LVForge: PE Malware Detection from Strong Baselines to Deep Metric Learning
A Practical and Reproducible Study

Ly Ngoc Vu
Industrial University of Ho Chi Minh City
Ho Chi Minh City, Vietnam
dezzhuge@gmail.com

***Abstract*—This paper explains, in a practical step-by-step manner, what was built in LVForge for Windows PE malware detection and why each component was added. The previous stage of this project compared classical machine learning and text-based deep learning baselines on a large PE dataset. In this stage, the pipeline is extended with deep metric learning on top of a shared Transformer backbone (LVModel). Five variants are evaluated under the same data and training protocol: baseline, ArcFace, Contrastive, Triplet, and Multi-Similarity. Results are reported with multi-seed statistics and deployment-oriented metrics, including TPR at low FPR. The key outcome is that Multi-Similarity provides the most reliable operating profile, while ArcFace is unstable at strict low-FPR thresholds. The paper focuses on clarity and reproducibility so that first-time readers can directly understand what was done, how it was implemented, and what should be used in practice.**

***Index Terms*—malware detection, PE files, Transformer, deep metric learning, low-FPR evaluation, reproducibility**

I. Introduction

Windows PE malware detection is important in real systems where even a small false-positive rate can create major operational cost. A model with high overall accuracy may still fail under strict deployment thresholds. Therefore, this work focuses on both global metrics (Accuracy/F1/AUC) and operating-point metrics (TPR@FPR).

This paper is written as a clear "what was built" report for first-time readers. It consolidates knowledge from the previous project stage and the current implementation:

- **Previous stage:** strong baselines using Logistic Regression, Random Forest, SVC, XGBoost, and text-based deep models (LSTM, BiLSTM, DistilBERT) [1].
- **Current stage:** a unified Flax/JAX Transformer (LVModel) plus deep metric learning objectives.

Main contributions of this paper:

1) A clear transition from baseline classification to metric-learning variants in one reproducible pipeline.
2) A detailed architectural breakdown of LVModel from tensor flow to training heads.
3) A practical interpretation of results under low-FPR deployment constraints.

II. What Was Done: Old vs New

A. Previous Project Stage

The previous paper established a strong foundation on PE malware detection using:

- classical ML baselines (Logistic Regression, Random Forest, SVC, XGBoost),
- ensemble methods,
- text-based deep models (LSTM, BiLSTM, DistilBERT),
- and a large PE dataset setting (˜34k samples) [1], [6], [7].

B. Current Project Stage

In this stage, the core addition is deep metric learning on top of the same backbone and data pipeline. Instead of changing everything at once, only the objective/head is changed while keeping the encoder trunk shared. This makes comparisons fair and easier to interpret.

III. Methodology

A. Task and Data Setting

The binary task is benign vs malware classification from PE-derived text-like features. The current training setting uses an imbalanced ratio near 1:19 (benign:malware), which is exactly why threshold-sensitive evaluation is required.

B. LVModel Architecture (Core Encoder)

LVModel is a Transformer encoder classifier implemented in Flax/JAX.

Let input token IDs be $\mathbf{X} \in \mathbb{N}^{B\times T}$.

(1) Input representation

$$\mathbf{H}_0 = E_{tok}(\mathbf{X}) + E_{pos}(1{:}T). \tag{1}$$

This combines token embedding and positional embedding.

(2) Multi-Head Self-Attention block A single dense layer computes concatenated QKV:

$$\mathrm{QKV} = W_{qkv}\mathbf{H} \in \mathbb{R}^{B\times T\times 3d}. \tag{2}$$

Then Q, K, V are reshaped per head and attention is computed by:

$$\mathbf{A} = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right). \tag{3}$$

The attended output is projected back to model dimension d.

(3) Transformer layer design (pre-norm + residual) Each layer uses pre-normalization and two residual paths:

$$\tilde{\mathbf{H}} = \mathrm{LN}(\mathbf{H}_{l-1}), \tag{4}$$

$$\mathbf{H}' = \mathbf{H}_{l-1} + \mathrm{MHA}(\tilde{\mathbf{H}}), \tag{5}$$

$$\mathbf{H}_l = \mathbf{H}' + \mathrm{FFN}(\mathrm{LN}(\mathbf{H}')). \tag{6}$$

The FFN is two linear layers with GELU and dropout.

(4) Classification path After stacking encoder layers, sequence features are mean-pooled:

$$\mathbf{z} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{H}_{L,t}. \tag{7}$$

Then LVModel applies dense+tanh pooler, dropout, LayerNorm, and final classifier logits.

Configuration used in experiments: $d_{model} = 256$, heads $= 8$, $d_{ff} = 512$, layers $= 2$, max length $= 380$.

