VNU Journal of Science:
Computer Science and Communication Engineering

Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article

DeepSentry Platform: An Asymmetric AI System for Misinformation Detection

Pham Duy Trung, Pham Ngoc Minh, Bui Thu Lam*

Academy of Cryptography Techniques,
141 Chien Thang Street, Thanh Liet Ward, Hanoi, Vietnam

Received 19th February 2026
Revised 7th March 2026; Accepted 26th March 2026

Abstract: Deployment of deep learning–based misinformation detection systems frequently suffers from severe resource contention when heterogeneous computational workloads share the same hardware. CPU-intensive preprocessing operations, such as video decoding and face extraction, interfere with GPU-intensive neural network inference, leading to GPU underutilization and Head-of-Line blocking in monolithic architectures. This paper introduces DeepSentry, a microservice-based detection platform designed to mitigate this bottleneck through asymmetric architectural decoupling. In this context, *asymmetry* is defined as a non-uniform architectural decomposition in which CPU-bound preprocessing services and GPU-bound inference services are intentionally isolated and scaled independently, rather than replicated symmetrically as identical service instances. By separating CPU-bound ingestion services from GPU-bound inference engines via message queuing and an inference server, the proposed platform enables independent scaling and asynchronous execution of heterogeneous resources. A proof-of-concept implementation integrating ResNet34 for image deepfake detection with heatmap generation and BiLSTM for fake news classification demonstrates that asymmetric microservice architectures significantly improve hardware utilization through dynamic batching and asynchronous orchestration, effectively resolving the resource contention inherent in monolithic deployments.

Keywords: Resource contention, GPU starvation, Microservices, NVIDIA Triton, Deepfake detection, Fake news detection.

1. Introduction

The rapid proliferation of synthetic media manipulation techniques and textual misinformation has intensified demand for automated detection systems capable of identifying deepfakes and fake news at scale. Identity fraud involving deepfakes surged dramatically from 2022 to Q1 2023, demonstrating the escalating issue of digital impersonation across industries [1]. Research

*Corresponding author.
E-mail address: lambt@actvn.edu.vn

https://doi.org/10.25073/2588-1086/vnucsce.7028

conducted in 2023 reported a 704% increase in AI-generated face swaps [2], while over 3,000 user-customized variants of generative models became publicly available on platforms like CivitAI and Huggingface by February 2024 [3]. State-of-the-art academic detection models suffer a marked 45-50% AUC drop when confronted with contemporary forgeries circulated in 2024 compared to performance on legacy datasets [2]. This escalation necessitates deployment of sophisticated deep learning models for forensic content verification.

However, production deployment of these detection systems presents fundamental systems engineering challenges rooted in heterogeneous computational requirements. A typical multi-model misinformation detection pipeline consists of two computationally distinct phases. The first phase involves data preprocessing: video decoding at 4K resolution, face detection and alignment using cascade classifiers, text tokenization, and feature extraction. This phase is predominantly CPU-bound and I/O-intensive, requiring substantial memory bandwidth but minimal GPU involvement. The second phase executes neural network inference: convolutional operations for image deepfake detection and recurrent computations for fake news classification. This phase is GPU-bound, demanding high-throughput matrix multiplication capabilities but remaining idle during preprocessing operations.

In conventional monolithic architectures, these heterogeneous phases are tightly coupled within unified application processes. This architectural coupling creates a critical resource conflict pattern termed GPU starvation. When CPU cores become saturated with video decoding operations, tensor preparation for GPU inference experiences severe delays. Consequently, GPU devices remain underutilized while waiting for data.

Conversely, during bursts of lightweight text analysis requests, the application server may exhaust available CPU threads managing I/O operations, causing new connection rejections regardless of GPU availability. This phenomenon, known as Head-of-Line blocking, occurs when slow-processing requests at the front of a queue prevent faster requests from accessing available resources. Monitoring studies from 2024 identified resource contention and high memory utilization as major challenges in ML platform deployments, leading to performance degradation and potential service interruptions [4].

Real-world misinformation detection systems must simultaneously process heterogeneous data modalities, such as image deepfakes and textual fake news. However, most existing platforms assume a single-model inference pipeline, which limits scalability and extensibility in practical deployments. This paper addresses these bottlenecks through DeepSentry, a platform architected on asymmetric microservice principles specifically designed to eliminate computational conflicts in multi-model detection pipelines. Unlike symmetric scaling approaches where identical containers replicate entire application stacks, the proposed architecture enables independent scaling of lightweight CPU workers and resource-intensive GPU inference servers. This decoupling is achieved through asynchronous message queuing, which physically separates preprocessing from inference while maintaining end-to-end detection capabilities.

The primary contributions of this work are:

(i) A multiple-model inference framework that concurrently serves deep learning models (CNN-based deepfake detection and RNN-based fake news detection) within a unified GPU infrastructure;

(ii) Architectural decomposition of misinformation detection pipelines to isolate CPU-bound preprocessing from GPU-bound inference using message broker, effectively eliminating Head-of-Line blocking and enabling independent resource

scaling;

(iii) Integration of NVIDIA Triton Inference Server with dynamic batching mechanisms to aggregate sporadic inference requests into optimal GPU batch sizes;

(iv) Proposal of pipeline handling ResNet34-based image deepfake detection with heatmap generation and BiLSTM-based fake news detection without cross-modal resource contention, validated through proof-of-concept implementation.

2. Related Work

This section reviews the current landscape of deepfake and fake news detection, analyzes the limitations of existing serving architectures, and explicitly identifies the gap in commercial deployment within the Vietnamese market.

