

fase3_1

May 18, 2025

```
[1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

```
2025-05-18 17:57:21.439904: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32]
Could not find cuda drivers on your machine, GPU will not be used.
2025-05-18 17:57:21.650285: I external/local_xla/xla/tsl/cuda/cudart_stub.cc:32]
Could not find cuda drivers on your machine, GPU will not be used.
2025-05-18 17:57:21.853118: 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 00:00:1747605441.955516    23238 cuda_dnn.cc:8579] Unable to register cuDNN
factory: Attempting to register factory for plugin cuDNN when one has already
been registered
E0000 00:00:1747605441.986077    23238 cuda_blas.cc:1407] Unable to register
cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has
already been registered
W0000 00:00:1747605442.114872    23238 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747605442.114932    23238 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747605442.114936    23238 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
W0000 00:00:1747605442.114939    23238 computation_placer.cc:177] computation
placer already registered. Please check linkage and avoid linking the same
target more than once.
2025-05-18 17:57:22.139672: 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.

```
[6]: # =====
# 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)
```

```
[7]: # =====
# 2. Treinar rede DNN para pseudo-label
# =====
X_train, X_val, y_train, y_val = train_test_split(X_real, y_real, test_size=0.
↳2, random_state=42)

model_dnn = Sequential([
    Dense(30, activation='relu', input_shape=(10,)),
    Dense(20, activation='relu'),
    Dense(2)
])
model_dnn.compile(optimizer=Adam(0.01), loss='mse')
model_dnn.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=200,↳
↳batch_size=50, verbose=0)
```

/home/darkcover/.cache/pypoetry/virtualenvs/gan-oPyfrVEv-

py3.12/lib/python3.12/site-packages/keras/src/layers/core/dense.py:87:

UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.

super().__init__(activity_regularizer=activity_regularizer, **kwargs)

2025-05-18 17:58:59.947858: 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)

```
[7]: <keras.src.callbacks.history.History at 0x73d161b385f0>
```

```
[8]: # =====
# 3. Pseudo-label nos vetores gerados
# =====
X_gen = df_generated.iloc[:, :10].values.astype(np.float32)
pseudo_coords = model_dnn.predict(X_gen, verbose=1)
df_generated[["X", "Y"]] = pseudo_coords
```

1250/1250

3s 2ms/step

```
[9]: # =====  
# 4. Reproduzir Figura 4  
# =====  
sample_1000 = df_generated.iloc[:1000]  
  
plt.figure(figsize=(6.5, 6.5))  
plt.scatter(sample_1000['X'], sample_1000['Y'],  
            c='purple', alpha=0.6, s=15, label='Pseudo-labeled positions')  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.title("FIGURE 4. 1000 positions generated by the GAN with pseudo labels.")  
plt.grid(True, linestyle='--', alpha=0.5)  
plt.axis('equal')  
plt.xlim(0, 20)  
plt.ylim(0, 20)  
plt.tight_layout()  
plt.show()
```

Ignoring fixed y limits to fulfill fixed data aspect with adjustable data limits.

Ignoring fixed x limits to fulfill fixed data aspect with adjustable data limits.

FIGURE 4. 1000 positions generated by the GAN with pseudo labels.

