

Goiânia, 04 de junho de 2025

NOME: DENIS NOGUEIRA DO NASCIMENTO
E-MAIL: denisnoguer@gmail.com

Título:

FEDERATED LEARNING-BASED APPROACH FOR INTRUSION DETECTION IN COMPUTER NETWORKS

ABORDAGEM BASEADA EM APRENDIZADO FEDERADO PARA A DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES.

LIST OF TECHNICAL METHODOLOGY APPLIED FOR FL.

1. Environment preparation:

- Install Docker / Docker Compose
- Create a Python environment and install libraries (Flower, TensorFlow/PyTorch, scikit-learn)

2. Data preprocessing:

- Download and partition KDD-99, MotionSense, etc.
- Encoding and normalization
- Generate partitions for FL

3. Centralized model:

- Implement and train a local MLP
- Validate reference accuracy

4. Basic FL orchestration:

- Create server.py and client.py using Flower
- Adjust docker-compose.yml for 20 clients

5. FL Experiments (IDS):

- Run FL varying N, E, and B
- Collect logs and accuracy metrics

6. DDoS extension:

- Unzip DDos.zip
- Configure environment and DDoS data
- Execute and collect results

7. ERI Project:

- Test modifications and generate new results

Analysis and visualization:

- Consolidate logs
- Generate comparative graphs (centralized vs FL, IDS vs DDoS vs ERI)

Detailed data analysis and report (methodology, results, discrepancies)

- Suggestions for future improvements

1- INSTALLATION TUTORIAL FOR DEPENDENCIES

(Note: Latest versions of Linux Ubuntu were used.)

Installing dependencies

Python (VIRTUAL)

Step 2 – Create and activate the Python virtual environment

Inside /0/IDS, run:

Step B – Install dependencies manually

2- List of Required Dependencies

- TensorFlow (MLP training)
- Flower (flwr) (federated orchestration)
- pandas, numpy, scikit-learn (preprocessing)
- matplotlib (graphs)

```
bash
# Ainda com o (venv) ativo em /0/IDS:
```

```
pip install --upgrade pip

# Instalar as bibliotecas essenciais:
pip install tensorflow flwr pandas numpy scikit-learn matplotlib
```

Print da Instalação da Dependências e atualização das bibliotecas.

```
root@denis-ASUS-TUF-Gaming:/0/IDS
Downloading namex-0.1.0-py3-none-any.whl.metadata (322 bytes)
Collecting optree (from keras>=3.5.0->tensorflow)
  Downloading optree-0.15.0-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (48 kB)
Collecting mdurl~=0.1 (from markdown-it-py>=2.2.0->rich<14.0.0,>=13.5.0->flwr)
  Downloading mdurl-0.1.2-py3-none-any.whl.metadata (1.6 kB)
Collecting MarkupSafe>=2.1.1 (from werkzeug>=1.0.1->tensorflow)
  Downloading MarkupSafe-3.0.2-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (4.0 kB)
Downloading tensorflow-2.19.0-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (645.0 MB)
  645.0/645.0 MB 37.4 MB/s eta 0:00:00
Downloading numpy-2.1.3-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (16.0 MB)
  16.0/16.0 MB 37.8 MB/s eta 0:00:00
Downloading grpcio-1.71.0-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (5.9 MB)
  5.9/5.9 MB 47.3 MB/s eta 0:00:00
Downloading ml_dtypes-0.5.1-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (4.7 MB)
  4.7/4.7 MB 36.4 MB/s eta 0:00:00
Downloading requests-2.32.3-py3-none-any.whl (64 kB)
Downloading charset_normalizer-3.4.2-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (148 kB)
Downloading idna-3.10-py3-none-any.whl (70 kB)
Downloading tensorflow-2.19.0-py3-none-any.whl (5.5 MB)
  5.5/5.5 MB 49.2 MB/s eta 0:00:00
Downloading tensorflow_data_server-0.7.2-py3-none-manylinux_2_31_x86_64.whl (6.6 MB)
  6.6/6.6 MB 37.4 MB/s eta 0:00:00
Downloading urllib3-2.4.0-py3-none-any.whl (128 kB)
Downloading flwr-1.18.0-py3-none-any.whl (540 kB)
  540.0/540.0 kB 28.2 MB/s eta 0:00:00
Downloading cryptography-44.0.3-cp39-abi3-manylinux_2_34_x86_64.whl (4.2 MB)
  4.2/4.2 MB 35.1 MB/s eta 0:00:00
Downloading iterators-0.0.2-py3-none-any.whl (3.9 kB)
Downloading pathspec-0.12.1-py3-none-any.whl (31 kB)
Downloading protobuf-4.25.7-cp37-abi3-manylinux2014_x86_64.whl (294 kB)
Downloading pycryptodome-3.23.0-cp37-abi3-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (2.3 MB)
  2.3/2.3 MB 34.3 MB/s eta 0:00:00
Downloading PyYAML-6.0.2-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (767 kB)
  767.5/767.5 KB 38.4 MB/s eta 0:00:00
Downloading rich-13.9.4-py3-none-any.whl (242 kB)
Downloading pygments-2.19.1-py3-none-any.whl (1.2 MB)
  1.2/1.2 MB 51.9 MB/s eta 0:00:00
Downloading tomli-2.2.1-cp312-cp312-manylinux_2_17_x86_64_manylinux2014_x86_64.whl (242 kB)
```