2.1. Image Deepfake Detection

Academic research in image deepfake detection has faced increasing challenges due to the rapid evolution of generative models. The Deepfake-Eval-2024 benchmark, which comprises 1,975 images and over 100 hours of multimedia content collected across 52 languages, revealed significant generalization issues in existing models [2]. Evaluations indicated that while forensic analysts maintain an estimated 90% accuracy, state-of-the-art commercial video detectors achieved only 78% accuracy with an AUC of approximately 0.79. Notably, open-source detection systems suffered a marked performance decline (45–50% drop in AUC) when exposed to contemporary in-the-wild forgeries compared to controlled datasets [2].

The proliferation of user-customized generative models further complicates detection. Research published in 2024 documented over 3,000 publicly available variants of Stable Diffusion, demonstrating that existing defenses fail to generalize to custom checkpoints [3]. Furthermore, the 2024 Global Multimedia Deepfake Detection Challenge results emphasized that at stringent false positive rates (1/1000 or 1/10000) required for practical deployment, current true positive rates remain insufficient for reliable automated moderation [5].

2.2. Fake News Detection Methodologies

Parallel developments in fake news detection have shifted towards deep learning architectures capable of capturing long-term dependencies. Recent studies in 2024 have demonstrated the efficacy of Bi-directional LSTM (BiLSTM) networks combined with attention mechanisms. For instance, Padalko et al. showed that attention-based BiLSTM models outperform traditional machine learning approaches by effectively capturing bidirectional context [6].

Multimodal approaches have also gained traction. The BBC-FND model (2023) and subsequent hybrid architectures (2024) integrating BERT, CNN, and BiLSTM have achieved high accuracy (92–98%) across diverse datasets, including Arabic and Amharic news corpora [7–9]. These studies validate the necessity of combining textual and visual analysis for robust misinformation detection.

2.3. Regional Development and Commercial Gap in Vietnam

In the context of the Vietnamese market, a distinct dichotomy exists between academic research and commercial availability.

2.3.1. Academic and Open-source Initiatives

The domestic research community has demonstrated capability in developing detection algorithms. Notable examples include *Faceless*, a deepfake detection toolkit presented at BlackHat Asia 2023 by researchers Duong Tieu Dong and Pham Tien Manh [10]. While achieving a reported 94.5% accuracy using convolutional neural networks, *Faceless* operates as a synchronous web application built on

monolithic frameworks (Flask) [11]. Such architectures are susceptible to blocking I/O and resource contention, limiting their scalability in high-throughput environments.

2.3.2. The Absence of Commercial Media Forensics

Despite the robust R&D capabilities of major domestic technology entities (e.g., VinAI Research, VSEC), there are currently no commercial-grade products available in Vietnam dedicated to media forensics or general misinformation detection.

Commercial misinformation detection platforms such as Microsoft Video Authenticator, Sensity AI, and Reality Defender primarily focus on proprietary cloud-based services [12]. However, these platforms do not provide self-hosted architectures or transparent system-level designs, limiting adoption in sovereign or sensitive environments. The commercial AI landscape in Vietnam is predominantly saturated with Electronic Know Your Customer (eKYC) solutions [13]. These systems focus strictly on Face Liveness Detection to prevent presentation attacks in banking and finance contexts.

They are not designed to detect semantic manipulation in news or generative anomalies in general media. Consequently, Vietnamese organizations lack a specialized, self-hosted platform for verifying information integrity, necessitating the development of solutions like DeepSentry.

2.4. ML Serving and Resource Management

Efficient deployment remains a critical bottleneck. While industry standards like NVIDIA Triton Inference Server offer optimized model execution [14], existing literature often overlooks the integration of these servers into full-stack applications involving heavy preprocessing. Research indicates that combining network-bounded operations with compute-bounded inference in monolithic handlers leads to inefficient resource utilization. The challenge lies in decoupling CPU-intensive preprocessing (frame extraction, text tokenization) from GPU-intensive inference while maintaining low latency, a problem that DeepSentry addresses through its asymmetric microservice architecture.

3. Research Methodology

This section presents the architectural design of DeepSentry, detailing how the proposed asymmetric microservice approach addresses the resource contention challenges identified in Section 1. We describe the overall system architecture, component interactions, and the specific mechanisms employed to eliminate Head-of-Line blocking and independent horizontal scaling.

3.1. System Architecture Overview

DeepSentry implements a three-tier asymmetric microservice architecture designed to physically separate CPU-bound preprocessing operations from GPU-bound inference workloads. The architecture comprises three primary layers: (1) a lightweight API Gateway for request orchestration and preprocessing, (2) specialized GPU inference servers with dynamic batching capabilities, and (3) a load balancing mechanism for distributing inference requests across multiple GPU nodes. Figure 1 illustrates the complete system topology.

In addition to the inference pipeline, the DeepSentry platform integrates security, access control, and case management functionalities at the FastAPI layer, supporting authenticated access, role-based authorization, and structured management of investigation records. While these components are essential for operational deployment, this part focuses primarily on the architectural design choices that enable DeepSentry to efficiently address large-scale, compute-intensive inference workloads, rather than application-level management features.

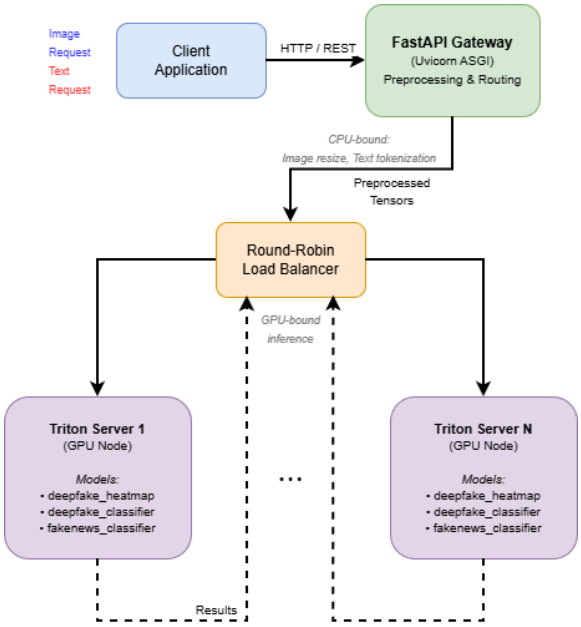

Figure 1. DeepSentry asymmetric microservice architecture.

