Topik : 2.3. Evaluasi dan Plotting

Objective : Membuat grafik akurasi/loss per round

Task : Tambahkan visualisasi (matplotlib)

Source : https://www.tensorflow.org/federated/tutorials/tutorials_overview

Getting started with federated learning

Federated Learning for image classification	Memperkenalkan bagian – bagian utama dari API Federated Learning (FL), dan mendemonstrasikan cara memakai TFF untuk mensimulasikan federated learning pada data mirip MNIST.
Federated Learning for text generation	Menunjukkan lebih lanjut cara memakai API FL milik TFF untuk menyempurnakan (refine) model pra-latih terserialisasi pada tugas pemodelan Bahasa
Tuning recommended aggregations for learning	Memperlihatkan bagaimana komputasi FL dasar di tff.learning dapat digabungkan dengan rutin agregasi khusus yang menawarkan kekokohan, privasi diferensial, kompresi dan lainnya.
Federated Recontruction for Matrix Factorization	Memperkenalkan federated learning yang Sebagian local , Dimana Sebagian parameter klien tidak pernah di agregasi di server.

Getting Started With Federated Analytics

Private Heavy Hitters	Menunjukkan cara menggunakan tff.analytics.heavy_hitters untuk membangun komputasi analitik terfederasi guna menemukan “heavy hitters” privat (item yang paling sering menonjol dengan perlindungan privasi)

Writing Custom Federated Computations

Building your own federated learning algoritm	Menunjukkan cara memakai TFF core API untuk mengimplementasikan algoritma federated learning dengan federated averaging sebagai contoh
Compossing learning algorithms	Menunjukkan cara memakai TFF learning API untuk dengan mudah mengimplementasikan algoritma federated learning baru, khususnya berbagai varian federated averaging
Custom Federated algorithms (introduction to the federated core & implementing federated averaging)	Memperkenalkan konsep – konsep kunci serta antarmuka yang disediakan oleh Federated Core API (FC API)_
Implementing custom aggregations	Menjelaskan prinsip desain dibalik modul tff.aggregators dan praktik terbaik untuk mengimplementasikan agregasi nilai dari klien ke server.

Simulation best pratices

TFF simulation with accelerators (GPU)	Menunjukkan bagaimana runtime berkinerja tinggi milik TFF dapat digunakan dengan GPU
Working with ClientData	Memberikan praktik terbaik untuk mengintergrasikan dataset simulasi berbasis **ClientData** milik TFF ke dalam komputasi TFF

Intermediate and advanced tutorials

Random noise generation	Menyoroti beberapa kehalusan saat menggunakan keacakan dalam komputasi terdesentralisasi, serta mengusulkan praktik terbaik dan pola yang direkomendasikam
Sending Different Data To Particular Clients with federated_language.federated_select	Memperkenalkan operator federated_language.federated_select dan memberi contoh sederhana algoritma federated kustom yang mengirim data berbeda ke klien yang berbeda
Client-efficient large-model federated learning via federated_select and sparse aggregation	Menunjukkan bagaimana TFF dapat digunakan untuk melatih model yang sangat besar, Dimana setiap perangkat klien hanya mengunduh dan memperbarui sebagaian kecil dari model menggunakan federated_langauge.federated_select dan agregasi sparse.
Federated Learning with Differential Privacy in TFF	Mendemonstrasikan cara menggunakan TFF untuk melatih model dengan privasi differensial Tingkat pengguna.

Matplotlib Example :

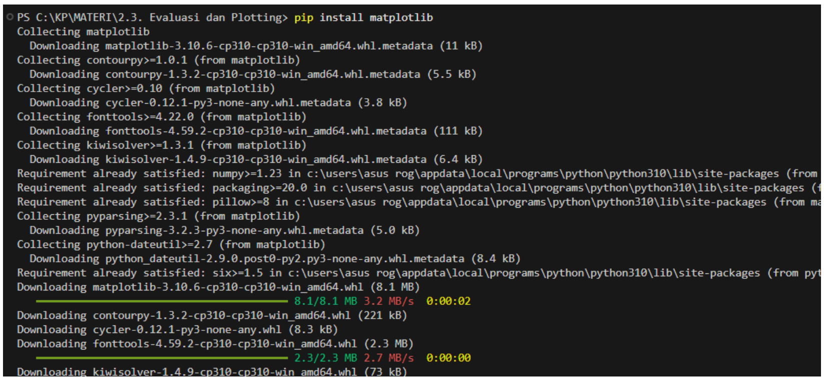
Source : https://matplotlib.org/stable/gallery/index.html