2- List of Required Dependencies

- TensorFlow (MLP training)
- Flower (flwr) (federated orchestration)
- pandas, numpy, scikit-learn (preprocessing)
- matplotlib (graphs)
- **matplotlib** (gráficos)

The terminal window shows the following output:

```

root@denis-ASUS-TUF-Gaming:/0/IDS
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Attack Type
dce      40000
normal   40000
probe    40000
r2l     1126
u2r     52
Names: count, dtype: int64
(11178, 27)
12178
(84824, 27)
(36354, 27)
/0/IDS/venv/lib/python3.12/site-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(s) shape` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
0x3000000000000000: E external/local_xla/xla/stream_executor/cuda/cuda_platform.cc:51] failed call to cuInit: INTERNAL: CUDA error: Failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
Model: "sequential"

```

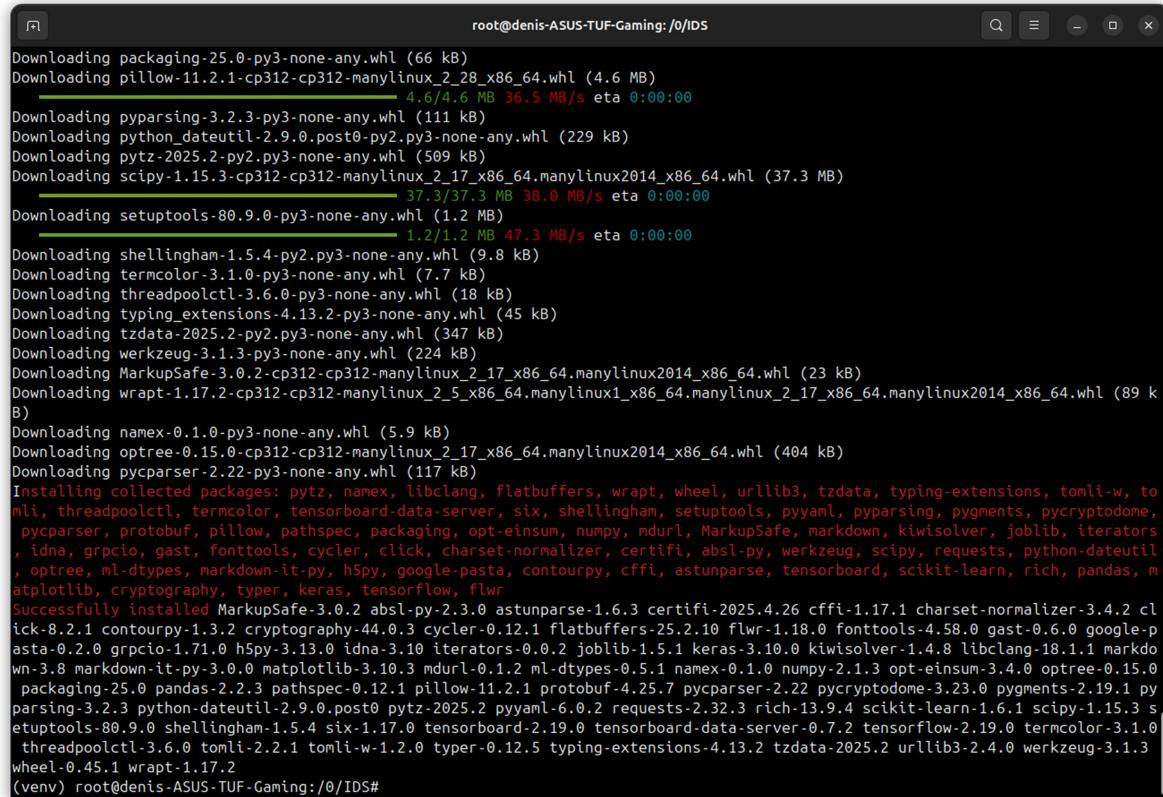
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 3)	224
dropout (Dropout)	(None, 3)	0
dense_1 (Dense)	(None, 5)	45

Total params: 269 (1.05 KB)
Trainable params: 209 (1.05 KB)
Non-trainable params: 60 (0.00 KB)