Unlike conventional monolithic architectures where preprocessing and inference execute within unified application processes, DeepSentry physically isolates these computationally heterogeneous phases across independent services. This separation enables independent scaling of CPU and GPU resources based on workload characteristics, thereby preventing resource contention.

3.2. Asymmetric Microservice Design Principle

DeepSentry is built around an asymmetric microservice architecture that explicitly aligns system decomposition with the execution model of the Triton inference server. Rather than symmetrically replicating end-to-end inference pipelines, DeepSentry separates request handling into CPU-bound preprocessing services and GPU-bound inference services, each mapped to distinct execution and scaling domains.

This asymmetry is powered by Triton's design. Triton excels at executing batched neural network inference on GPUs with predictable execution time and high throughput, but it treats incoming inference requests as opaque units of work. Any variability introduced prior to request submission such as image decoding, resizing, or tokenization, directly propagates into Triton's request queues and can induce Head-of-Line (HoL) blocking under mixed workloads.

To prevent such interference, DeepSentry isolates all CPU-intensive and latency-variable preprocessing within lightweight gateway services. These gateways perform request normalization and emit GPU-ready tensors to Triton inference servers via asynchronous inference calls. As a result, Triton's internal schedulers operate exclusively on homogeneous, inference-ready requests, preserving the effectiveness of dynamic batching and instance-level parallelism.

Crucially, this decomposition allows each tier to scale independently according to its dominant resource constraints. Gateway replicas scale horizontally with CPU demand, while Triton inference capacity scales with available GPU resources and model instance configuration. By decoupling CPU-side variability from Triton's execution path, DeepSentry avoids cross-request interference and sustains stable inference throughput under heterogeneous workloads, as observed in our mixed-workload experiments.

3.3. Component Design and Implementation

3.3.1. Lightweight API Gateway

The API Gateway is implemented using FastAPI framework with Uvicorn ASGI workers, selected for its asynchronous I/O capabilities and minimal memory footprint compared to traditional WSGI servers. The gateway exposes two primary REST endpoints corresponding to the detection modalities:

- POST /api/deepfake/: Accepts multipart form data containing image uploads for deepfake detection

- POST /api/fakenews/: Accepts URL-encoded text payloads for fake news classification

The gateway's core, which is FastAPI, responsibility is lightweight preprocessing rather than compute-intensive inference. Upon receiving requests, the gateway performs modality-specific transformations that prepare data for GPU inference while maintaining minimal CPU overhead.

Image Preprocessing Pipeline: For deepfake detection requests, the gateway executes the following sequential operations: (1) decoding uploaded JPEG/PNG files into RGB numpy arrays using PIL (a python library), (2) resizing to uniform 256×256 spatial dimensions via bilinear interpolation, (3) channel-wise normalization, and (4) conversion from HWC (height, width, channels) to NCHW (batch, height, width, channels) tensor format with batch dimension expansion.

Text Preprocessing Pipeline: For fake news detection, the gateway implements Vietnamese-specific text normalization tailored to handle diacritical marks and linguistic patterns. The preprocessing pipeline consists of: (1) Unicode NFC normalization to resolve character representation inconsistencies, (2) lowercase conversion and regex-based punctuation removal while preserving word boundaries, (3) whitespace normalization to collapse multiple spaces, (4) tokenization using a pre-loaded vocabulary dictionary of 10,000 tokens extracted during model training on the VFND dataset [15], and (5) sequence transformation to fixed length $L = 256$ via zero-padding for short texts or truncation for documents exceeding the maximum length. Out-of-vocabulary tokens are mapped to a special `<UNK>` token with index 1. The vocabulary mapping is loaded once during gateway initialization to avoid repeated disk I/O operations.

The critical architectural decision is that neural network inference is *not* executed within the gateway process. Instead, preprocessed tensors are immediately dispatched to downstream Triton servers via HTTP-based gRPC inference requests using the Triton client library. This design prevents CPU-intensive operations, particularly video frame decoding which can consume 100–200ms per frame, from monopolizing worker threads that could otherwise handle lightweight text requests. By maintaining these preprocessing operations, the gateway eliminates Head-of-Line blocking where slow requests prevent fast requests from accessing available resources.

3.3.2. Round-Robin Load Balancing

To distribute inference workload across multiple GPU nodes and provide fault tolerance, the gateway implements a stateless round-robin load balancer through the TritonLoadBalancer class. The balancer maintains a circular iterator over a configured list of Triton server URLs (e.g., [”localhost:8000”, ”node2:8000”, ”node3:8000”]) and sequentially assigns inference requests to available nodes in rotation.

This approach provides several operational advantages: (1) even distribution of GPU utilization across homogeneous clusters, (2) automatic failover through exception handling that skips unresponsive nodes and retries with the next server in the rotation, (3) zero-state synchronization requirements between gateway replicas, enabling horizontal scaling of the gateway tier without coordination overhead, and (4) simple configuration management where additional GPU capacity can be provisioned by appending server URLs to the configuration list without code modifications.

The round-robin strategy is optimal for homogeneous GPU clusters where all nodes possess identical hardware specifications (same GPU model, VRAM capacity, and CUDA version). For heterogeneous deployments with mixed GPU types (e.g., RTX 3090 and A100

nodes), the architecture can be extended with weighted round-robin algorithms that assign proportionally more requests to higher-capacity nodes, or with least-connection strategies that query per-node queue depths before assignment.