Menerapkan 2 contoh yangb akan digunakan visualisasi akurasi/loss per round dan yang paling umum untuk dipakai :

1. Line Plot
    - Cocok untuk melihat tren naik/turun dari round ke round
    - Biasanya dibuat dua figure terpisah (satu untuk Accuracy , satu untuk Loss)
2. Log scale untuk loss
    - Dipakai kalau nilai loss turun tajam (rentang besar), supaya kurva bisa lebih terbaca

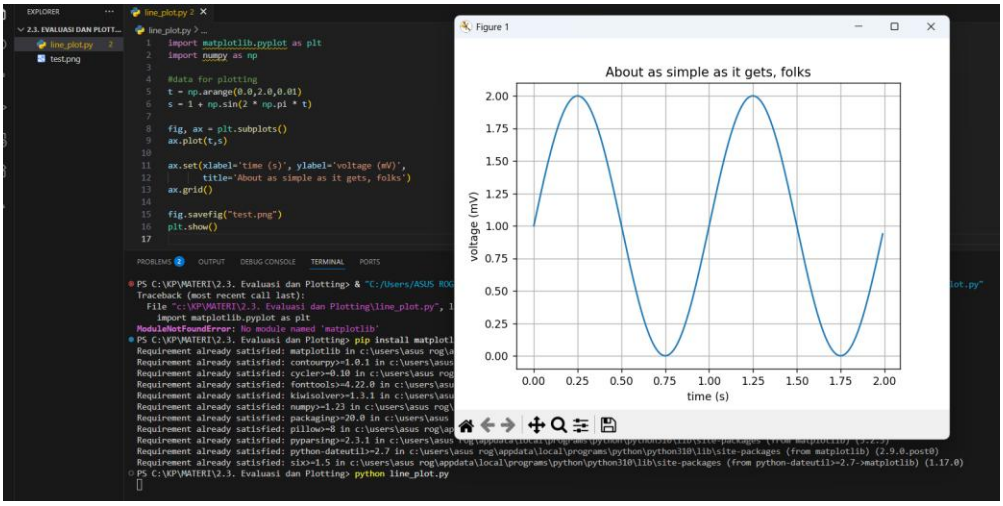
Example : Uji Coba

Install Matplotlib



```
○ PS C:\KP\MATERI\2.3. Evaluasi dan Plotting> pip install matplotlib
  Collecting matplotlib
    Downloading matplotlib-3.10.6-cp310-cp310-win_amd64.whl.metadata (11 kB)
  Collecting contourpy>=1.0.1 (from matplotlib)
    Downloading contourpy-1.3.2-cp310-cp310-win_amd64.whl.metadata (5.5 kB)
  Collecting cycler>=0.10 (from matplotlib)
    Downloading cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
  Collecting fonttools>=4.22.0 (from matplotlib)
    Downloading fonttools-4.59.2-cp310-cp310-win_amd64.whl.metadata (111 kB)
  Collecting kiwisolver>=1.3.1 (from matplotlib)
    Downloading kiwisolver-1.4.9-cp310-cp310-win_amd64.whl.metadata (6.4 kB)
  Requirement already satisfied: numpy>=1.23 in c:\users\asus rog\appdata\local\programs\python\python310\lib\site-packages (from
  Requirement already satisfied: packaging>=20.0 in c:\users\asus rog\appdata\local\programs\python\python310\lib\site-packages (f
  Requirement already satisfied: pillow>=8 in c:\users\asus rog\appdata\local\programs\python\python310\lib\site-packages (from ma
  Collecting pyparsing>=2.3.1 (from matplotlib)
    Downloading pyparsing-3.2.3-py3-none-any.whl.metadata (5.0 kB)
  Collecting python-dateutil>=2.7 (from matplotlib)
    Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
  Requirement already satisfied: six>=1.5 in c:\users\asus rog\appdata\local\programs\python\python310\lib\site-packages (from pyt
  Downloading matplotlib-3.10.6-cp310-cp310-win_amd64.whl (8.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.1/8.1 MB 3.2 MB/s  0:00:02
  Downloading contourpy-1.3.2-cp310-cp310-win_amd64.whl (221 kB)
  Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
  Downloading fonttools-4.59.2-cp310-cp310-win_amd64.whl (2.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.3/2.3 MB 2.7 MB/s  0:00:00
  Downloading kiwisolver-1.4.9-cp310-cp310-win_amd64.whl (73 kB)
```