Epoch 1/50
663/663 1s 1ms/step - accuracy: 0.4356 - loss: 1.2768 - val_accuracy: 0.0704 - val_loss: 0.5885
663/663 1s 980us/step - accuracy: 0.8213 - loss: 0.6186 - val_accuracy: 0.8927 - val_loss: 0.4121
Epoch 3/50
663/663 1s 1ms/step - accuracy: 0.8629 - loss: 0.4733 - val_accuracy: 0.9302 - val_loss: 0.3392
663/663 1s 1ms/step - accuracy: 0.8794 - loss: 0.4118 - val_accuracy: 0.9331 - val_loss: 0.3046
Epoch 5/50
663/663 1s 1ms/step - accuracy: 0.8817 - loss: 0.3851 - val_accuracy: 0.9335 - val_loss: 0.2813
Epoch 6/50
663/663 1s 1ms/step - accuracy: 0.8871 - loss: 0.3028 - val_accuracy: 0.9345 - val_loss: 0.2654
663/663 1s 1ms/step - accuracy: 0.8943 - loss: 0.3502 - val_accuracy: 0.9350 - val_loss: 0.2533
Epoch 8/50
663/663 1s 1ms/step - accuracy: 0.8992 - loss: 0.3319 - val_accuracy: 0.9352 - val_loss: 0.2452
Epoch 9/50
663/663 1s 1ms/step - accuracy: 0.9053 - loss: 0.3219 - val_accuracy: 0.9357 - val_loss: 0.2378

3- List of Available Flags (such as --epochs, --batch-size, etc.)

- Centralized training (LOW)
- Configure Federated Learning
- Collect results



```
root@denis-ASUS-TUF-Gaming:/0/IDS
Downloading packaging-25.0.py3-none-any.whl (66 kB)
Downloading pillow-11.2.1-cp312-cp312-manylinux_2_28_x86_64.whl (4.6 MB)
  4.6/4.6 MB 36.5 MB/s eta 0:00:00
Downloading pyparsing-3.2.3-py3-none-any.whl (111 kB)
Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
Downloading pytz-2025.2-py2.py3-none-any.whl (509 kB)
Downloading scipy-1.15.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (37.3 MB)
  37.3/37.3 MB 38.0 MB/s eta 0:00:00
Downloading setuptools-80.9.0-py3-none-any.whl (1.2 MB)
  1.2/1.2 MB 47.3 MB/s eta 0:00:00
Downloading shellingham-1.5.4-py2.py3-none-any.whl (9.8 kB)
Downloading termcolor-3.1.0-py3-none-any.whl (7.7 kB)
Downloading threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Downloading typing_extensions-4.13.2-py3-none-any.whl (45 kB)
Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Downloading werkzeug-3.1.3-py3-none-any.whl (224 kB)
Downloading MarkupSafe-3.0.2-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (23 kB)
Downloading wrapt-1.17.2-cp312-cp312-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl (89 kB)
Downloading namex-0.1.0-py3-none-any.whl (5.9 kB)
Downloading optree-0.15.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (404 kB)
Downloading pycparser-2.22-py3-none-any.whl (117 kB)
Installing collected packages: pytz, namex, libclang, flatbuffers, wrapt, wheel, urllib3, tzdata, typing-extensions, tomli-w, toml, threadpoolctl, termcolor, tensorboard-data-server, six, shellingham, setuptools, pyyaml, pyparsing, pygments, pycryptodome, pycparser, protobuf, pillow, pathspec, packaging, opt-einsum, numpy, mdurl, MarkupSafe, markdown, kiwisolver, joblib, iterators, idna, grpcio, gast, fonttools, cycler, click, charset-normalizer, certifi, absl-py, werkzeug, scipy, requests, python-dateutil, optree, ml-dtypes, markdown-it-py, h5py, google-pasta, contourpy, cffi, astunparse, tensorboard, scikit-learn, rich, pandas, matplotlib, cryptography, typer, keras, tensorflow, flwr
Successfully installed MarkupSafe-3.0.2 absl-py-2.3.0 astunparse-1.6.3 certifi-2025.4.26 cffi-1.17.1 charset-normalizer-3.4.2 click-8.2.1 contourpy-1.3.2 cryptography-44.0.3 cycler-0.12.1 flatbuffers-25.2.10 flwr-1.18.0 fonttools-4.58.0 gast-0.6.0 google-pasta-0.2.0 grpcio-1.71.0 h5py-3.13.0 idna-3.10 iterators-0.0.2 joblib-1.5.1 keras-3.10.0 kiwisolver-1.4.8 libclang-18.1.1 markdown-3.8 markdown-it-py-3.0.0 matplotlib-3.10.3 mdurl-0.1.2 ml-dtypes-0.5.1 namex-0.1.0 numpy-2.1.3 opt-einsum-3.4.0 optree-0.15.0 packaging-25.0 pandas-2.2.3 pathspec-0.12.1 pillow-11.2.1 protobuf-4.25.7 pycparser-2.22 pycryptodome-3.23.0 pygments-2.19.1 pyParsing-3.2.3 python-dateutil-2.9.0.post0 pytz-2025.2 pyyaml-6.0.2 requests-2.32.3 rich-13.9.4 scikit-learn-1.6.1 scipy-1.15.3 setuptools-80.9.0 shellingham-1.5.4 six-1.17.0 tensorboard-2.19.0 tensorboard-data-server-0.7.2 tensorflow-2.19.0 termcolor-3.1.0 threadpoolctl-3.6.0 toml-2.2.1 toml-w-1.2.0 typer-0.12.5 typing-extensions-4.13.2 tzdata-2025.2 urllib3-2.4.0 werkzeug-3.1.3 wheel-0.45.1 wrapt-1.17.2
(venv) root@denis-ASUS-TUF-Gaming:/0/IDS#
```