C. Metric-Learning Extensions

All variants share the same encoder trunk. For metric-learning variants, an additional projection to $d_{emb} = 256$ is used, followed by LayerNorm and ℓ_2 normalization of embeddings. Variant-specific heads/objectives are:

- **ArcFace:** angular margin classification [2],
- **Contrastive:** pair similarity/distance regularization [5],
- **Triplet:** relative-distance margin optimization [4],
- **Multi-Similarity:** weighted hard pair optimization [3].

D. End-to-End Pipeline

Figure 1 summarizes the full workflow used in practice.

IV. Results

A. Reference Baselines from Previous Stage

TABLE I
ML baselines from prior paper [1]

Model	Precision	Recall	F1	Accuracy
Logistic Regression	0.990086	0.990232	0.990111	0.990112
Random Forest	0.990000	0.990382	0.990402	0.990403
SVC	0.990000	0.988060	0.988075	0.988076
XGBoost	0.990000	0.991542	0.991566	0.991566

TABLE II
Deep learning baselines from prior paper [1]

Model	Precision	Recall	F1	Accuracy
DistilBERT	0.9864895	0.9865220	0.9864705	0.986471
LSTM	0.9674135	0.9674490	0.9674130	0.967413
BiLSTM	0.9507005	0.9499545	0.9500705	0.950102

TABLE III
Aggregated 5-seed metrics (mean)

Variant	Accuracy	F1	ROC-AUC	PR-AUC	TPR@FPR=1e-2
baseline	0.9924	0.9960	0.9983	0.9999	0.9754
arcface	0.7979	0.7934	0.9704	0.9984	0.0000
contrastive	0.9931	0.9964	0.9971	0.9997	0.9533
triplet	0.9932	0.9964	0.9969	0.9998	0.9351
multi_similarity	**0.9946**	**0.9972**	0.9978	0.9999	**0.9851**

B. Current Stage: DML Comparison

What these numbers mean in practice:

- Multi-Similarity is the strongest overall choice for deployment.
- Baseline is still robust and can serve as fallback/reference.
- ArcFace is unstable for strict low-FPR operation in this setting.
- Runtime across variants remains in a comparable practical range (about 94–133s per run variant in current setup).

V. Discussion and Deployment Guidance

This work shows that adding metric learning is useful when objective choice is careful. The shared trunk + variant-head design made it easy to isolate what actually improved performance. For production-oriented usage:

1) prioritize operating-point metrics (TPR@FPR), not only accuracy,
2) use Multi-Similarity as primary candidate,
3) keep baseline as a stable reference,
4) recalibrate thresholds when data distribution shifts.

A. Threats to Validity

- single-domain pipeline evaluation,
- limited seed count,
- potential calibration sensitivity under temporal drift,
- static-feature limitations for packed/obfuscated binaries.

VI. Conclusion

For first-time readers, the core message is simple: this project started from strong baselines, then added a carefully controlled metric-learning extension on the same LVModel backbone. The implementation is reproducible, the architectural changes are explicit, and results indicate that Multi-Similarity gives the best operational profile under the tested PE malware setting.

Acknowledgment

This work builds on the author's previous project stage and extends it with architecture-level and objective-level analysis.

References

[1] L. N. Vu, "Windows Malware Detection: Exploring from Machine Learning to Text-Based Deep Learning Approaches," in Proc. ATC, 2024.

[2] J. Deng *et al.*, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," CVPR, 2019.

[3] X. Wang *et al.*, "Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning," CVPR, 2019.

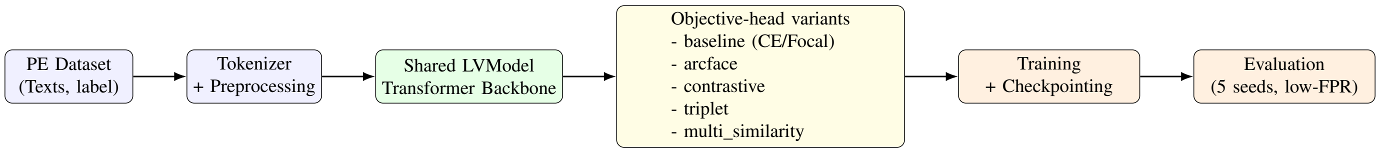

Fig. 1. Unified pipeline for baseline and DML variants.

[4] F. Schroff *et al.*, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” CVPR, 2015.

[5] R. Hadsell *et al.*, “Dimensionality Reduction by Learning an Invariant Mapping,” CVPR, 2006.

[6] A. Harang and E. M. Rudd, “SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection,” arXiv:2012.07634, 2020.

[7] BODMAS Dataset, “Benchmark for malware analysis at scale,” 2021.

[8] C. Guo *et al.*, “On Calibration of Modern Neural Networks,” ICML, 2017.

[9] S. Wang *et al.*, “A Comprehensive Survey of Out-of-Distribution Detection Methods for AI Security,” ACM Comput. Surv., 2024.