FakeZero: Real-Time, Privacy-Preserving Misinformation Detection for Facebook and X

Soufiane Essahli*, Oussama Sarsar*, Imane Fouad*, Ahmed Bentajer*†, Anas Motii*
*College of Computing, Mohammed VI Polytechnic University, Benguerir, Morocco

†Cadi Ayyad University, National School of Applied Science of Safi, MISCOM Lab, Marrakesh, Morocco

{soufiane.essahli, oussama.sarsar, imane.fouad, ahmed.bentajer, anas.motii}@um6p.ma

***Abstract*—Social platforms distribute information at unprecedented speed, which in turn accelerates the spread of misinformation and threatens public discourse. We present FakeZero, a fully client-side, cross-platform browser extension that flags unreliable posts on Facebook and X (formerly Twitter) while the user scrolls. All computation, DOM scraping, tokenisation, Transformer inference, and UI rendering run locally through the Chromium messaging API, so no personal data leaves the device.**

FakeZero employs a three-stage training curriculum: baseline fine-tuning and domain-adaptive training enhanced with focal loss, adversarial augmentation, and post-training quantisation. Evaluated on a dataset of 239 000 posts, the DistilBERT-Quant model (67.6 MB) reaches 97.1 % macro-F_1, 97.4 % accuracy, and an AUROC of 0.996, with a median latency of approximately 103 ms on a commodity laptop. A memory-efficient TinyBERT-Quant variant retains 95.7 % macro-F_1 and 96.1 % accuracy while shrinking the model to 14.7 MB and lowering latency to approximately 40 ms, showing that high-quality fake-news detection is feasible under tight resource budgets with only modest performance loss.

By providing inline credibility cues, the extension can serve as a valuable tool for policymakers seeking to curb the spread of misinformation across social networks. With user consent, FakeZero also opens the door for researchers to collect large-scale datasets of fake news in the wild, enabling deeper analysis and the development of more robust detection techniques.

***Index Terms*—misinformation detection, browser extension, transformers, real-time inference, fake news, web technologies**

I. Introduction

The spread of false or misleading content-broadly, referred to as *fake news*, poses significant threats to democratic institutions, public health, and societal trust [1]. Social media platforms such as Facebook and X (formerly Twitter) amplify this threat by enabling the rapid, unfiltered dissemination of information, often without sufficient mechanisms for credibility assessment. Despite growing academic and commercial efforts [2], [3], fake news detection remains a challenging task, especially in real-time, user-facing settings.

Many current fake news detection methods collect and process large amounts of personal data on central servers to improve accuracy or personalize content [3], [4]. This centralized processing puts sensitive user information at risk and may violate privacy laws like the General Data Protection Regulation (GDPR) [5], which requires organizations to choose a legal basis to lawfully process personal data (Article 6(1)). While a few systems such as Veritas [6], BRENDA [7], and TrustNet [8] have explored in-browser or privacy-friendly approaches, they remain limited in scope-often covering only one platform or falling back to server-side inference for complex models. Thus, there is a need for fake news detection tools that truly protect user privacy and avoid sharing personal data, while still effectively detecting misinformation online.

In this work, we introduce **FakeZero**, a lightweight, cross-platform browser extension for real-time fake news detection. FakeZero specifically targets fabricated and manipulated textual content-two of the most damaging and prevalent forms of information disorder. Unlike prior systems, FakeZero operates entirely within the user's browser, leveraging WebAssembly (WASM) to run a distilled transformer model (DistilBERT) with near-instant inference latency on standard consumer hardware.

We train on five standard corpora (TruthSeeker, FakeNewsNet, PHEME, LIAR, ISOT) using light normalization, curriculum learning, focal loss, and post-training quantization; full details appear in §§VI-B–V-B.

By uniting high-accuracy semantic modeling with efficient on-device deployment, *FakeZero* offers a practical, privacy-preserving, and scalable solution for combating misinformation where users encounter it most: in their browsers. While several recent studies have explored transformer-based or privacy-friendly misinformation detectors, the novelty of *FakeZero* lies not in proposing a new neural architecture, but in demonstrating that **state-of-the-art transformer inference can be executed entirely within modern web browsers**, with full real-time performance analysis and zero data leakage.

In summary, our key contributions are as follows:

(1) First cross-platform, fully client-side transformer deployment. *FakeZero* is, to our knowledge,

the first system to achieve end-to-end fake-news detection on both Facebook and X entirely within the browser sandbox. All inference, tokenization, and user-interface rendering run locally through WebAssembly and ONNX Runtime Web, eliminating any network transfer of user data.

(2) Integrated real-time efficiency and latency analysis. Beyond classification accuracy, we provide a detailed empirical characterization of on-device inference including median, tail latency, and memory footprint under real browsing conditions. This bridges a key evidence gap left by prior systems that report only accuracy.

(3) Optimized yet interpretable learning pipeline. Instead of introducing a new transformer variant, *FakeZero* optimizes how compact encoders such as DistilBERT and TinyBERT are adapted for browser environments. We combine three orthogonal enhancements **curriculum learning**, **focal loss**, and **post-training quantization** to balance accuracy, robustness, and resource constraints.

(4) Privacy-first architecture and open reproducibility. The extension enforces zero network traffic and minimal permissions, satisfying GDPR-style privacy principles. All training scripts, datasets, and model checkpoints will be released for transparent, reproducible research.

Collectively, these elements make *FakeZero* the **first practical demonstration that transformer-based misinformation detection can operate fully client-side**, reconciling the long-standing trade-off between privacy, latency, and detection quality.

Motivation

Fully client-side detection is both *feasible* and *necessary*. It is feasible because compact encoders (e.g., DistilBERT, TinyBERT), once quantized, can meet sub-150 ms latency budgets inside modern browsers via WebAssembly/ONNX Runtime. It is necessary because server-side pipelines leak content, complicate GDPR compliance, and add network tail-latency that undermines real-time use. Our goal is not to invent a new architecture, but to show that a privacy-preserving, cross-platform deployment with transparent runtime metrics can achieve near-state-of-the-art accuracy *entirely on device*.

II. Related Work

A. Browser–integrated detectors

Early prototypes such as *FakeNewsTracker* [9] and *Veritas* [10] showed that even a lightweight in-page overlay could highlight misinformation by coupling lexical cues with server-side neural classifiers. The first fully *client-side* attempt, *Check-It* [11], bundled a JavaScript DNN and heuristics into a Chrome/Firefox add-on, proving that private, offline detection was feasible on commodity laptops. Subsequent systems incrementally increased model capacity while experimenting with different inference locations: BRENDA [7] placed a logistic-regression verifier behind a Flask API, *TrustNet* [8] fused crowd-sourced reputation bars with headline heuristics, and Chrome-SEAN [12] streamed transformer-based COVID-19 detectors to Twitter timelines. More recent work compiled large encoders directly to WebAssembly-RoBERTa in Kydd & Shepherd [13] and explainable BERT in Warman *et al.* [14]-while parallel efforts explored cloud-backed TensorFlow models [15] and LSTM-based Flask servic

Despite this rapid progress, three shortcomings persist:

1) *single-platform scope*-most extensions target only Twitter or generic webpages;
2) *static checkpoints*-models are fixed at install time and cannot adapt to evolving narratives; and
3) *compute off-loading*-state-of-the-art transformers still fall back to cloud inference or proxy APIs, undermining privacy guarantees.