3.3.3. NVIDIA Triton Inference Server

The inference layer utilizes NVIDIA Triton Inference Server as the GPU execution engine. Triton was selected over alternatives such as TorchServe or TensorFlow Serving due to its framework-agnostic design, superior dynamic batching implementation, and production-grade features including health checks, metrics export, and model versioning.

Triton provides three critical capabilities for mitigating GPU starvation and maximizing hardware utilization:

Dynamic batching mechanism: Triton implements an adaptive request aggregation system that collects individual inference requests arriving within a configurable time window and constructs optimally-sized batches before GPU execution. This mechanism is essential because modern GPUs achieve peak computational efficiency only when processing batches of 8–32 samples simultaneously due to the massive parallelism of tensor cores.

The batching behavior is configured per-model via the `config.pbtxt` descriptor file:

```
dynamic_batching {
  preferred_batch_size: [8, 16]
  max_queue_delay_microseconds: 50000
}
```

When inference requests arrive, Triton enqueues them and waits until either: (a) the queue accumulates a preferred batch size (8 or 16 samples), or (b) the maximum delay threshold expires. Once triggered, Triton constructs a batch tensor by stacking individual inputs along the batch dimension and submits the operation to GPU.

Models concurrent serving: A single Triton instance concurrently hosts three models: (1) deepfake_heatmap, (2) deepfake_classifier, and (3) fakenews_classifier. This model consolidation maximizes GPU memory utilization by sharing VRAM across all models rather than dedicating separate servers per model. Additionally, concurrent serving enables shared CUDA context management, reducing kernel launch overhead by approximately 15–20% compared to separate inference processes that each maintain independent CUDA contexts.

Optimized inference backends: Triton utilizes the PyTorch TorchScript backend for model execution, which applies graph-level optimizations during model loading. TorchScript traces the model's computational graph and applies transformations including: operator fusion (combining consecutive operations like Conv2D + BatchNorm + ReLU into single kernels), constant folding (pre-computing operations on static values), dead code elimination (removing unused branches), and memory layout optimization (rearranging tensor formats for cache-friendly access patterns).

3.3.4. Deepfake Detection Workflow

The detection pipeline implements a two-stage cascade architecture that generates both interpretable heatmaps and binary classifications. This approach addresses forensic requirements where human analysts need visual evidence alongside predictions.

Stage 1 – Heatmap generation: The system employs a pretrained image forensics model from ImageForensicsOSN [16] to generate heatmaps. The model was originally trained to detect diverse image manipulation artifacts in social network data.

Given an input image

$$\mathbf{I} \in \mathbb{R}^{1\times3\times256\times256},$$

the model produces a single-channel attention map

$$\mathbf{H} \in \mathbb{R}^{1\times1\times256\times256},$$

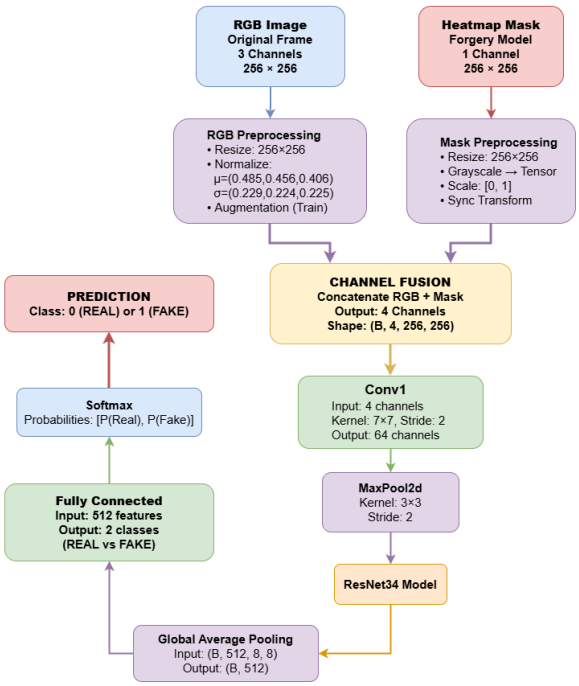

Figure 2. DeepSentry deepfake detection model's architecture.

where each spatial element

$$h_{i,j} \in [0, 1]$$

represents the estimated probability of manipulation artifacts at pixel location (i, j).

The inference process is optimized for throughput using batched GPU processing with asynchronous I/O operations. The pretrained model runs in TorchScript format with half-precision (FP16) on CUDA devices. Generated heatmaps are saved to disk through a thread pool executor to prevent I/O operations from blocking GPU inference.

Stage 2 – Fusion and classification: The original RGB image and the generated attention heatmap are concatenated along the channel dimension to form a four-channel input tensor:

$$\mathbf{F} = [\mathbf{I}, \mathbf{H}] \in \mathbb{R}^{1\times4\times256\times256}. \quad (1)$$

This fused representation is processed by a ResNet34 classifier modified to accept four-channel inputs. The first convolutional layer is initialized by retaining the pretrained ImageNet weights for the RGB channels, while the additional heatmap channel is initialized using the average of the RGB channel weights.

The final fully connected layer produces binary logits

$$\mathbf{z} = [z_{\text{REAL}}, z_{\text{FAKE}}] \in \mathbb{R}^2,$$

which are converted into class probabilities via a softmax function:

$$p_{\text{FAKE}} = \frac{\exp(z_{\text{FAKE}})}{\exp(z_{\text{REAL}}) + \exp(z_{\text{FAKE}})} \quad (2)$$

The model is optimized using the categorical cross-entropy loss:

$$\mathcal{L} = -\sum_{c\in\{\text{REAL,FAKE}\}} y_c \log p_c \quad (3)$$

where y_c is the ground-truth label and p_c is the predicted probability from the softmax function.