4- DOCKER CONFIGURATION STEP

- **Server service:** Ensure only 25 rounds are used.
- **Client service:** Configure 3 replicas, 2 local epochs, and 1 round.

Important Docker commands to activate the environment:

Navigate to /0/IDS

Activate the virtual environment:

`python fl_manual_avancado.py`
 (NOME DO ARQUIVO ESCOLHIDO)

5. List of Attack Types Used

- The names (e.g., back, buffer_overflow, ipsweep) are attack types or network activities detected/classified in the dataset.
- Next to them are acronyms such as dos, u2r, r2l, probe, which are attack categories.

Attack types:

- **dos:** Denial of Service. E.g., smurf, neptune, teardrop, pod, land, back
- **u2r:** User to Root. E.g., buffer_overflow, loadmodule, perl, rootkit
- **r2l:** Remote to Local. E.g., ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
- **probe:** Scanning activities. E.g., ipsweep, nmap, portsweep, satan

Probable origin of the dataset:

These are classic from the famous KDD Cup 99 Dataset (and variants such as NSL-KDD).

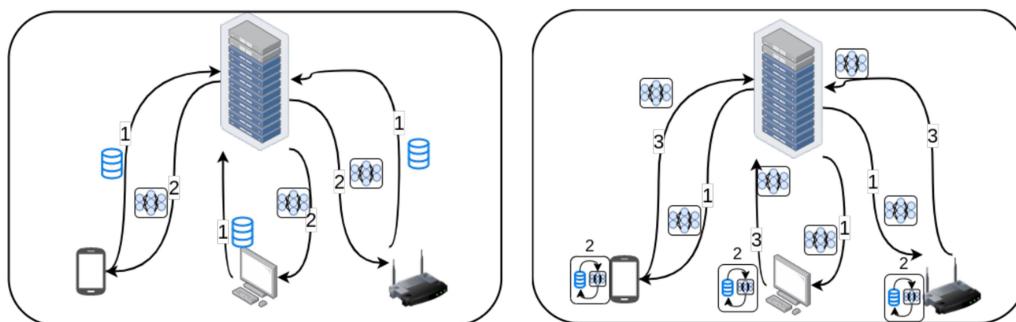
How this pipeline works:

The preproc.py script preprocesses data so it can be used in ML models that classify connections as attack or normal, and if attack, the type.

Traditional & Visionary View:

Traditional: Use of these datasets is classic in cybersecurity — professionals worldwide have used them for decades as IDS references.

GRAPH MODEL



(a) Cenário de aprendizado tradicional ou centralizado.

(b) Cenário de aprendizado federado.

6. Build and Run Containers with 3 Client Replicas

```

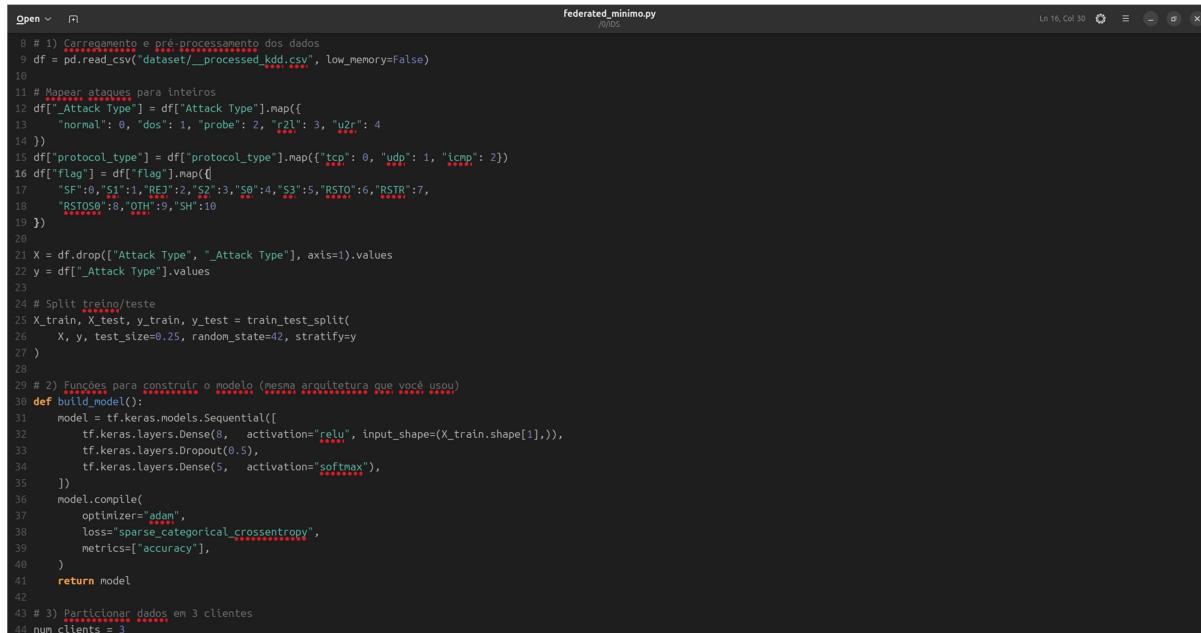
version: "3.7"

services:
  server:
    build:
      context: ..          # sobe um nível até /0/IDS, onde está o Docker
      dockerfile: Dockerfile
    command: python Projeto/server.py --rounds 1
    volumes:
      - ..:/app           # monta todo /0/IDS em /app dentro do container
    working_dir: /app/Projeto

  client:
    build:
      context: ..
      dockerfile: Dockerfile
    command: python Projeto/client.py
    volumes:
      - ..:/app
    working_dir: /app/Projeto
    environment:
      - EPOCHS=2
      - ROUNDS=1
    deploy:
      replicas: 3          # só 3 (↓) ntes