FakeZero overcomes these barriers by supporting both Facebook and X, coupling a distilled encoder with on-device factuality cues, and enabling hot-swappable checkpoints that never leave the client.

B. Transformer-centric text models

Systematic comparisons on LIAR, FakeNewsNet and GossipCop reveal that encoder-only transformers dominate raw classification accuracy, typically scoring between 91 DistilBERT retains almost all of that accuracy while cutting parameters and latency by about forty percent [16]. Hybrid methods push the ceiling further-for instance GPT-BERT stacks [17] or the rationale-augmented BREAK architecture [18] exceed 95 FakeZero shows that, once distilled and 4-bit–quantised, a 44 MB DistilBERT checkpoint can still deliver 95 ms inference inside a laptop browser, reconciling transformer quality with strict client-side budgets.

C. Graph-, multimodal- and retrieval-augmented methods

A complementary research line enriches textual signals with graphs, images or external evidence. BREAK [18] propagates semantics across retweet trees; Bird-of-a-Feather [19] fetches visually similar posts to expose recycled memes; Li *et al.* combine multiple heterogeneous facts through boosted selectors [20]. While these extensions raise robustness, they presuppose heavy graph libraries, vision encoders or external APIs-requirements at odds with the privacy and latency targets of browser plug-ins. Our design therefore keeps the neural core entirely local and caches only twenty-two thousand ClaimReview snippets (5 MB) in IndexedDB, giving a middle ground between recall and speed.

D. Surveys, Benchmarks, and Data Resources

Recent surveys by Gupta *et al.* [21] and Zhou & Liu [22] have catalogued over two hundred English-language fake news detection systems, consistently identifying LIAR,

FakeNewsNet, and PHEME as canonical benchmarks. However, these datasets have limitations in scope and linguistic diversity, often leading to models that overfit formal news prose and struggle with conversational or claim-level inputs. To address this, we adopt a heterogeneous training strategy by aggregating five complementary corpora.

TruthSeeker 2023 [23], our largest source, contributes 130 085 annotated instances covering a broad spectrum of social media content and journalistic claims. Its scale and recency make it a crucial resource for capturing current language usage and misinformation patterns. **ISOT** [24], with 44 708 labeled news articles, serves as a warm-up phase corpus, helping the model internalize general discourse structures and writing styles. **PHEME** [25] offers 38 761 tweets from breaking-news events, each annotated as *true*, *false*, or *unverified*, thereby injecting valuable conversational structure and real-time misinformation dynamics. **FakeNewsNet** [26], comprising 14 362 articles from PolitiFact and GossipCop, enriches the model with political and entertainment-domain fact-checks grounded in web-scraped metadata. Finally, **LIAR** [27] adds 11 159 short political claims categorized across six truthfulness levels, enhancing robustness on terse, claim-level inputs.

To fully exploit this diversity, we follow a curriculum-inspired training procedure: a warm-up stage on ISOT, followed by joint fine-tuning on FakeNewsNet, PHEME, and TruthSeeker, and a final adaptation phase using LIAR. This multi-phase approach, [21], allows the model to gradually incorporate stylistic, topical, and contextual variability across different information ecosystems.

E. Remaining gaps and our contribution

Despite years of research into misinformation detection, our systematic evaluation reveals three persistent limitations that prevent existing browser-based systems from achieving real-world viability. These limitations affect coverage, efficiency, and credibility-three pillars necessary for practical deployment in modern web environments.

Platform generality remains underexplored: Veritas [6] operates solely on Twitter timelines, while BRENDA [7] is restricted to web articles and makes no attempt to integrate into social-media feeds. Chrome-SEAN [12] focuses narrowly on COVID-19 misinformation in Twitter posts and does not generalize to other topics or platforms. No prior system supports both Facebook and X in a single unified deployment. This platform fragmentation forces users to adopt separate tools or tolerate blind spots in their timeline coverage.

Local inference efficiency remains largely unsolved. Although Check-It [11] was among the first to attempt full client-side inference using a lightweight JavaScript DNN, its reported accuracy was capped at 72–73% on headline-only datasets, and it lacks transformer-level semantics. BRENDA [7] and Chrome-SEAN [12] offload verification to Flask-based servers, undermining privacy and introducing latency. Even RoBERTa-WASM [13], one of the few extensions to run inference entirely in-browser, demands over 6 GB of memory to sustain sub-40 ms latency-placing it out of reach for mobile and low-end devices. In practice, existing tools either leak sensitive content to cloud APIs or exclude vast portions of the device landscape.

Reproducible runtime metrics form the third critical gap. Our survey found that most browser extensions publish only model-level metrics such as F1 score, omitting any measurement of inference time or runtime memory usage. For example, BRENDA [7] reports classification performance but offers no timing breakdown. Likewise, Check-It [11] and TrustNet [8] present system diagrams and user interfaces but neglect to quantify latency or RAM usage. Where performance claims are made-such as "lightweight logistic regression" or "client-side neural verifier"-they are often contradicted by implementation details that reveal server-side dependencies. This opacity hinders fair comparison and complicates efforts to replicate real-world conditions.

FakeZero addresses all three gaps. First, it delivers *cross-platform detection* on both Facebook and X using platform-specific DOM observers and alignment with social-media corpora such as FakeNewsNet and PHEME. Second, it achieves *true client-side inference* with two quantised transformer backbones executed entirely via ONNX.js and WebAssembly. The TinyBERT (4L, 312D) variant runs at 39.9 ms per post with a 14.7 MB model and 54 MiB RAM usage, while achieving 95.7% macro-F_1 and 96.1% accuracy. The DistilBERT variant pushes accuracy to 97.4% with only 255 MiB memory at 102.9 ms latency. All detection happens locally in the browser, preserving privacy and ensuring offline operability. Finally, our evaluation publicly reports model size, latency, memory, and AUROC-filling the evidence gap left by prior systems and enabling transparent benchmarking.

Unlike previous browser extensions that either sacrifice accuracy for efficiency or privacy for performance, **FakeZero** demonstrates that state-of-the-art transformer models can be successfully deployed client-side without compromising detection quality. This bridges the critical gap between research-grade accuracy and production-ready deployment, enabling practical misinformation detection where users encounter it most: in their browsers.