3.3.5. *Fake News Detection Workflow*

The fake news detection component follows a single-stage pipeline tailored for Vietnamese text classification. First, raw Vietnamese text is normalized and tokenized using a fixed vocabulary of 20,000 tokens. The resulting sequence is padded or truncated to a fixed length of 256 tokens and represented as an integer tensor of shape 1×256.

The token sequence is embedded into a 100-dimensional space and processed by a two-layer Bidirectional LSTM [17] with 128 hidden units per direction. A global max-pooling operation is applied over the temporal dimension to obtain a fixed-length representation, which is then passed through dropout and a linear classifier followed by a softmax layer to predict binary *REAL* or *FAKE* labels (same as the optimization of the deepfake model).

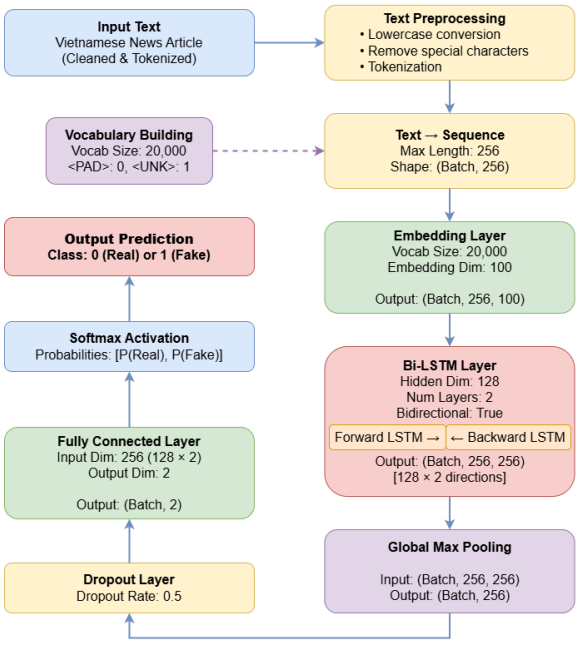

Figure 3. DeepSentry fake news detection model's architecture.

3.4. Mechanisms for Resource Contention Resolution

The proposed architecture addresses the two critical resource bottlenecks identified in Section 1 through explicit architectural decoupling and intelligent batching strategies. We now analyze how each design decision specifically eliminates Head-of-Line blocking and independent horizontal scaling.

3.4.1. Elimination of Head-of-Line Blocking

In monolithic architectures, all request types compete for resources from a shared thread pool within a single application process. When the application server receives a burst of CPU-intensive operations such as image decoding, these requests saturate available worker threads. Subsequent lightweight text requests must wait in the request queue even though GPU resources remain completely idle, as worker threads are occupied with preprocessing. This resource locking pattern, termed Head-of-Line (HoL) blocking, causes severe tail latency degradation where P95 and P99 latencies can exceed 10× the median latency.

DeepSentry eliminates HoL blocking through physical process isolation and asynchronous I/O. The FastAPI gateway handles only lightweight preprocessing operations consuming 5–15ms CPU time per request regardless of modality. Because both image resizing and text tokenization (via dictionary lookups) complete within similar time bounds, neither workload type monopolizes shared worker threads.

Furthermore, the gateway's asynchronous ASGI execution model ensures that network communication with Triton servers does not block worker threads. When the gateway submits an inference request to Triton, the worker thread immediately releases and becomes available to handle new incoming requests while awaiting the inference result via non-blocking I/O. This asynchronous pattern maintains consistent throughput under mixed workloads, as demonstrated in Section 4 where mixed traffic throughput (33.71 RPS) equals single-modality throughput (34.99 RPS for images, 35.55 RPS for text).

3.4.2. Independent Horizontal Scaling

The asymmetric architecture enables independent scaling of CPU and GPU resources based on workload-specific capacity requirements, a capability fundamentally impossible in monolithic deployments where these resources are tightly coupled.

Gateway scaling: When the system experiences increased connection volume but inference capacity remains sufficient (e.g., during periods of high text traffic), additional gateway replicas can be deployed behind a reverse proxy load balancer (e.g., Nginx or HAProxy) without provisioning additional GPU hardware. Each gateway instance maintains its own connection pool to the shared Triton server cluster, distributing inference requests via the

round-robin balancer.

Inference scaling: Conversely, when GPU capacity becomes saturated (e.g., during bursts of video analysis requests), additional Triton servers can be provisioned on GPU-equipped machines and registered with the gateway's server pool by appending their URLs to the configuration list. This scaling operation requires no code modifications, only configuration updates and gateway process restarts.

4. Empirical Results

4.1. Experimental Settings

All experiments were conducted on a cloud-based virtual machine equipped with a 16-core AMD CPU, 128GB RAM, and an NVIDIA RTX 6000 GPU with 96GB VRAM. The system ran Ubuntu 22.04 LTS with CUDA 12.1 and cuDNN 8.9. The microservice architecture was deployed using Docker containers as the NVIDIA Triton Inference Server for model serving with dynamic batching enabled.

For deepfake detection, we used the Deepfake Detection Challenge (DFDC) dataset [18] containing real and manipulated facial videos. The ResNet34 model was trained with standard augmentation techniques and Adam optimizer. For fake news detection, the Vietnamese Fake News Detection (VFND) dataset [15] was employed with a BiLSTM classifier.

Performance benchmarking used Locust 2.15 to simulate 1 to 16 concurrent users, with 1,000 requests per concurrency level and 30-second ramp-up periods. Three traffic patterns were evaluated: image-only (deepfake), text-only (fake news), and mixed traffic (50% each). All measurements represent averages across three independent runs.