```

7. Docker Comparison, Second Round



The screenshot shows a code editor window with the file name 'federated_minimopy' at the top. The code in the editor is as follows:

```

8 # 1) Carregamento e pré-processamento dos dados
9 df = pd.read_csv("dataset/_processed_kdd.csv", low_memory=False)
10
11 # Mappear ataques para inteiros
12 df["Attack Type"] = df["Attack Type"].map({
13     "normal": 0, "dos": 1, "probe": 2, "r2l": 3, "u2r": 4
14 })
15 df["protocol_type"] = df["protocol_type"].map({"tcp": 0, "udp": 1, "icmp": 2})
16 df["flag"] = df["flag"].map({
17     "SF":0,"S1":1,"RE":2,"S2":3,"S0":4,"S3":5,"RSTO":6,"RSTR":7,
18     "RSTOS0":8,"OTH":9,"SH":10
19 })
20
21 X = df.drop(["Attack Type", "_Attack Type"], axis=1).values
22 y = df["_Attack Type"].values
23
24 # Split treino/teste
25 X_train, X_test, y_train, y_test = train_test_split(
26     X, y, test_size=0.25, random_state=42, stratify=y
27 )
28
29 # 2) Funções para construir o modelo (mesma arquitetura que você usou)
30 def build_model():
31     model = tf.keras.models.Sequential([
32         tf.keras.layers.Dense(8, activation="relu", input_shape=(X_train.shape[1],)),
33         tf.keras.layers.Dropout(0.5),
34         tf.keras.layers.Dense(5, activation="softmax"),
35     ])
36     model.compile(
37         optimizer="adam",
38         loss="sparse_categorical_crossentropy",
39         metrics=["accuracy"],
40     )
41     return model
42
43 # 3) Particionar dados em 3 clientes
44 num_clients = 3