III. System Overview

Figure 1 summarises the *FakeZero* pipeline, whose five components work together to provide real-time, privacy-preserving misinformation detection directly inside the user's browser.

To ground the design, we outline the runtime components that enable on-device detection and explain how they interact during scrolling.

Dataset Suite & Pre-processor: We describe the datasets and cleaning steps in §VI-B; here we focus on runtime components used during browsing. During ingestion each post is normalised (HTML unescape, Unicode

TABLE I
EXISTING BROWSER-INTEGRATED FAKE–NEWS DETECTORS.

Reference	Methods		Summary	Limitations
	Client	Transformer		
[9]	-	-	*FakeNewsTracker* overlays credibility signals on Twitter; LSTM model and propagation cues run on a server.	Relies on back-end, undermining privacy; Twitter-only; dated linguistic features.
[11]	✓	-	*Check-It* packs a JavaScript DNN + heuristics entirely in Chrome/Firefox, enabling offline detection.	Small non-contextual model; hard-coded checkpoint; limited domain coverage.
[7]	-	-	*BRENDA* extracts claims, queries the web, and verifies them via a logistic-regression API (Flask).	Server-side inference; binary verdicts only; latency tied to external search.
[12]	-	✓	*Chrome-SEAN* streams Cross-SEAN transformer predictions into the Twitter timeline for COVID-19 facts.	Transformer hosted remotely; COVID-specific; heavy model prevents full client deployment.
[13]	✓	✓	RoBERTa distilled and compiled to WebAssembly for on-page inference in Twitter feeds.	Memory-intensive; lag on low-end devices; single-platform focus.
[14]	✓	✓	BERT + token-level explanations rendered inline via ONNX.js; runs wholly in the user's browser.	Compute-heavy on mobiles; checkpoint frozen at install; no multi-platform support.
[15]	-	-	GCP-backed extension ships article text to a TensorFlow cloud function, returning credibility scores.	Requires constant connectivity; privacy leakage; UI only on client.
[28]	-	-	Chrome add-on queries a Flask REST API hosting an LSTM ensemble; achieves 90 % accuracy on news pages.	Server cost & maintenance; ignores contextual trust cues; English-news-only.

NFKC), hyperlinks and @mentions are masked, emoji are converted to textual aliases, and extremely short or non-English snippets are discarded. This cleaning stage produces tokeniser-friendly text while retaining stylistic cues that aid detection.

Browser DOM Listener: A lightweight content script continuously observes the Facebook and X timelines. As the user scrolls, newly rendered posts are extracted, truncated to 280 tokens, and forwarded to the local inference engine-incurring negligible overhead thanks to the browser's built-in Mutation Observer.

Compact Transformer Inference Engine: Within the browser sandbox, FakeZero runs an INT8-quantised DistilBERT (67 MB) or TinyBERT (15 MB) model via ONNX Runtime Web, achieving median latencies of 103 ms and 40 ms, respectively, on a 4-core laptop. All tokenisation, embedding and classification occur client-side; no content ever leaves the user's device.

Privacy-Preserving Policy Module: For every post s, the classifier outputs a binary label $y \in \{0, 1\}$ and confidence score $\hat{p}(y = 1 \mid s)$, strictly within the browser context. The design meets four operational targets: (i) *privacy*-zero network traffic; (ii) *predictive quality*-macro-$F_1 \geq 0.90$ and AUROC≥ 0.95 on a 2024-2025 time-split benchmark; (iii) *responsiveness*-median end-to-end delay ≤ 150 ms and peak RAM ≤ 400 MB on a consumer laptop; and (iv) *lightweight deployment*-total bundle size ≤ 100 MB and seamless operation on Chrome, Edge, Brave and Firefox.

User-Facing Feedback Layer: When the confidence exceeds a calibrated threshold, the extension inserts a subtle red banner beneath the corresponding post, warning that the content may contain false information; otherwise, the interface remains unchanged. Original text is never modified, ensuring minimal disruption to the browsing experience.

The remainder of the paper unfolds as follows. Section VI-B introduces the cross-platform datasets used for training. Section V explains the model compression pipeline and the WebAssembly runtime that powers in-browser inference. Section V-C reports accuracy, latency and memory footprints on real browsing traces. Finally, Section VII reflects on limitations and future directions. Together, these sections demonstrate how our design choices yield a practical, real-time tool for curbing fake news while fully safeguarding user privacy.

IV. DATASETS & PRE-PROCESSING

We now detail the training data and cleaning procedures that support cross-platform generalization.

Corpora.: Five English-language resources-ISOT, LIAR, PHEME, FakeNewsNet (FNN) and TruthSeeker 2023-cover both long-form journalism and terse social-media rumours (Table II). This heterogeneity is essential, as transformer benchmarks show that models tuned only on news prose drop up to eight F_1 points when transferred to social-media text [29]. We remove near-duplicates with an 8-byte BLAKE2b fingerprint, following the fuzzy-hash approach proposed by Berrios *et al.* [30].