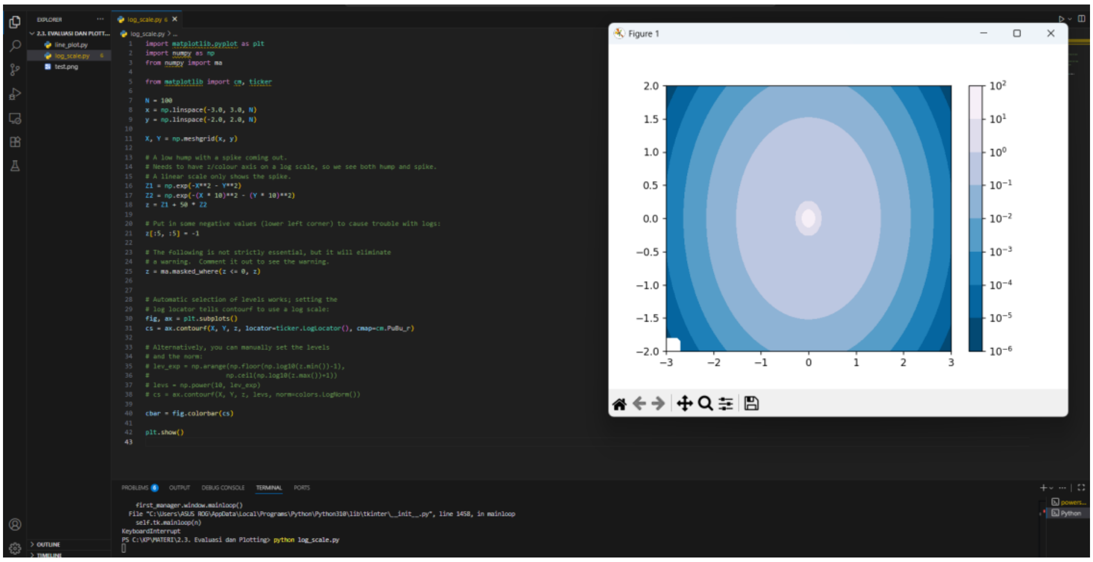
Line Plot



```python
import matplotlib.pyplot as plt
import numpy as np

#data for plotting
t = np.arange(0.0,2.0,0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t,s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```