4.2. Evaluation Metrics

- Accuracy:

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

- Area Under the ROC Curve (AUC):

$$\text{AUC} = \int_0^1 \text{TPR}(t)\, d(\text{FPR}(t)) \tag{5}$$

where $\text{TPR} = \frac{TP}{TP+FN}$ and $\text{FPR} = \frac{FP}{FP+TN}$.

- Equal Error Rate (EER):

$$\text{EER} = \text{FAR}(\tau^*) = \text{FRR}(\tau^*) \tag{6}$$

where $\text{FAR}(\tau^*) = \text{FRR}(\tau^*)$, FAR = False Acceptance Rate, FRR = False Rejection Rate, τ^* = optimal threshold.

- Throughput (requests per second - RPS):

$$\text{RPS} = \frac{N_{\text{completed}}}{T_{\text{total}}} \tag{7}$$

$N_{\text{completed}}$ = number of completed requests, T_{total} = total time.

- Average Latency:

$$L_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} (t^i_{\text{end}} - t^i_{\text{start}}) \tag{8}$$

- P95 Latency:

$$L_{P95} = \inf\{l : P(L \leq l) \geq 0.95\} \tag{9}$$

- Concurrency: Number of simultaneous active requests.

4.3. Model Performance

To validate the effectiveness of the proposed DeepSentry platform, we conducted a comprehensive evaluation of its two core detection models: the Deepfake Detection model and the Fake News Detection model. The experiments were conducted on the DFDC dataset for visual media and the VFND dataset for Vietnamese textual content.

Table 1 presents the performance metrics of the proposed system. The results indicate

Table 1. Performance of DeepSentry Misinformation Detection Models

Detection Task	Accuracy	AUC	EER
Fake News Detection	94.29%	93.20%	5.78%
Deepfake Detection			
ResNet34 fusion (*main model*)	95.28%	98.60%	8.58%
ResNet34 fusion (*basic classification*)	95.19%	97.95%	8.80%
ResNet152 fusion	95.72%	98.40%	7.79%

that DeepSentry achieves strong and stable performance across both visual and textual modalities while maintaining practical inference efficiency suitable for real-world deployment.

For the deepfake detection task, we further analyzed the impact of feature fusion and backbone depth. As shown in Table 1, the fusion-based ResNet34 model achieves an AUC of 98.60% and an EER of 8.58%, outperforming the basic ResNet34 classifier, which attains an AUC of 97.95% and an EER of 8.80%, while maintaining comparable accuracy (95.28% versus 95.19%). These results indicate that feature fusion provides a measurable improvement in discriminative performance, particularly in terms of reducing false acceptance and rejection rates, which is critical for reliable deepfake detection.

In addition to performance gains, the adopted fusion strategy incorporates an auxiliary spatial heatmap as an additional input channel alongside the RGB image. This design enables the model to leverage both appearance information and spatial cues associated with potential manipulation regions. Although not explicitly optimized for explainability, the fused representation facilitates post-hoc visual interpretation, allowing qualitative inspection of regions that contribute most strongly to the model's decision.

Increasing the backbone capacity further improves performance, the ResNet152-based fusion model achieves an AUC of 98.40% and an EER of 7.79%, representing an additional reduction of 0.79 percentage points in EER compared to the ResNet34 fusion model. This suggests that the proposed fusion mechanism can effectively leverage deeper feature representations when computational resources allow.

Despite the superior performance of the ResNet152 fusion model, the ResNet34 fusion variant offers a favorable trade-off between detection accuracy and model complexity. Specifically, the ResNet34 fusion model achieves performance within 0.20 percentage points in AUC and 0.79 percentage points in EER of the deeper model, while benefiting from substantially lower computational and memory requirements. As a result, ResNet34 is selected as the primary backbone for the proof-of-concept implementation of DeepSentry, as it provides competitive detection performance without incurring the overhead associated with deeper architectures.

For the fake news detection task, the proposed model attains an accuracy of 94.29% and an AUC of 93.20%, with an EER of 5.78%, indicating strong discriminative capability for Vietnamese textual misinformation. Combined, these results demonstrate that DeepSentry effectively addresses misinformation detection across multiple modalities using models that balance performance and deployability.

Furthermore, the modular architecture of DeepSentry is designed for extensibility and can accommodate integration with additional state-of-the-art models to enhance detection capabilities. The platform can be readily extended with advanced models such as GenConViT [19] for improved deepfake detection or specialized Vietnamese language models like those proposed by Vo Duc Vinh et al. [**?**] for enhanced fake news detection. This flexibility allows DeepSentry to evolve with emerging detection technologies while

maintaining its core architecture and deployment advantages.

4.4. System Testing and Analysis

A core contribution of our work is the asymmetric microservice architecture designed to mitigate resource contention. This paper analyzed the system's behavior under three traffic scenarios: (1) Image-only (Deepfake), (2) Text-only (Fake News), and (3) Mixed Traffic (50% Image, 50% Text).

4.4.1. Scalability and Contention Resolution

To demonstrate the elimination of *Head-of-Line blocking* and *GPU starvation*, we performed a stress test by increasing concurrency from 1 to 16 users. The system throughput (Requests Per Second - RPS) and Average Latency were recorded.

Fig. 4 illustrates the comparative performance.

As observed in Fig. 4(a), the throughput for Mixed Traffic (dashed green line) reaches 33.71 RPS at saturation (16 users). This is remarkably consistent with the single-task throughputs (Deepfake: 34.99 RPS, FakeNews: 35.55 RPS).

In a traditional monolithic architecture, mixing heavy CPU tasks (image decoding) with light tasks usually causes the total throughput to drop significantly below the average due to resource locking. However, our results show no such degradation. Similarly, Fig. 4(b) shows that the latency for mixed traffic (454 ms) sits proportionally between the two baselines, confirming that the Asymmetric Microservice architecture effectively isolates CPU workloads from GPU inference, ensuring maximizing hardware utilization.