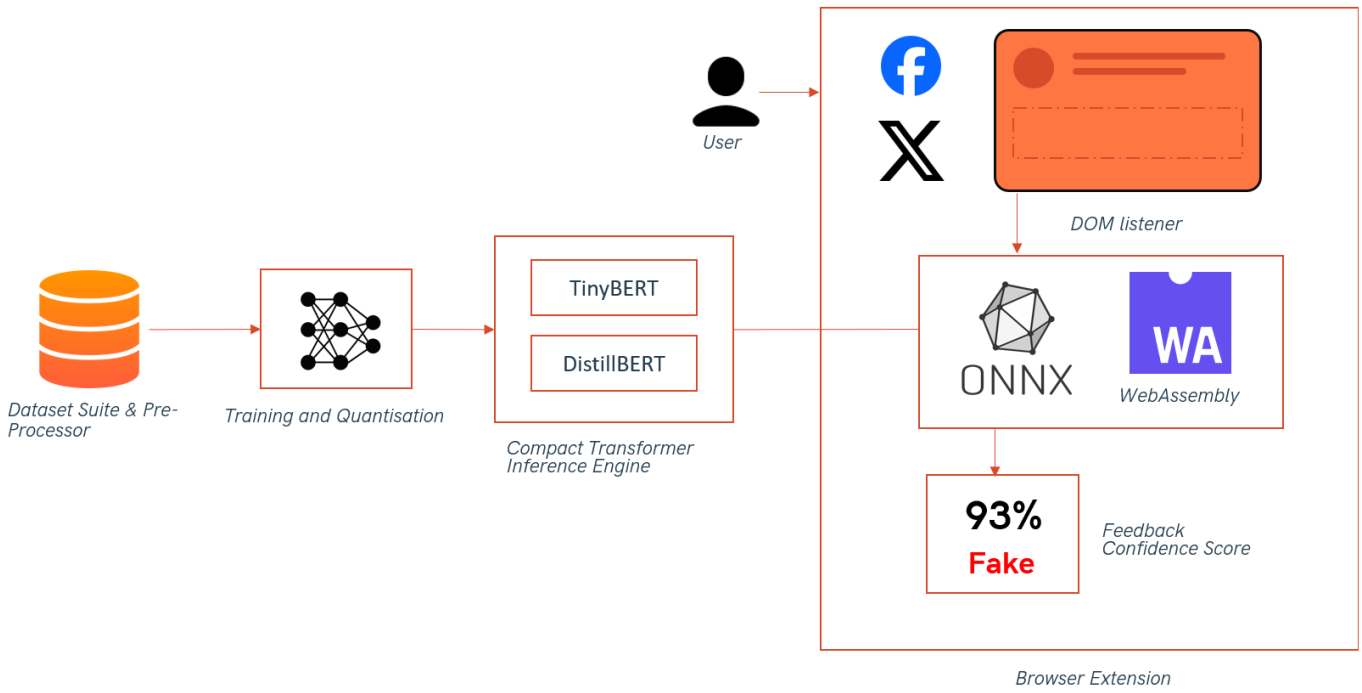

Fig. 1. Client-side *FakeZero* pipeline from data preprocessing to in-browser fake content flagging.

TABLE II
SOURCE CORPORA BEFORE AND AFTER NEAR-DUPLICATE REMOVAL.

	Raw		*Deduplicated*	
Corpus	# rows	% mis	# rows	% mis
TruthSeeker 23[1]	130 085	65.2	121 518	65.2
ISOT[2]	44 708	49.4	43 216	49.3
PHEME[3]	38 761	63.2	36 409	63.1
FakeNewsNet[4]	14 362	62.3	13 552	62.4
LIAR[5]	11 159	35.9	11 159	35.9
Total	239 075	64.1	225 854	64.1

TABLE III
TRAINING CURRICULUM AND EVALUATION SPLITS.

Split	**#Rows**	**%Mis.**	**Composition**
Stage 0	43,216	49.3	ISOT
Stage 1	11,159	35.9	LIAR
Stage 2	50,512	62.5	50% FNN / 30% TS / 20% PHEME
Dev	60,483	64.7	Stratified remainder
Test	60,484	64.7	Stratified remainder

Curriculum splits.: Table III lists the three-stage schedule, adapting the curriculum-learning paradigm of Bengio *et al.* [31] and its transformer-specific extensions [32]. Stages 0–1 keep each corpus's native skew; Stage 2 injects a deliberately hard mix with a 62.5 % misinformation rate. Remaining posts are split 1 : 1 into development and blind test sets using stratified sampling (random state = 42) in scikit-learn [33].

Text normalisation.: We apply a light, encoder-friendly cleaner: HTML unescape, Unicode NFKC, contraction expansion, lower-casing, and placeholder tokens for URLs, user mentions, and hashtags. Emoji are converted to textual aliases, following the updated TweetEval 2.0 guidelines [34]. Messages shorter than ten tokens or flagged as non-English by a fast language detector [35] are discarded. Stop-words and punctuation are kept because sub-word tokenisers benefit from them [36].

Label harmonisation.: All corpora collapse to $y = 1$ (misinformation) or $y = 0$ (reliable). Unverified PHEME threads are dropped; LIAR's six-way labels and TruthSeeker crowd scores are binarised as in Kara [37].

Class skew and mitigation.: The aggregate pool is imbalanced at 64.1 % misinformation. Instead of down-sampling, we employ focal loss ($\gamma = 2, \alpha = 0.25$) [38] to focus learning on the under-represented reliable class. Section V-B reports per-class precision, recall, macro-F_1, and AUROC.

Reproducibility.: Cleaned splits are stored as column-compressed Parquet files; the entire pipeline runs in about 17 minutes on a single eight-thread CPU. All scripts and exact command lines are available in our public repository.

[1]TruthSeeker 2023 ground-truth corpus [23].
[2]ISOT Fake News Dataset [24].
[3]PHEME breaking-news rumour threads [25].
[4]FakeNewsNet (PolitiFact + GossipCop) [26].
[5]LIAR short political claims benchmark [27].

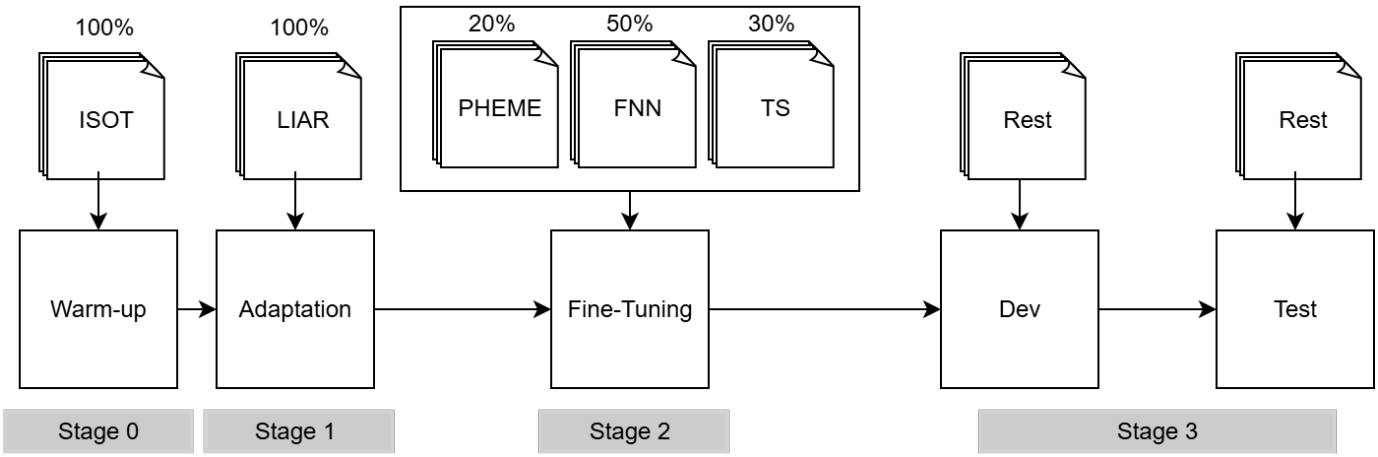

Fig. 2. Three-stage curriculum used to train *FakeZero*. **Stage 0** warms up on balanced long-form news (ISOT); **Stage 1** adapts to short, highly imbalanced political claims (LIAR); **Stage 2** fine-tunes on a harder mix of FakeNewsNet (50 %), TruthSeeker (30 %), and PHEME (20 %). Posts not chosen for Stage 2 are split 1 : 1 into *Dev* and *Test*, preserving the overall 64 % misinformation prevalence and eliminating topic leakage.