```
● PS C:\KP\MATERI\2.3. Evaluasi dan Plotting> & "C:/Users/ASUS ROG                                                    lot.py"
  Traceback (most recent call last):
    File "c:\KP\MATERI\2.3. Evaluasi dan Plotting\line_plot.py", l
      import matplotlib.pyplot as plt
  ModuleNotFoundError: No module named 'matplotlib'
● PS C:\KP\MATERI\2.3. Evaluasi dan Plotting> pip install matplotl
  Requirement already satisfied: matplotlib in c:\users\asus rog\a
  Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus
  Requirement already satisfied: cycler>=0.10 in c:\users\asus rog
  Requirement already satisfied: fonttools>=4.22.0 in c:\users\asu
  Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\asu
  Requirement already satisfied: numpy>=1.23 in c:\users\asus rog\
  Requirement already satisfied: packaging>=20.0 in c:\users\asus
  Requirement already satisfied: pillow>=8 in c:\users\asus rog\ap
  Requirement already satisfied: pyparsing>=2.3.1 in c:\users\asus
  Requirement already satisfied: python-dateutil>=2.7 in c:\users\asus rog\appdata\local\programs\python\python310\lib\site-packages (from matplotlib) (2.9.0.post0)
  Requirement already satisfied: six>=1.5 in c:\users\asus rog\appdata\local\programs\python\python310\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
○ PS C:\KP\MATERI\2.3. Evaluasi dan Plotting> python line_plot.py
  ▯
```

LogScale

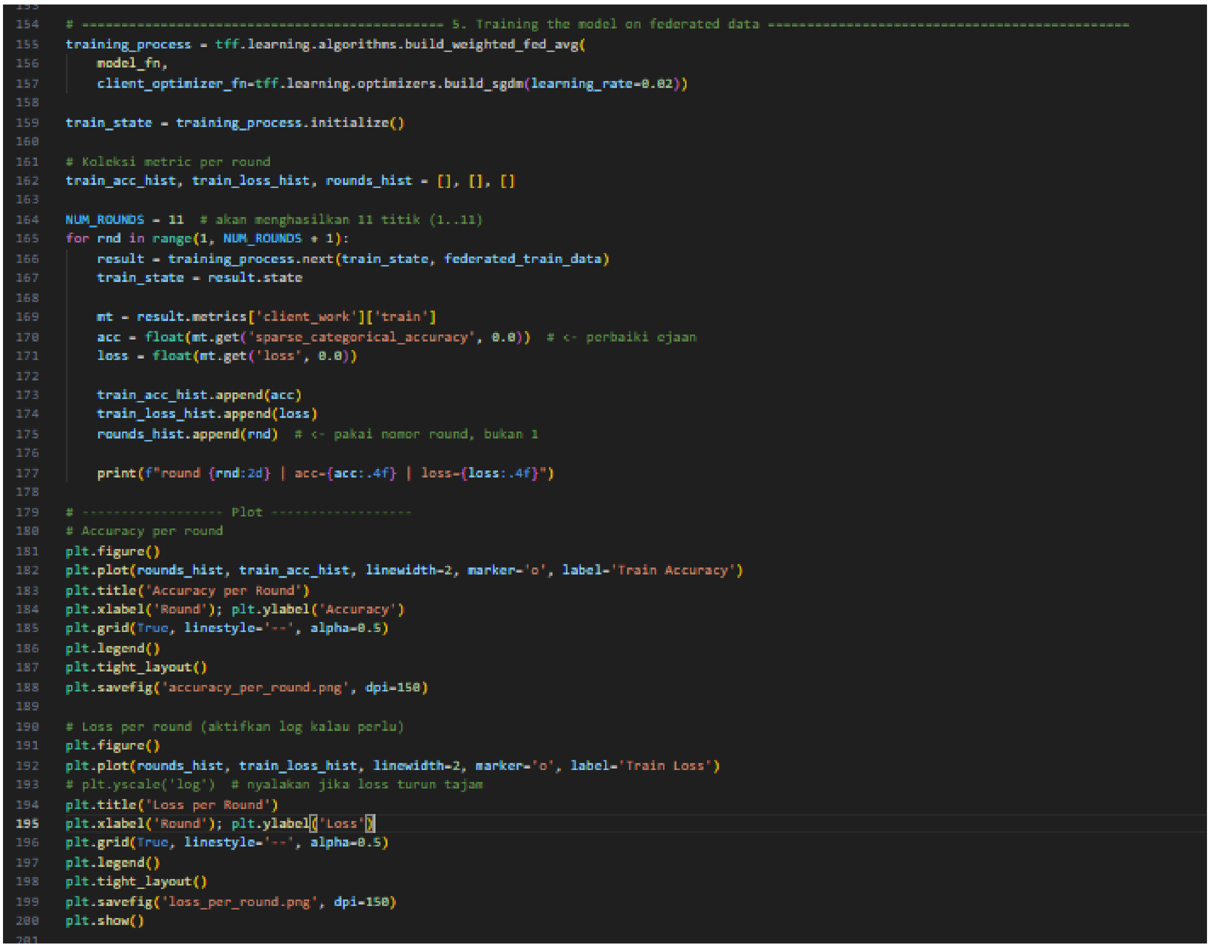


