Enhanced Dynamic Malware Detection from API Call Sequences Using Multi-Architecture Deep Learning and Explainable AI

Configaration Manual

Rigved Devendra Londhe

23267828

This is compatible with Windows as well as Mac

Version Information:

Software versions

Category	Package / Tool	Version	Notes
Language / Runtime	**Python**	**3.10.x**	Use 3.10 for widest TF/PyTorch compatibility.
Notebook format	**Jupyter nbformat**	**4.0**	Your .ipynb is nbformat 4.0.
Deep Learning (TF)	**TensorFlow (incl. Keras)**	**2.15.x**	Stable with this notebook; newer 2.16–2.17 also okay on Colab.
Deep Learning (Torch)	**PyTorch**	**2.3.1**	Works smoothly with Transformers 4.44.x.
NLP stack	**transformers**	**4.44.2**	Used for DistilBERT fine-tuning.
NLP stack	**tokenizers**	**0.19.1**	Matches transformers 4.44.x.
Hugging Face	**huggingface-hub**	**0.24.5**	Pulled by transformers.
HF runtime	**accelerate**	**0.33.0**	Transitive; not used directly.
HPO	**keras-tuner**	**1.4.7**	For Hyperband search of CNN/MLP.
ML toolkit	**scikit-learn**	**1.4.2**	Metrics, vectorizer, train/test split.
Data	**pandas**	**2.2.2**	Data loading/cleaning.
Math	**numpy**	**1.26.4**	Arrays & seeds.
Plots	**matplotlib**	**3.8.4**	Curves and confusion matrices.
(Optional) XAI	**shap**	**0.45.x**	Only needed if you enable SHAP plots.

Category	Package / Tool	Version	Notes
Environment (typical)	**Google Colab GPU**	varies	Colab-managed CUDA/cuDNN; no pin required.

Step by Step Guide to Run this on a different device.

1. Here's the whole thing in one clear run-through: open "Malware_detection_Multimodel post Hyperparameter Tuning.ipynb" in Google Colab (fastest route), switch on a GPU (Runtime → Change runtime type → Hardware accelerator → GPU → Save), and keep your CSV named exactly "Malware 2025 Dataset.csv".
2. When you run from the top, the notebook loads that CSV (it first tries "/content/Malware 2025 Dataset.csv" and, if it can't see it, Colab will pop a file picker—select your CSV).
3. The dataset must have a binary column called "labels" and fifty ordered API-call columns named "0" through "49"; your notebook fills any missing tokens with "End", builds a vocabulary from the 50 columns, converts each sample into a fixed-length sequence of 50 token IDs, and makes a stratified train/test split.
4. Then it trains Keras models (MLP, 1D-CNN, BiLSTM, and a CNN+LSTM hybrid with early stopping), fine-tunes a DistilBERT classifier on text created by joining the 50 API tokens with spaces (use the default batch sizes; if you hit out-of-memory, halve them), uses Keras-Tuner (Hyperband) to search better hyperparameters for the Keras models, and finally trains three improved PyTorch models (mean-pooled MLP, BiLSTM, multi-kernel 1D-CNN) with BCEWithLogitsLoss and an automatically calculated class weight.
5. Every model prints Accuracy, Precision, Recall, F1, and ROC-AUC; the improved PyTorch section also shows ROC/PR curves and confusion matrices.
6. When that improved section finishes, it automatically picks the best of its three models by F1 and saves two files under "artifacts_impr/": "<BEST_MODEL_NAME>state.pt" (weights) and "metadata.json" (helper info).
7. To download those, open the Files panel on the left in Colab and right-click to download, or run the small "download best model" helper provided below.
8. If you run locally instead of Colab, create a fresh Python 3.10 environment, install the packages listed below, open the notebook in Jupyter/VS Code, and change the CSV path from "/content/Malware 2025 Dataset.csv" to "./Malware 2025 Dataset.csv".
9. If your dataset's 50 columns aren't literally named "0"..."49", edit the column list where the notebook selects them (replace "call_cols = [str(i) for i in range(50)]" with your own pattern such as "call_cols = [f'API{i}' for i in range(50)]").
10. If you see "ModuleNotFoundError" for shap, keras-tuner, or transformers, (re)install with the one-liner below.
11. For a quick sanity check before training, use the pre-flight snippet below to verify the file is found, the "labels" column exists, all 50 sequence columns are present, the number of missing tokens is reported, and an approximate vocabulary size is printed.

For steady results across runs, you can set seeds (snippet below).