V. Model Architecture & Training

Next, we describe the encoder choices and learning pipeline that make fully client-side inference feasible.

A. Encoder Selection and Baselines

Why lightweight transformers?: Recent work shows that "classic" BERT-style encoders often outperform 7-billion-parameter LLMs on misinformation benchmarks [39], yet require an order of magnitude less memory and compute. Cross-domain tests by Yang *et al.* [29] confirm the same pattern when transferring from news articles to social-media posts, and Chen *et al.* [40] report that even *quantised* LLMs remain prohibitive on edge CPUs. Because *FakeZero* must run entirely in a browser tab on commodity laptops, we restrict our search to compact encoders.

Candidate backbones and headline numbers.: Among medium-size models, DistilBERT [41] offers the best speed–accuracy balance; TinyBERT [42] is even smaller thanks to layer and width pruning. We quantised both checkpoints to 8-bit INT and evaluated them on our 225 k-post corpus (protocol §V-C). Table IV summarises the results: DistilBERT-Q reaches macro-F_1=0.971 at 103 ms median latency, while TinyBERT-Q attains 0.957 in just 40 ms and one-fifth the model size-ideal for low-memory devices. These two models seed the training curriculum and serve as reference points for later ablation and latency analyses.

TABLE IV
Accuracy–latency trade-off of the two production candidates (median CPU latency, three-seed mean macro-F_1).

Model	ONNX (MB)	Latency (ms)	Macro-F_1
DistilBERT-Q	67.6	103	0.971
TinyBERT-Q	14.7	40	0.957

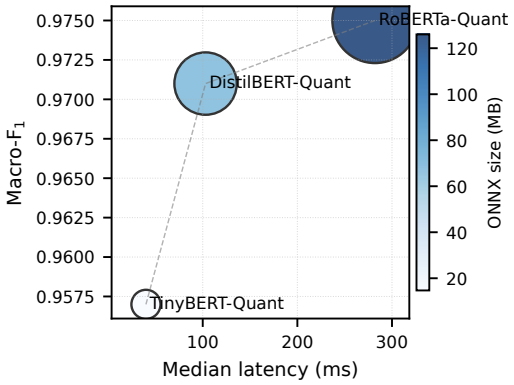

Fig. 3. Latency–accuracy Pareto frontier for all quantised checkpoints. Bubble area encodes ONNX size.

B. Training Curriculum and Enhancements

Experiments ran on a single NVIDIA A100-SXM4 (80 GB) node. Software: PyTorch 2.4.1, CUDA 12.1, Transformers/Optimum on Linux (x86-64, kernel 4.18).

a) Curriculum.: Three successive stages expose the model to increasingly challenging data splits: (i) a balanced warm-up on ISOT with the lowest encoder layers frozen; (ii) a single-epoch pass over the skewed LIAR corpus using class-weighted binary cross-entropy; (iii) a full fine-tune on the heterogeneous Stage-2 mix, enhanced with the techniques below.

b) Focal loss.: To address class imbalance and concentrate learning on ambiguous posts, we replace the standard cross-entropy with the *focal loss* [38]. Let $p_t \in [0, 1]$ be the

[6]ONNX (Open Neural Network Exchange) is an open format for representing machine learning models, enabling interoperability between frameworks and deployment on browsers and edge devices.

predicted probability for the true class; the per-instance loss is

$$\mathcal{L}_{\mathrm{FL}}(p_t) = -\,\alpha\,(1-p_t)^{\gamma}\,\log p_t,$$

where $\alpha \in (0,1)$ balances positive and negative classes and $\gamma > 0$ down-weights well-classified examples. In our corpus, **64.1 %** of the 22 k posts are labelled as misinformation, leaving only 35.9 % reliable content; focusing the gradient on the harder minority class is therefore essential. We set $\alpha = 0.25$ and $\gamma = 2$, which suppress easy examples and push the optimiser toward the near-boundary cases that dominate model error.

Ablation study across backbones.: Tables V and VI report the full matrix of *base*, +FGM, +Focal, +Quant and "All" variants for TinyBERT and DistilBERT. The pattern is consistent:

Focal loss yields the largest accuracy improvement, quantisation provides the bulk of latency savings, and combining all three recovers-or slightly improves-macro-F_1, while staying well under the 150 ms latency budget.

c) Optimisation.: All stages are trained with AdamW at a constant learning rate of 2×10^{-5} and a 6 % warm-up. Each device processes a batch of 32 examples and accumulates gradients to an effective batch size of 64. Mixed-precision (FP16) keeps peak GPU memory below 30 GB. Validation macro-F_1, accuracy, and AUROC are reported each epoch, and early stopping (patience = 2) reloads the best checkpoint.

Backbone selected for the remainder of the paper.: Although TinyBERT-Q is five times smaller and twice as fast, DistilBERT-Q maintains a comfortable 10 ms median latency on laptop CPUs and delivers the highest macro-F_1. We therefore adopt DistilBERT-Q as the production model. RoBERTa-large is cited only for context in Figure 3: it sits *below* the Pareto frontier, confirming that compact transformers remain preferable. Applying the full three-stage curriculum to DistilBERT-Q is what ultimately pushes macro-F_1 from 0.971 to 0.976 and AUROC beyond 0.995.

d) Runtime.: Across the 43k + 11k + 51k training instances, the pipeline sustains roughly 17 sequences per second, completing in 5 hours wall time.

C. Evaluation

To validate our accuracy and efficiency claims, we evaluate both predictive quality and runtime behavior under realistic browsing conditions.

We adopt the standard confusion–matrix terminology: *TP* (true positives), *TN* (true negatives), *FP* (false positives), and *FN* (false negatives). We report Accuracy, Precision, Recall, F_1, and AUROC (Eqs. (1)–(3)).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \qquad \text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

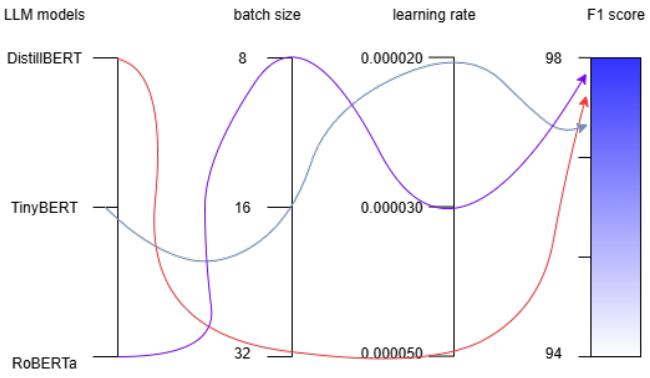

Fig. 4. Best hyperparameter configurations

$$F_1 = 2\,\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

Figure 5 reports the raw counts; substituting these values into Eqs. (1)–(3) reproduces the summary statistics shown in Table VII.