```python
import matplotlib.pyplot as plt
import numpy as np
from numpy import ma

from matplotlib import cm, ticker

N = 100
x = np.linspace(-3.0, 3.0, N)
y = np.linspace(-2.0, 2.0, N)

X, Y = np.meshgrid(x, y)

# A low hump with a spike coming out.
# Needs to have z/colour axis on a log scale, so we see both hump and spike.
# A linear scale only shows the spike.
Z1 = np.exp(-X**2 - Y**2)
Z2 = np.exp(-(X * 10)**2 - (Y * 10)**2)
z = Z1 + 50 * Z2

# Put in some negative values (lower left corner) to cause trouble with logs:
z[:5, :5] = -1

# The following is not strictly essential, but it will eliminate
# a warning.  Comment it out to see the warning.
z = ma.masked_where(z <= 0, z)


# Automatic selection of levels works; setting the
# log locator tells contourf to use a log scale:
fig, ax = plt.subplots()
cs = ax.contourf(X, Y, z, locator=ticker.LogLocator(), cmap=cm.PuBu_r)

# Alternatively, you can manually set the levels
# and the norm:
# lev_exp = np.arange(np.floor(np.log10(z.min())-1),
#                    np.ceil(np.log10(z.max())+1))
# levs = np.power(10, lev_exp)
# cs = ax.contourf(X, Y, z, levs, norm=colors.LogNorm())

cbar = fig.colorbar(cs)