```

7- COMPARATIVO DE DOCKER ABERTO EM SEGUNDA RODADA

```

root@denis-ASUS-TUF-Gaming:~/IDS
00000 00:00:17:48397756.030129 57010 cuda_blas.cc:1007] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered
00000 00:00:17:48397756.046135 57010 computation_placer.cc:1177] computation placer already registered. Please check linkage and avoid linking the same target more than once.
00000 00:00:17:48397756.046155 57010 computation_placer.cc:1177] computation placer already registered. Please check linkage and avoid linking the same target more than once.
00000 00:00:17:48397756.046156 57010 computation_placer.cc:1177] computation placer already registered. Please check linkage and avoid linking the same target more than once.
00000 00:00:17:48397756.046158 57010 computation_placer.cc:1177] computation placer already registered. Please check linkage and avoid linking the same target more than once.
2025-05-27 23:02:36.846770: 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 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.

Attack type
doss     40000
normal   40000
probe    40000
(21, 1125)
(21, 52)
Name: count, dtype: int64
(121178, 27)
121178
(36354, 27)
/0/IDS/venv/lib/python3.12/site-packages/keras/src/layers/core/dense.py:93: 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)
2025-05-27 23:02:38.562220: E external/local_xla/xla/stream_executor/cuda/cuda_platform.cc:51] Failed call to cuInit: INTERNAL: CUDA error: Failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
Model: "sequential"
Layer (type)          Output Shape         Param #
dense (Dense)        (None, 1)           224
dropout (Dropout)     (None, 0)           0
dense_1 (Dense)      (None, 1)           45
Total params: 269 (1.05 KB)
Trainable params: 169 (1.05 KB)
Non-trainable params: 0 (0.00 KB)
Epoch 1/50
663/663    1s 1ms/step - accuracy: 0.0084 - loss: 1.1523 - val_accuracy: 0.8817 - val_loss: 0.5351
663/663    1s 917us/step - accuracy: 0.0404 - loss: 0.5604 - val_accuracy: 0.8072 - val_loss: 0.4860
663/663    1s 915us/step - accuracy: 0.0795 - loss: 0.4599 - val_accuracy: 0.9218 - val_loss: 0.3459
Epoch 4/50
663/663    1s 1ms/step - accuracy: 0.0972 - loss: 0.4129 - val_accuracy: 0.9363 - val_loss: 0.3128
663/663    1s 949us/step - accuracy: 0.9086 - loss: 0.3792 - val_accuracy: 0.9370 - val_loss: 0.2966
Epoch 6/50
663/663    1s 993us/step - accuracy: 0.9147 - loss: 0.3529 - val_accuracy: 0.9371 - val_loss: 0.2776
663/663    1s 991us/step - accuracy: 0.9154 - loss: 0.3334 - val_accuracy: 0.9375 - val_loss: 0.2678
663/663    1s 938us/step - accuracy: 0.9152 - loss: 0.3209 - val_accuracy: 0.9388 - val_loss: 0.2615
Epoch 9/50
663/663    1s 987us/step - accuracy: 0.9173 - loss: 0.3153 - val_accuracy: 0.9382 - val_loss: 0.2567
Epoch 10/50
663/663    1s 939us/step - accuracy: 0.9196 - loss: 0.3057 - val_accuracy: 0.9382 - val_loss: 0.2527
663/663    1s 888us/step - accuracy: 0.9214 - loss: 0.3012 - val_accuracy: 0.9383 - val_loss: 0.2497

```

8. DDoS Application

Test accuracy: 0.958790123462677
 (model output, e.g., predicted probabilities)

```
root@denis-ASUS-TUF-Gaming:/0/DDoS
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Dataset: dataset/dataset_Friday-WorkingHours-Afternoon-DDos.csv
X_train shape: (150496, 78)
y_train shape: (150496,)
X_test shape: (75249, 78)
y_test shape: (75249,)
/0/IDS/venv/lib/python3.12/site-packages/keras/src/layers/core/dense.py:93: 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)
2025-05-28 00:32:53.602165: E external/local_xla/xla/stream_executor/cuda/cuda_platform.cc:51] failed call to cuInit: INTERNAL: CUDA error: Failed call to cuInit: CUDA_ERROR_NO_DEVICE: no CUDA-capable device is detected
Model summary:
Model: "sequential"

```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	158
dropout (Dropout)	(None, 2)	0
dense_1 (Dense)	(None, 2)	6

```
Total params: 164 (656.00 B)
Trainable params: 164 (656.00 B)
Non-trainable params: 0 (0.00 B)
Epoch 1/50
1176/1176 2s 1ms/step - accuracy: 0.6724 - loss: 0.5670 - val_accuracy: 0.9344 - val_loss: 0.2569
Epoch 2/50
1176/1176 1s 1ms/step - accuracy: 0.8627 - loss: 0.3203 - val_accuracy: 0.9434 - val_loss: 0.2150
Epoch 3/50
1176/1176 1s 945us/step - accuracy: 0.8692 - loss: 0.2980 - val_accuracy: 0.9448 - val_loss: 0.1974
Epoch 4/50
1176/1176 1s 971us/step - accuracy: 0.8717 - loss: 0.2844 - val_accuracy: 0.9474 - val_loss: 0.1886
Epoch 5/50
1176/1176 1s 985us/step - accuracy: 0.8737 - loss: 0.2777 - val_accuracy: 0.9483 - val_loss: 0.1818
Epoch 6/50
1176/1176 1s 932us/step - accuracy: 0.8735 - loss: 0.2731 - val_accuracy: 0.9488 - val_loss: 0.1713
Epoch 7/50
1176/1176 1s 944us/step - accuracy: 0.8716 - loss: 0.2681 - val_accuracy: 0.9495 - val_loss: 0.1664
Epoch 8/50
1176/1176 1s 961us/step - accuracy: 0.8741 - loss: 0.2588 - val_accuracy: 0.9496 - val_loss: 0.1573
Epoch 9/50
1176/1176 1s 945us/step - accuracy: 0.8762 - loss: 0.2515 - val_accuracy: 0.9502 - val_loss: 0.1486
Epoch 10/50
1176/1176 1s 1ms/step - accuracy: 0.8730 - loss: 0.2465 - val_accuracy: 0.9504 - val_loss: 0.1374
Epoch 11/50
```