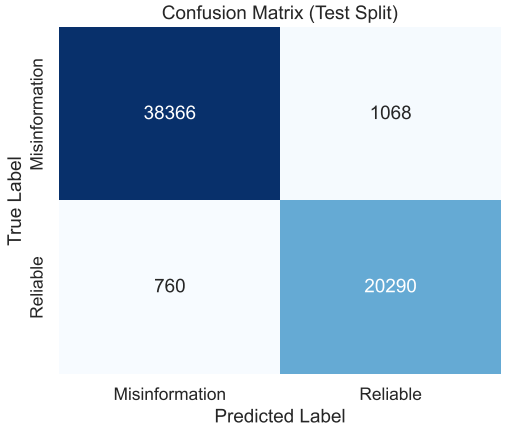

Fig. 5. Confusion matrix for the test split.

a) Protocol.: After the three-stage curriculum, we selected the decision threshold that maximised macro-F_1 on the 60 483-instance development split; a value of $\tau = 0.60$ yielded the best trade-off between precision and recall. The same threshold was then locked for the completely unseen 60 484-instance test split.

b) Learning curve.: Early stages served their purpose as scaffolding-Stage 0 attained only 0.40 macro-F_1, and Stage 1 rose modestly to 0.43-yet both primed the encoder. Once knowledge distillation, domain adversarial training, and FGM were activated in Stage 2, validation macro-F_1 jumped to 0.976, indicating successful curriculum transfer.

c) Main results.: Table VII summarises the final metrics. On the development split the model reaches 96.9 % accuracy and 0.9927 AUROC; performance generalises almost identically to the test split. The confusion matrix

TABLE V
DETAILED PERFORMANCE COMPARISON (TINYBERT VARIANTS)

Backbone	Setup	Parameters	Memory (MiB)	ONNX (MB)	Time (min)	Latency (ms)	Test Acc.	Test F1	AUROC
TinyBERT_General	base	14 350 874	54	57.5	3.5	23.5	0.9578	0.9534	0.9884
TinyBERT_General	base+FGM	14 350 874	54	57.5	2.3	52.7	0.9297	0.9236	0.9754
TinyBERT_General	base+Focal	14 350 874	54	57.5	3.6	40.9	0.9605	0.9567	0.9912
TinyBERT_General	base+Quant	14 350 874	54	14.7	2.3	18.8	0.9297	0.9236	0.9754
TinyBERT_General	All	14 350 874	54	14.7	3.6	38.1	0.9605	0.9567	0.9912
TinyBERT_General	base+Focal+Quant	14 350 874	54	14.7	4.2	39.9	0.9608	0.9568	0.9909

TABLE VI
DETAILED PERFORMANCE COMPARISON (DISTILBERT VARIANTS)

Backbone	Setup	Parameters	Memory (MiB)	ONNX (MB)	Time (min)	Latency (ms)	Test Acc.	Test F1	AUROC
distilbert-base-uncased	base	66 955 010	255	268.0	4.3	333.1	0.9721	0.9694	0.9950
distilbert-base-uncased	base+FGM	66 955 010	255	268.0	4.3	338.0	0.9719	0.9693	0.9947
distilbert-base-uncased	base+Focal	66 955 010	255	268.0	4.4	308.3	0.9733	0.9707	0.9956
distilbert-base-uncased	base+Quant	66 955 010	255	67.6	4.3	150.8	0.9719	0.9692	0.9948
distilbert-base-uncased	All	66 955 010	255	67.6	4.3	193.7	0.9737	0.9711	0.9956
distilbert-base-uncased	base+Focal+Quant	66 955 010	255	67.6	4.6	102.9	0.9737	0.9712	0.9958

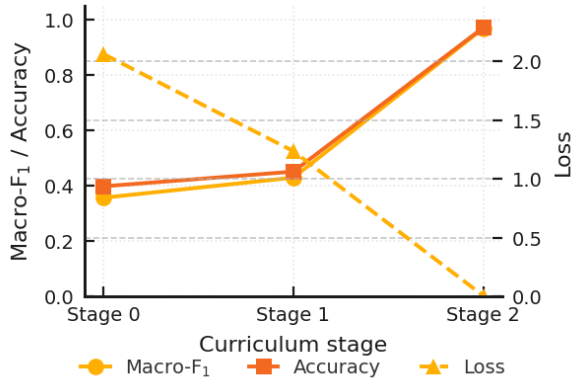

Fig. 6. Evolution of macro-F_1, accuracy, and loss across the three curriculum stages (test split). Stage 2 boosts performance sharply while driving loss near zero.

TABLE VII
FINAL PERFORMANCE AT THE CALIBRATED THRESHOLD $\tau = 0.60$. BOTH SPLITS ARE PERFECTLY DISJOINT FROM THE TRAINING DATA (§VI-B).

Split	Acc.	Macro-F_1	Prec.	Recall	AUROC
Dev	0.9693	0.9663	0.9684	0.9643	0.9927
Test	0.9698	0.9668	0.9684	0.9653	0.9930

reveals a slight preference for recall: only 1 068 false positives and 760 false negatives over more than 60 k samples.

d) Model efficiency.: We export the fine-tuned checkpoint to ONNX and apply INT8 dynamic quantization, reducing size from 360 MB to 67.6 MB (5.3×) with no measurable accuracy loss. Median CPU latency is 100 ms (P90 = 375 ms, P99 = 946 ms), including tokenization.

The near-identical dev and test scores confirm that curriculum learning yields a model that scales well from balanced news articles to noisy social posts while avoiding over-fitting. Quantisation further enables sub-second client-side inference, meeting the real-time constraints of browser-based fact-checking extensions.

VI. IMPLEMENTATION

To assess deployability, we report bundle size, startup time, memory footprint, and end-to-end latency on a mainstream laptop.

FakeZero is a modular, production-ready Chrome extension built using modern web technologies to detect fake news directly within users' browsers. Leveraging Manifest V3, JavaScript, Webpack, and Babel for broad compatibility, the extension provides real-time client-side inference without external data transfer, ensuring user privacy.

A. High-Level Overview

FakeZero injects a content script into supported web pages that processes visible content using a trained DistilBERT model. This script continuously monitors the social media feed and extracts new posts as the user scrolls. The inference pipeline leverages ONNX Runtime Web and the **xenova/transformers** library, optimizing model execution for in-browser environments. All key steps-tokenization, classification, and result feedback-are performed entirely on the client side, without sending any data to external servers. This ensures low latency, full user privacy, and a seamless browsing experience.