plt.show()
```

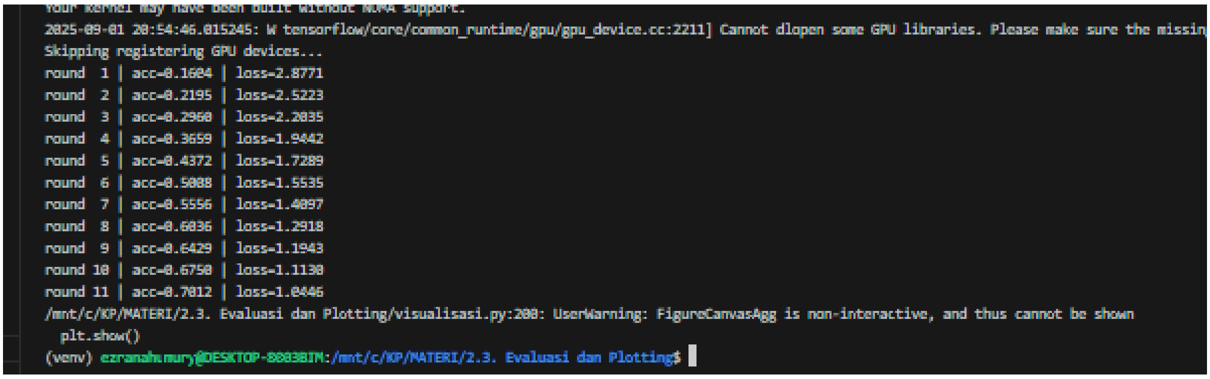
```
    first_manager.window.mainloop()
  File "C:\Users\ASUS ROG\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 1458, in mainloop
    self.tk.mainloop(n)
KeyboardInterrupt
PS C:\KP\MATERI\2.3. Evaluasi dan Plotting> python log_scale.py
▯
```

Task

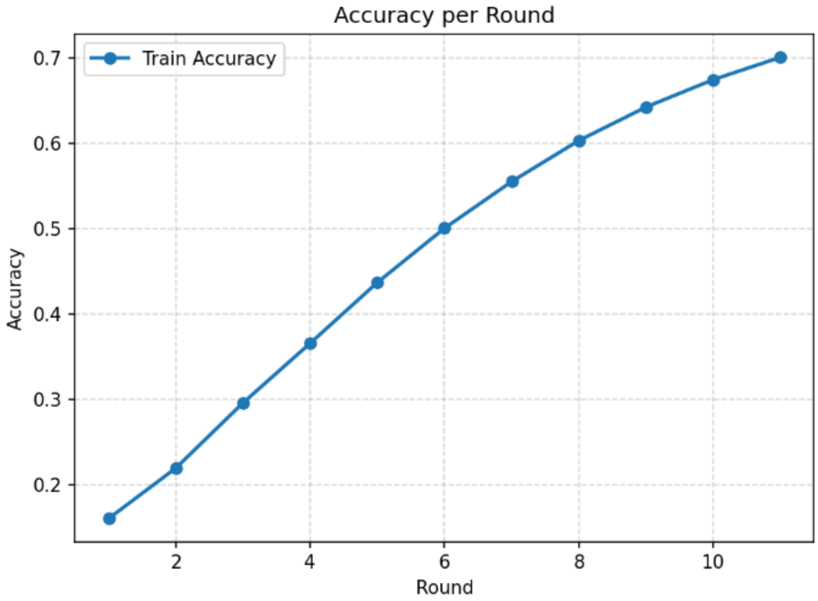
Visualisasi Line Plot :

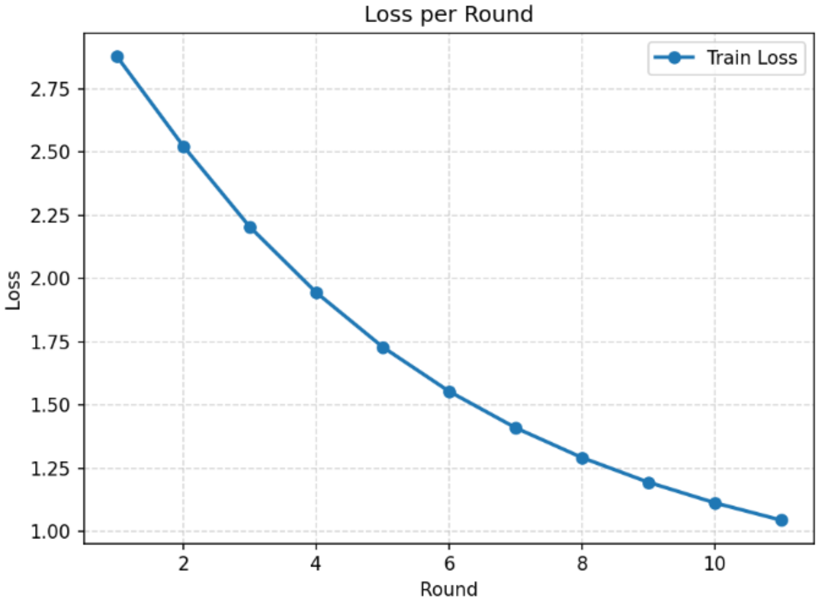


```
# ============================================== 5. Training the model on federated data ==============================================
training_process = tff.learning.algorithms.build_weighted_fed_avg(
    model_fn,
    client_optimizer_fn=tff.learning.optimizers.build_sgdm(learning_rate=0.02))

train_state = training_process.initialize()

# Koleksi metric per round
train_acc_hist, train_loss_hist, rounds_hist = [], [], []

NUM_ROUNDS = 11  # akan menghasilkan 11 titik (1..11)
for rnd in range(1, NUM_ROUNDS + 1):
    result = training_process.next(train_state, federated_train_data)
    train_state = result.state

    mt = result.metrics['client_work']['train']
    acc = float(mt.get('sparse_categorical_accuracy', 0.0))  # <- perbaiki ejaan
    loss = float(mt.get('loss', 0.0))

    train_acc_hist.append(acc)
    train_loss_hist.append(loss)
    rounds_hist.append(rnd)  # <- pakai nomor round, bukan 1

    print(f"round {rnd:2d} | acc={acc:.4f} | loss={loss:.4f}")