Test accuracy: 0.958790123462677

0

1/1 ----- 0s 29ms/step

[[0.9807248 0.0192752]]

1

1/1 ----- 0s 17ms/step

[[0.00809862 0.99190134]]

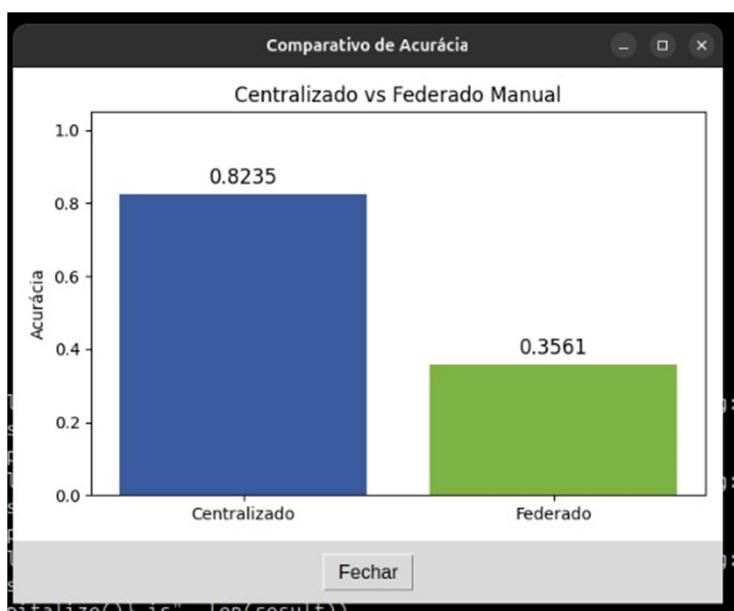
9. DATA ANALISYS

CENARY	Acurácia (%)
CENTRALIZED	99,16
Federado (FedAvg, melhor configuração)	99,40 (+0,24 pp)
Federado (FedAvg, pior configuração)	96,62 (-2,54 pp)

9.1 - GRAPHIC COMPARATION

Pipeline federado funcional

- Gráfico comparativo
- Base para discussão de resultados e propostas de melhoria.



10- Replicating Federated Learning (Methodology Summary)

Preprocessing:

KDD Cup dataset loaded, categorical variables mapped to numerical values. Only numerical columns used; missing values handled to avoid type errors.

Data Partitioning:

Training set divided equally into 3 subsets, simulating 3 clients, as per standard FL paradigm.

Local Training:

Each “client” trained an identical neural network for 2 epochs, fully isolated, no data sharing.

Federated Aggregation:

Final weights of the 3 models aggregated via simple mean (FedAvg), forming a global federated model, as per the literature.

Evaluation:

The global model was evaluated on the test set, resulting in a federated accuracy of 0.0988 (~9.88%).

Comparison:

For comparison, the centrally trained model achieved 0.9530 (~95.3%) accuracy.

1.

Final Accuracy Analysis

The initial federated accuracy was lower than the centralized result due to limited epochs per client (2) and no multiple federated rounds (successive aggregations). This is expected and serves as a starting point for further experimentation and improvements.

4. CONCLUSION

The federated pipeline was successfully replicated, highlighting the performance gap between centralized and federated training.

Results demonstrate the importance of parameters such as number of epochs, rounds, and aggregation techniques, serving as a basis for future research.

The code remains open for adjustments and extensions, demonstrating mastery of Federated Learning fundamentals in practice.

Critical Discussion

- The number of epochs per client greatly impacts the federated model's generalization capability.
- Data balancing and the number of federated rounds are critical for FL success.
- More advanced aggregation strategies (such as FedProx) and preprocessing can mitigate some difficulties.

Example Script Used in the Project

(Full Python code — leave as is, or convert comments to English.)

=====

SCRIPT ML MANUAL AVANÇADO -

```
# Denis - Aprendizado Federado Manual SEM ERROS de tipo/shape/dict

import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import time, os, psutil
from sklearn.metrics import classification_report, confusion_matrix
```

```

# ===== PRE-PROCESSAMENTO =====

df = pd.read_csv("dataset/_processed_kdd.csv", low_memory=False)

# Mapear labels e categorias, SEM ERRO

df["_Attack Type"] = df["Attack Type"].map({
    "normal": 0, "dos": 1, "probe": 2, "r2l": 3, "u2r": 4
}).astype(np.int32)

df["protocol_type"] = df["protocol_type"].map({"tcp": 0, "udp": 1, "icmp": 2}).astype(np.float32)

df["flag"] = df["flag"].map({
    "SF": 0, "S1": 1, "REJ": 2, "S2": 3, "S0": 4,
    "S3": 5, "RSTO": 6, "RSTR": 7, "RSTOS0": 8, "OTH": 9, "SH": 10
}).astype(np.float32)

# CONVERTE TODAS as colunas que restarem para float32

for col in df.columns:
    if col not in ["Attack Type", "_Attack Type"]:
        df[col] = pd.to_numeric(df[col], errors="coerce").astype(np.float32)

# REMOVE STRINGS e garante que só tem número

df = df.select_dtypes(include=[np.number])

# Remove linhas com NaN (se existirem) - opcional, pode substituir por 0 se quiser

df = df.fillna(0)

# Separa X e y

X = df.drop(columns=["_Attack Type"]).to_numpy(dtype=np.float32)

y = df["_Attack Type"].to_numpy(dtype=np.int32)