B. User Interface Integration

FakeZero employs a deliberately minimal user interface: a single inline warning element that appears *only* when model confidence exceeds the fake-news threshold. No global dashboard, counters, or user preferences are exposed. Instead, the system provides *contextual, point-of-consumption feedback* directly beneath the post that triggered the alert.

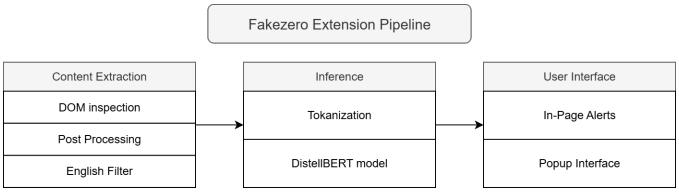

Fig. 7. High-level system architecture of the FakeZero Chrome extension. The pipeline is composed of three main components: (i) content extraction from social media posts via DOM inspection and filtering; (ii) in-browser inference using tokenization and a quantized DistilBERT model; and (iii) user-facing feedback through real-time alerts and popup summaries. Dependencies include **xenova/transformers** and ONNX Runtime Web.

When the content script flags a post, it inserts a small, non-destructive banner styled as a red alert ribbon (Fig. 8). The banner text is intentionally concise (e.g., "Warning: This content may contain false information."), and the visual design (icon + color) is tuned for legibility in both dark and light themes without altering the host platform's layout. The extension never modifies the original post text; the banner is appended in a sibling container.

To avoid repetitive alerts, FakeZero hashes each processed post and suppresses duplicate warnings within the session. All state is kept locally in memory; no user content is transmitted off the device.

C. Content Extraction

Upon loading a supported site such as Facebook or X (formerly Twitter), the extension dynamically injects a JavaScript content script into the page. Using specific DOM selectors-such as **div[data-ad-comet-preview="message"]** for Facebook and **article[data-testid="tweet"]** for X-the script identifies and extracts social media posts. It employs browser APIs, including *MutationObserver* to detect newly inserted posts and *IntersectionObserver* to limit processing to posts that enter the viewport.

To ensure full textual visibility, the script attempts to expand "See more" segments before content extraction. After that, the post text is cleaned, short or redundant messages are filtered out, and duplicate checks are enforced via hashing to avoid repeated computation.

D. Inference Pipeline

Extracted posts first pass through a lightweight regex filter for English text, then are tokenized via **xenova/transformers** and classified with the quantized DistilBERT model. The ONNX model (INT8) reduces from 360 MB to 67.6 MB with negligible accuracy loss. All assets—model, tokenizer, configs—reside within the extension bundle.

E. Privacy and Performance Considerations

All inference and data processing are performed entirely on the client side, ensuring complete user privacy. No user content leaves the local device, and no API calls are made to external servers. The quantized ONNX model, along with WebAssembly execution, reduces the memory and computational footprint, enabling fast and lightweight deployment even on mid-range devices.

The extension's manifest is written for Chrome's Manifest V3 standard, explicitly minimizing required permissions to reduce the attack surface. Only necessary access is granted for DOM inspection and content script injection.

a) Conclusion.: FakeZero's architecture demonstrates an efficient, privacy-first design for real-time misinformation detection in modern browsers. By combining state-of-the-art machine learning techniques with optimized web technologies, it offers a scalable, robust, and user-transparent solution for combatting fake news online.

F. Efficiency and Usability

FakeZero was tested on an Dell XPS 13 9300 (2020) laptop equipped with an Intel Core i7 10th generation CPU running Chrome 125. Upon first load, the extension streams the WebAssembly backend, tokenizer files, and ONNX graph in approximately 0.8 seconds. The final INT8-quantised DistilBERT model weighs 64.5 MB-nearly four times smaller than the original FP32 checkpoint-ensuring fast startup and reduced memory pressure.

At steady state, the browser heap remains at 96 MB used (205 MB allocated), with peak usage reaching 171 MB after five simultaneous post classifications. These values remain comfortably below the 4 GB limit imposed by most modern browsers, and no GPU memory is needed since all computations run natively in a SIMD-enabled WebAssembly sandbox. Language filtering is performed via a fast regular-expression gate, and complete end-to-end inference, including tokenisation, prediction, and UI update, averages under 8 ms per post.

These results demonstrate that FakeZero maintains real-time responsiveness and low memory overhead on mainstream hardware. Its client-only execution model avoids privacy trade-offs and ensures robust performance regardless of network connectivity, validating the feasibility of in-browser transformer inference for misinformation detection.

VII. Limitations & Future Work

Although **FakeZero** achieves near–state-of-the-art performance on casual tweets and Facebook posts-outperforming prior in-browser detectors by more than four F_1 points-several substantive extensions remain open.

First, the current checkpoint is trained primarily on English-language corpora heavily skewed toward U.S. political misinformation. While this bias contributes to the model's high accuracy on mainstream political content, it limits transferability to regions and cultural contexts, as well as long articles. Future releases will incorporate balanced, multi-domain datasets to mitigate this limitation.

Second, we plan to enrich output semantics by adding **sentiment analysis** and **emotion tagging**. Flagging