# ------------------ Plot ------------------
# Accuracy per round
plt.figure()
plt.plot(rounds_hist, train_acc_hist, linewidth=2, marker='o', label='Train Accuracy')
plt.title('Accuracy per Round')
plt.xlabel('Round'); plt.ylabel('Accuracy')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.tight_layout()
plt.savefig('accuracy_per_round.png', dpi=150)

# Loss per round (aktifkan log kalau perlu)
plt.figure()
plt.plot(rounds_hist, train_loss_hist, linewidth=2, marker='o', label='Train Loss')
# plt.yscale('log')  # nyalakan jika loss turun tajam
plt.title('Loss per Round')
plt.xlabel('Round'); plt.ylabel('Loss')
plt.grid(True, linestyle='--', alpha=0.5)
plt.legend()
plt.tight_layout()
plt.savefig('loss_per_round.png', dpi=150)
plt.show()
```

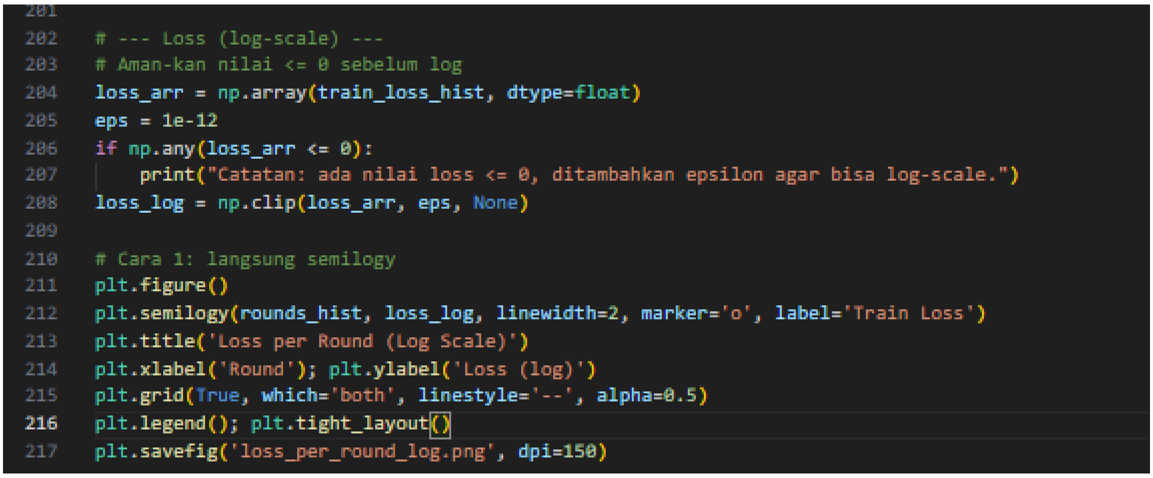


