3)
```

```
model = tf.keras.models.load_model(
    '/home/darkcover/Documentos/Gan/Models/pseudo_label_dnn.h5',
    compile=False
)

# 3) Prediz as coordenadas
pred_coords = model.predict(X_gen, verbose=0)

# 4) Anexa as colunas de pseudo-rótulo
df_generated['LONGITUDE'] = pred_coords[:,0]
df_generated['LATITUDE']  = pred_coords[:,1]

# 5) Salva de volta para uso na Fase 4
df_generated.to_csv('/home/darkcover/Documentos/Gan/Data/df_generated_pseudo.
 ↪csv', index=False)
```

[9]:
```
df_generated.describe()
```

[9]:

	WAP001	WAP002	WAP003	WAP004	WAP005
count	40000.000000	40000.000000	40000.000000	40000.000000	40000.000000
mean	-72.304300	-66.314075	-84.710575	-55.601425	-91.486500
std	2.072235	2.581579	3.678491	3.424910	3.693394
min	-80.000000	-74.000000	-100.000000	-64.000000	-107.000000
25%	-74.000000	-68.000000	-87.000000	-58.000000	-94.000000
50%	-72.000000	-66.000000	-84.000000	-56.000000	-91.000000
75%	-71.000000	-65.000000	-82.000000	-53.000000	-89.000000
max	-65.000000	-57.000000	-76.000000	-40.000000	-80.000000

	WAP006	WAP007	WAP008	WAP009	WAP010
count	40000.0000	40000.000000	40000.000000	40000.000000	40000.000000
mean	-56.7046	-86.928525	-59.681625	-80.286775	-92.850525
std	3.7596	4.390803	2.787877	3.347576	3.430230
min	-66.0000	-107.000000	-70.000000	-95.000000	-110.000000
25%	-60.0000	-90.000000	-62.000000	-82.000000	-95.000000
50%	-57.0000	-87.000000	-60.000000	-80.000000	-92.000000
75%	-54.0000	-84.000000	-58.000000	-78.000000	-90.000000
max	-42.0000	-75.000000	-49.000000	-73.000000	-85.000000

	LONGITUDE	LATITUDE
count	40000.000000	40000.000000
mean	2.852250	12.679367
std	1.591303	1.664991
min	-0.164581	8.877346
25%	1.526815	11.362357
50%	2.732364	12.441347
75%	3.963216	13.768739
max	8.701089	19.392925

[10]:
```
df_generated.head()
```

[10]:

	WAP001	WAP002	WAP003	WAP004	WAP005	WAP006	WAP007	WAP008	WAP009	\
0	-73	-62	-91	-54	-90	-60	-89	-62	-83	
1	-69	-69	-80	-55	-93	-48	-88	-55	-79	
2	-71	-63	-90	-55	-88	-59	-90	-59	-79	
3	-70	-66	-81	-58	-86	-53	-84	-60	-81	
4	-70	-70	-78	-60	-90	-57	-83	-59	-78	

	WAP010	LONGITUDE	LATITUDE
0	-90	3.749696	14.222166
1	-98	1.526628	12.504369
2	-88	2.897660	14.263527
3	-91	3.809544	12.450233
4	-92	2.807864	10.858861