That’s it—run top-to-bottom, watch the printed metrics, and you’ll end up with metrics for all models plus a saved best PyTorch model you can download and deploy.

Install/upgrade the few extra libraries (Colab already has TF, Torch, sklearn, pandas, numpy, matplotlib):
!pip -q install transformers keras-tuner shap

Pre-flight dataset check (path detection, schema assertions, quick stats):
CSV_NAME = "Malware 2025 Dataset.csv"
import os, pandas as pd
cand = [f"/content/{CSV_NAME}", f"./{CSV_NAME}"]
src = next((p for p in cand if os.path.exists(p)), None)
if src is None:
try:
from google.colab import files
up = files.upload()
src = next(iter(up))
except Exception as e:
raise SystemExit("Could not find or upload the dataset. Place the CSV next to the notebook and rerun.")
df = pd.read_csv(src)
print("Loaded:", src, "Shape:", df.shape)
assert "labels" in df.columns, "'labels' column missing."
seq_cols = [str(i) for i in range(50)]
missing = [c for c in seq_cols if c not in df.columns]
assert not missing, f"Missing sequence columns: {missing[:5]}..."
n_nans = int(df[seq_cols].isna().sum().sum())
vocab = set()
for c in seq_cols:
vocab.update(df[c].dropna().astype(str).unique())
print("Missing tokens across sequence columns:", n_nans)
print("Approx vocabulary size (excl. PAD/UNK/End):", len(vocab))
print("Pre-flight checks passed.")

Optional seeds for steadier results (use before training):
import random, numpy as np
random.seed(42); np.random.seed(42)
try:
import torch; torch.manual_seed(42)
except Exception: pass
try:
import tensorflow as tf; tf.random.set_seed(42)
except Exception: pass

For strict PyTorch determinism (optional, slower):

import torch; torch.use_deterministic_algorithms(True)

Download the best improved-PyTorch model and metadata (reads metadata.json to find the right filename):

```
import os, json
from pathlib import Path
try:
from google.colab import files
except Exception:
files = None
meta_path = Path("artifacts_impr/metadata.json")
if meta_path.exists():
meta = json.loads(meta_path.read_text())
best = meta.get("best_model", None)
if best is None:
# Fall back to any _state.pt
cands = sorted(Path("artifacts_impr").glob("_state.pt"))
if not cands:
raise SystemExit("No saved model found in artifacts_impr/. Run training first.")
pt_path = str(cands[0])
else:
pt_path = f"artifacts_impr/{best}_state.pt"
else:
cands = sorted(Path("artifacts_impr").glob("*_state.pt"))
if not cands:
raise SystemExit("No saved model or metadata found in artifacts_impr/. Run training first.")
pt_path = str(cands[0])
print("Prepared for download:", pt_path)
print("Also downloading metadata if present.")
if files:
if os.path.exists(pt_path):
files.download(pt_path)
if meta_path.exists():
files.download(str(meta_path))
else:
print("Not running in Colab; files are at:", pt_path, "and", str(meta_path) if meta_path.exists()
else "(no metadata.json)")
```

Local (non-Colab) one-time environment setup (run in your terminal/venv):
python -m venv .venv

Windows: .venv\Scripts\activate

Mac/Linux:

source .venv/bin/activate

pip install tensorflow torch transformers keras-tuner shap scikit-learn pandas numpy matplotlib

Adjustment if your 50 sequence columns have different names (edit this line in the notebook where call columns are selected):
call_cols = [f"API_{i}" for i in range(50)] # replace with your prefix if needed

Reference list

Google (2024). *Colaboratory – Google*. [online] research.google.com. Available at: https://research.google.com/colaboratory/faq.html [Accessed 11 Aug. 2025].

huggingface.co. (n.d.). *Transformers*. [online] Available at: https://huggingface.co/docs/transformers/en/index [Accessed 11 Aug. 2025].

NumPy (2022). *NumPy Documentation*. [online] numpy.org. Available at: https://numpy.org/doc/ [Accessed 11 Aug. 2025].

Pandas (2018). *Python Data Analysis Library*. [online] Pydata.org. Available at: https://pandas.pydata.org/ [Accessed 11 Aug. 2025].

Team, G. (2025). *Gradio Documentation*. [online] www.gradio.app. Available at: https://www.gradio.app/docs [Accessed 11 Aug. 2025].

www.pytorch.org. (n.d.). *PyTorch*. [online] Available at: https://pytorch.org/get-started/locally/ [Accessed 11 Aug. 2025].