```
Your kernel may have been built without NUMA support.
2025-09-01 20:54:46.015245: W tensorflow/core/common_runtime/gpu/gpu_device.cc:2211] Cannot dlopen some GPU libraries. Please make sure the missin
Skipping registering GPU devices...
round  1 | acc=0.1604 | loss=2.8771
round  2 | acc=0.2195 | loss=2.5223
round  3 | acc=0.2960 | loss=2.2035
round  4 | acc=0.3659 | loss=1.9442
round  5 | acc=0.4372 | loss=1.7289
round  6 | acc=0.5088 | loss=1.5535
round  7 | acc=0.5556 | loss=1.4097
round  8 | acc=0.6036 | loss=1.2918
round  9 | acc=0.6429 | loss=1.1943
round 10 | acc=0.6750 | loss=1.1130
round 11 | acc=0.7012 | loss=1.0446
/mnt/c/KP/MATERI/2.3. Evaluasi dan Plotting/visualisasi.py:200: UserWarning: FigureCanvasAgg is non-interactive, and thus cannot be shown
  plt.show()
(venv) ezranahumury@DESKTOP-8083BIM:/mnt/c/KP/MATERI/2.3. Evaluasi dan Plotting$
```

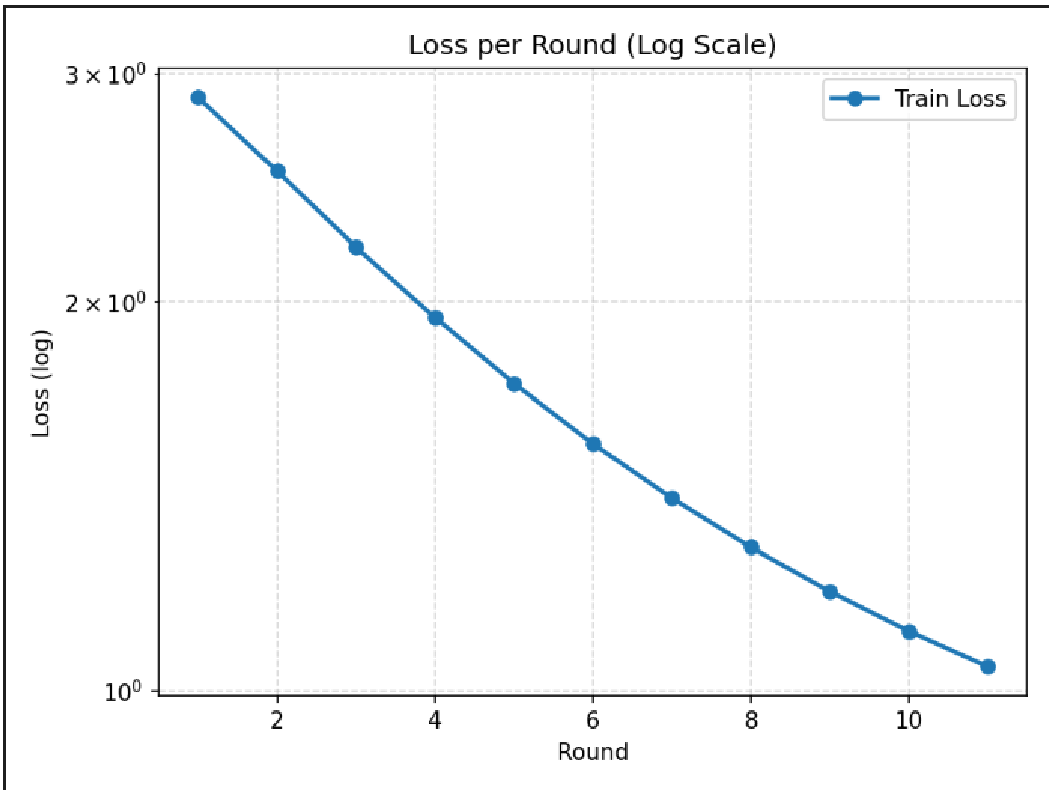

Accuracy per Round
Train Accuracy
Accuracy
0.7
0.6
0.5
0.4
0.3
0.2
2
4
6
8
10
Round


Loss per Round
Train Loss
Loss
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
2
4
6
8
10
Round

Visualisasi Log Scale



```
# --- Loss (log-scale) ---
# Aman-kan nilai <= 0 sebelum log
loss_arr = np.array(train_loss_hist, dtype=float)
eps = 1e-12
if np.any(loss_arr <= 0):
    print("Catatan: ada nilai loss <= 0, ditambahkan epsilon agar bisa log-scale.")
loss_log = np.clip(loss_arr, eps, None)

# Cara 1: langsung semilogy
plt.figure()
plt.semilogy(rounds_hist, loss_log, linewidth=2, marker='o', label='Train Loss')
plt.title('Loss per Round (Log Scale)')
plt.xlabel('Round'); plt.ylabel('Loss (log)')
plt.grid(True, which='both', linestyle='--', alpha=0.5)
plt.legend(); plt.tight_layout()
plt.savefig('loss_per_round_log.png', dpi=150)
```


Loss per Round (Log Scale)
Train Loss
3×10^0
2×10^0
10^0
Loss (log)
Round