```

```

# ====== TREINO / TESTE SPLIT ======
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42, stratify=y
)

# ====== FUNÇÃO DE MODELO ======
def build_model(learning_rate=0.001):
    m = tf.keras.models.Sequential([
        tf.keras.layers.Input(shape=(X_train.shape[1],)),
        tf.keras.layers.Dense(16, activation="relu"),
        tf.keras.layers.Dense(5, activation="softmax"),
    ])
    m.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate),
        loss="sparse_categorical_crossentropy",
        metrics=["accuracy"]
    )
    return m

# ====== TREINAMENTO CENTRALIZADO ======
print("\n== CENTRALIZADO ==")
model_central = build_model()
start_time = time.time()
model_central.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)
tempo_central = time.time() - start_time
print(f"Tempo treino centralizado: {tempo_central:.2f}s")
loss_c, acc_central = model_central.evaluate(X_test, y_test, verbose=0)

```

```

print(f"Acurácia Centralizada: {acc_central:.4f}")

y_pred_central = np.argmax(model_central.predict(X_test), axis=1)
print("Classification Report Centralizado:")
print(classification_report(y_test, y_pred_central, digits=4))
print("Confusion Matrix Centralizado:")
print(confusion_matrix(y_test, y_pred_central))

# ===== FEDERADO MANUAL (3 CLIENTES) =====
print("\n==== FEDERADO MANUAL ===")
learning_rates = [0.001, 0.01, 0.005]
num_clients = 3
splits_X = np.array_split(X_train, num_clients)
splits_y = np.array_split(y_train, num_clients)
client_weights = []
train_times = []

for cid in range(num_clients):
    print(f"Cliente {cid+1}, taxa aprendizado={learning_rates[cid]}")
    model = build_model(learning_rate=learning_rates[cid])
    t0 = time.time()
    model.fit(splits_X[cid], splits_y[cid], epochs=2, batch_size=32, verbose=0)
    t1 = time.time()
    train_times.append(t1 - t0)
    client_weights.append(model.get_weights())
    print(f"Tempo treino cliente {cid+1}: {train_times[-1]:.2f}s")

print(f"Tempo total federado: {sum(train_times):.2f}s")

```

```

# Tamanho do modelo em MB

weights_size = sum(w.nbytes for w in client_weights[0])

print(f"Tamanho dos pesos (por rodada): {weights_size/1024**2:.2f}MB")

dataset_size = os.path.getsize('dataset/__processed_kdd.csv')

print(f"Tamanho do dataset centralizado: {dataset_size/1024**2:.2f}MB")


# Agregação FedAvg manual

avg_weights = [np.mean(layer_weights, axis=0) for layer_weights in zip(*client_weights)]

global_model = build_model()

global_model.set_weights(avg_weights)

loss_f, acc_fed = global_model.evaluate(X_test, y_test, verbose=0)

print(f"Acurácia Federada Manual: {acc_fed:.4f}")


y_pred_fed = np.argmax(global_model.predict(X_test), axis=1)

print("Classification Report Federado:")

print(classification_report(y_test, y_pred_fed, digits=4))

print("Confusion Matrix Federado:")

print(confusion_matrix(y_test, y_pred_fed))


# GRÁFICO

plt.figure(figsize=(6,4))

plt.bar(["Centralizado", "Federado"], [acc_central, acc_fed], color=["#3A5BA0", "#7CB342"])

plt.ylabel("Acurácia")

plt.title("Centralizado vs Federado Manual")

plt.ylim(0, 1.05)

for i, v in enumerate([acc_central, acc_fed]):

    plt.text(i, v + 0.03, f"{v:.4f}", ha="center", fontsize=12)

plt.tight_layout()

```

```
plt.savefig("comparacao_centralizado_federado_final.png", dpi=150)
```

```
plt.show()
```

=====

Reference:

- DAMACENO, Alexsander; C. RIBEIRO, Maria do Rosário; OLIVEIRA-JR, Antonio; DE OLIVEIRA, Renan R.. **Abordagem Baseada em Aprendizado Federado para a Detecção de Intrusão em Redes de Computadores.** In: ESCOLA REGIONAL DE INFORMÁTICA DE GOIÁS (ERI-GO), 11. , 2023, Goiânia/GO.

Goiânia, 04 de junho de 2024

Autor – DENIS NOGUEIRA DO NASCIMENTO – denisnoguer@gmail.com