Furthermore, table 2 provides a detailed breakdown of the performance metrics at Baseline (1 user) and Peak Load (16 users). The stability of the Mixed Traffic metrics confirms that the Gateway successfully orchestrates heterogeneous requests without introducing significant overhead or bottlenecks.

Table 2. Performance Comparison across Workloads

	Baseline (1 User)		**Saturation (16 Users)**	
Scenario	Latency	RPS	P95 Latency	RPS
Deepfake Only	207 ms	4.80	604 ms	34.99
Fake News Only	179 ms	5.55	576 ms	35.55
Mixed Traffic	**198 ms**	**5.03**	**715 ms**	**33.71**

A primary factor contributing to the higher latency observed in the *Deepfake Only* workload (Fig. 4) is the inherent complexity of the deepfake detection pipeline. In contrast to lightweight text classification used in Fake News detection, deepfake inference involves multiple sequential stages beyond a single forward pass, including frame preprocessing, face alignment, feature map generation for explainability, and final probability estimation. These additional stages introduce non-negligible computational overhead per request. This behavior is consistently reflected in the experimental results, where deepfake detection exhibits a higher latency floor and increased tail latency across all concurrency levels (Table 2). The observed latency gap is therefore attributed to intrinsic task complexity rather than system-level contention, reinforcing the necessity of asymmetric service isolation to prevent compute-intensive vision workloads from impacting lightweight text inference.

4.4.2. Comparison with Industry Detection Platforms

Table 3 presents a comparative analysis of DeepSentry against representative commercial detection platforms. [12] report that industry-leading platforms such as Sensity AI achieve 95-98% accuracy, while Intel FakeCatcher demonstrates 96% accuracy with millisecond-level latency using photoplethysmography-based detection. Reality Defender provides 90-95% accuracy across multimodal content (video, audio, image), and DeepBrain AI reports

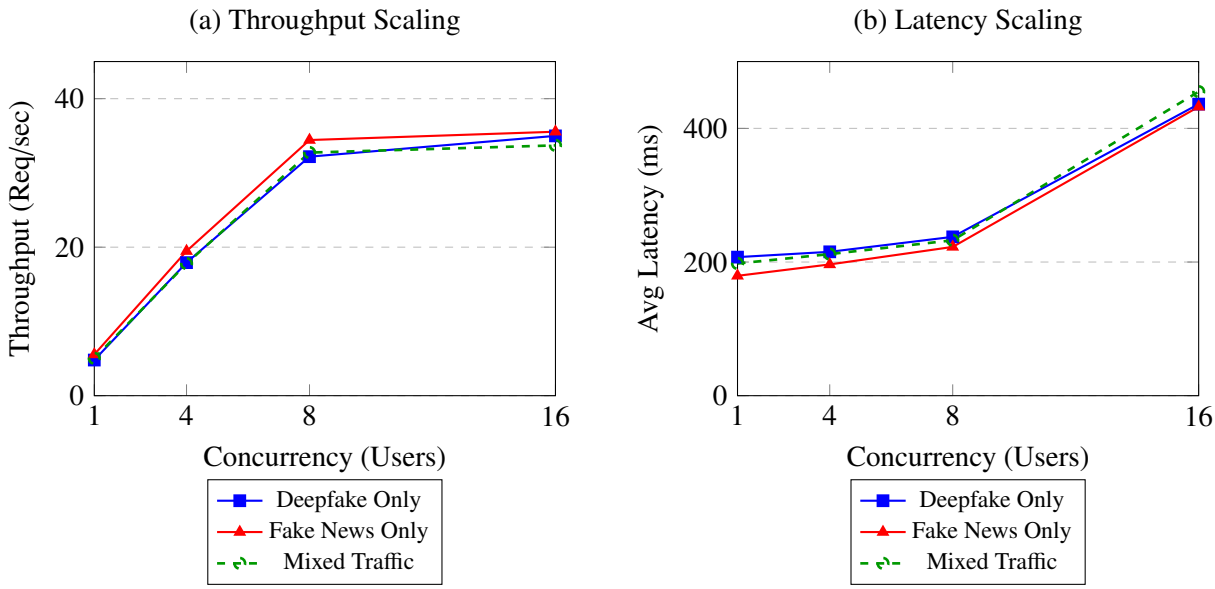

Figure 4. System performance comparison.

accuracy exceeding 90% with processing times of 5-10 minutes per video. Budget-oriented solutions include Hive AI ($25/month, 85-90% accuracy) and Deepware ($8/month, 80-85% accuracy), while enterprise platforms operate on custom licensing without disclosed pricing. In total, DeepSentry achieves competitive accuracy (94%) while maintaining real-time performance suitable for heterogeneous inference workloads.

Table 3. Comparative Analysis of Commercial Misinformation Detection Platforms [12]

Platform	**Accuracy**	**Latency**	**Pricing**
Sensity AI	95-98%	Real-time	Enterprise
Intel FakeCatcher	96%	Milliseconds	Enterprise
Reality Defender	90-95%	Real-time	Freemium
DeepBrain AI	90%+	5-10 min	$24-216/mo
Hive AI	85-90%	2-5 sec	$25/mo+
Deepware	80-85%	3-5 min	$8/mo
DeepSentry	**94-95%**	**198-715ms**	**On-prem/Cloud**

Beyond these metrics, the operational effectiveness of a detection system is defined by its transparency and adaptability. As illustrated in Table 4, DeepSentry introduces several critical capabilities designed for high-stakes forensic environments.

The integration of **Explainable AI (XAI)** through attention heatmaps provides visual evidence of manipulation, transforming the system from a ”black-box” classifier into a forensic support tool. This is particularly vital for Vietnamese news agencies and legal entities where justification of a ”fake” label is required. Furthermore, unlike closed-source commercial APIs, DeepSentry’s **Custom Model Support** allows organizations to integrate new detection kernels as generative threats evolve, ensuring the platform remains future-proof.

Table 4. Qualitative feature comparison: DeepSentry vs. other platforms

Feature	**Deep Sentry**	**Sensity AI**	**Hive AI**	**Faceless**
Technical Capabilities				
Explainable AI (Heatmaps)	✓	✓	–	–
Custom Model Support	✓	–	–	–
Deployment & Sovereignty				
On-premise Deployment	✓	✓	–	✓
Full Data Privacy Control	✓	–	–	✓

A pivotal differentiator is the necessity of **data sovereignty** in the context of Vietnam’s

cybersecurity landscape. While international platforms offer robust features, they typically rely on cloud-based processing that requires sensitive media to leave the domestic jurisdiction, posing significant privacy risks. DeepSentry's support for **on-premise deployment** ensures that government agencies and financial institutions maintain full privacy control over their data. Consequently, DeepSentry provides detection across diverse manipulation techniques, and as discussed in the Introduction and surveyed in the Related Work section, the first of its kind with commercial readiness in Vietnam.

5. Conclusion

This paper presented DeepSentry, an asymmetric microservice-based platform that addresses critical resource contention challenges in misinformation detection systems. By decoupling CPU-intensive preprocessing from GPU-bound inference through message queuing and dedicated inference servers, the proposed architecture eliminates Head-of-Line blocking and GPU starvation inherent in monolithic deployments. Experimental validation demonstrated that DeepSentry maintains consistent throughput (33.71 RPS) and latency (454ms at 16 concurrent users) under mixed workloads, achieving 94-95% detection accuracy across both deepfake and fake news detection. The system architecture enables independent scaling of heterogeneous computational resources while integration with NVIDIA Triton's dynamic batching mechanism maximizes GPU utilization without compromising per-request latency.

Future work will focus on three primary directions: (i) academic advancement through construction of Vietnamese-specific deepfake and fake news datasets to address the current scarcity of regional training corpora, enabling improved generalization to local linguistic patterns and cultural contexts; (ii) algorithmic enhancement by incorporating emerging detection techniques for latest generative models and multimodal fusion architectures; and (iii) commercial deployment as Vietnam's first specialized media forensics platform, targeting news agencies, financial institutions, and government entities requiring self-hosted verification capabilities beyond existing eKYC solutions.

References

[1] Advancements in Detecting Deepfakes: AI Algorithms and Future Prospects, Discover Internet of Things (May 2025).

[2] N. Chandra, et al., Deepfake-Eval-2024: A Multi-Modal In-the-Wild Benchmark of Deepfakes Circulated in 2024, arXiv preprint arXiv:2503.02857 (Mar. 2025).

[3] S. M. Abdullah, A. Cheruvu, S. Kanchi, T. Chung, P. Gao, M. Jadliwala, B. Viswanath, An Analysis of Recent Advances in Deepfake Image Detection in an Evolving Threat Landscape, in: Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2024, pp. 1–18.

[4] Datadog, ML Platform Monitoring: Best Practices, https://www.datadoghq.com/blog/managed-ml-best-practices/ (2024).

[5] Global Multimedia Deepfake Detection Challenge: Towards Multi-dimensional Face Forgery Detection, arXiv preprint arXiv:2412.20833 (Jun. 2024).

[6] H. Padalko, V. Chomko, D. Chumachenko, A Novel Approach to Fake News Classification Using LSTM-based Deep Learning Models, Frontiers in Big Data 6 (Jan. 2024).

[7] BBC-FND: An Ensemble of Deep Learning Framework for Textual Fake News Detection, Computers and Electrical Engineering 110 (Sep. 2023).

[8] Ensemble Based High Performance Deep Learning Models for Fake News Detection, Scientific Reports 14 (26591) (Nov. 2024).

[9] Multimodal based Amharic Fake News Detection Using CNN and Attention-based BiLSTM, Scientific Reports (Oct. 2025).

[10] D. T. Dong, T. M. Pham, Faceless: A Deepfake Detection Toolkit, BlackHat Asia, Singapore (2023).

[11] T. M. Pham, Faceless: Open-Source Deepfake Detection System, GitHub repository, https://github.com/ManhNho/Faceless (2023).

[12] Best AI Video Detector Tools 2025: Comprehensive Comparison & Review, https://www.aivideodetector.

org/blog/best-ai-video-detector-tools-2025, online; accessed 2025 (Aug. 2025).

[13] Vietnam Posts and Telecommunications Group, VNPT eKYC – the first Vietnamese AI identification and authentication platform to exceed 1 billion requests.

[14] MultimodalAI, NVIDIA Triton Inference Server Made Simple, Medium (2024).

[15] V. D. Vinh, D. Phuc, Detecting Vietnamese Fake News, journal = CTU Journal of Innovation and Sustainable Development 15 (2023) 39–46.

[16] H. Wu, J. Zhou, Image Forensics for Social Networks, GitHub repository, https://github.com/HighwayWu/ImageForensicsOSN (2022).

[17] A. Graves, J. Schmidhuber, Framewise Phoneme Classification with Bidirectional LSTM Networks, in: Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 2005, pp. 2047–2052.

[18] B. Dolhansky, et al., The Deepfake Detection Challenge Dataset, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 10838–10847.

[19] D. W. Deressa, H. Mareen, P. Lambert, S. Atnafu, Z. Akhtar, G. Van Wallendael, GenConViT: Deepfake Video Detection Using Generative Convolutional Vision Transformer, Applied Sciences (2025).