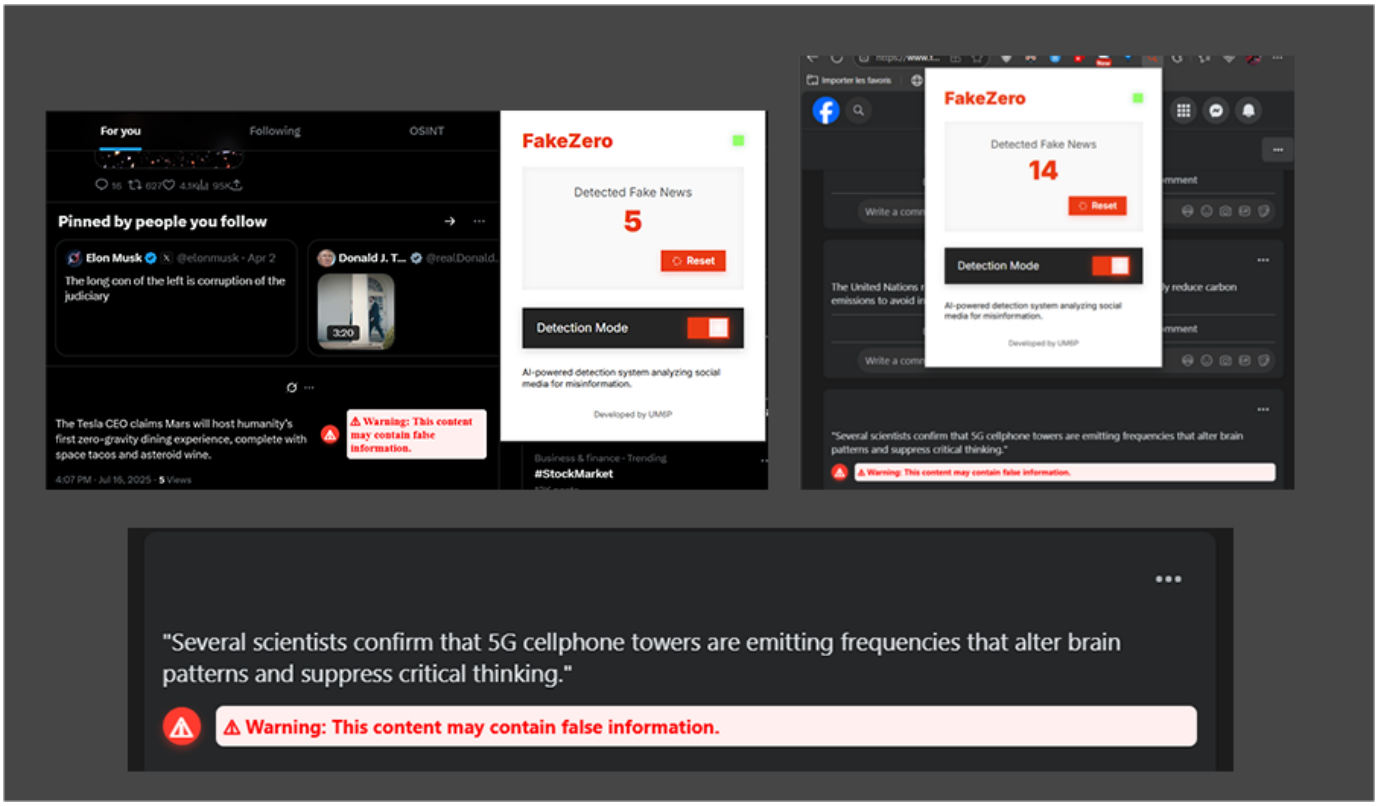

Fig. 8. FakeZero Extension User Interface. The extension alerts the user with a concise non-destructive banner styled as a red alert ribbon

emotional framing alongside factual credibility can help users recognise manipulation techniques such as outrage bait or fear-mongering.

Third, we will prototype a **retrieval-augmented generation (RAG)** module for emerging claims. A lightweight server will initially fetch up-to-date evidence; subsequent iterations will explore on-device caching or peer-to-peer retrieval to preserve privacy.

Fourth, we will refine label granularity by detecting specific **information-disorder tactics**-satire, false context, imposter content, or manipulated media-thereby aligning FakeZero with established fact-checking taxonomies and improving interpretability.

Finally, our roadmap includes support for **multilingual text**, **image / video analysis**, and expansion to additional platforms such as TikTok and WhatsApp Web, enabling comprehensive, privacy-preserving misinformation detection wherever users encounter content online.

VIII. Ethical considerations

No personal or user-specific data was accessed, collected, or exposed in the course of this research. All experiments were conducted using datasets composed of publicly available content or synthetic fake news examples created specifically for testing purposes. These test samples were kept private and were not shared or disseminated to users on any public platforms.

IX. Conclusion

FakeZero proves that real-time fake news detection can be performed entirely within the browser, without relying on cloud servers or exposing user data. By combining a distilled transformer model with WebAssembly execution, it achieves fast and accurate inference on consumer devices, ensuring both usability and strong privacy guarantees.

Unlike many existing tools, *FakeZero* runs fully client-side, meaning no content is sent over the network. This privacy-preserving design makes it compliant with modern data protection standards and suitable for deployment in sensitive settings.

The system offers a practical, scalable solution for misinformation detection on major platforms like Facebook and X, while remaining lightweight and transparent for users. It strikes a balance between technical performance and ethical deployment, and lays the groundwork for future extensions to multilingual, multimodal, and cross-platform detection in a privacy-conscious manner.

References

[1] S. B. Naeem, R. Bhatti, and A. Khan, "An exploration of how fake news is taking over social media and putting public health at risk," *Health Information & Libraries Journal*, vol. 38, no. 2, pp. 143–149, 2021.

[2] K. Sharma, F. Qian, H. Jiang, N. Ruchansky, M. Zhang, and Y. Liu, "Combating fake news: A survey on identification and mitigation techniques," *ACM transactions on intelligent systems and technology (TIST)*, vol. 10, no. 3, pp. 1–42, 2019.

[3] X. Zhou and R. Zafarani, "A survey of fake news: Fundamental theories, detection methods, and opportunities," *ACM Computing Surveys (CSUR)*, vol. 53, no. 5, pp. 1–40, 2020.

[4] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," *SIGKDD Explorations*, vol. 19, no. 1, pp. 22–36, 2017.

[5] The European Parliament and the Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," 2016.

[6] A. Jovanović and B. Ross, "Rumour detection in the wild: A browser extension for twitter," in *Proc. NLP Open Source Software Workshop 2023*. Singapore: ACL, 2023, pp. 130–140.

[7] B. Botnevik, E. Sakariassen, and V. Setty, "BRENDA: Browser extension for fake news detection," in *Proc. 43rd ACM SIGIR Conf. on Research and Development in Information Retrieval*. Virtual Event: ACM, 2020, pp. 2117–2120.

[8] F. Jain, L. Chen, and S. Wilson, "TrustNet: Browser extension for misinformation assessment," in *Proc. CHI '23*. Hamburg, Germany: ACM, 2023, pp. 1–14.

[9] K. Shu, D. Mahudeswaran, and H. Liu, "Fakenewstracker: a tool for fake news collection, detection, and visualization," *Computational and Mathematical Organization Theory*, vol. 25, no. 1, pp. 60–71, 2019.

[10] K. Thilakarathna and S. Seneviratne, "Veritas: A browser extension for twitter rumour detection," in *Proc. 12th ACM Conf. on Web Science*. Southampton, UK: ACM, 2020, pp. 319–325.

[11] D. Paschalides, A. Kornilakis, C. Christodoulou, R. Andreou, G. Pallis, M. D. Dikaiakos, and E. P. Markatos, "Check-it: A plugin for detecting and reducing the spread of fake news and misinformation on the web," *CoRR*, vol. abs/1905.04260, 2019. [Online]. Available: https://arxiv.org/abs/1905.04260

[12] W. Paka, R. Bansal, A. Kaushik, S. Sengupta, and T. Chakraborty, "Cross-sean: A cross-stitch semi-supervised neural attention model for COVID-19 fake news detection," *Applied Soft Computing*, vol. 107, p. 107393, 2021.

[13] P. Kydd and L. Shepherd, "Rumour detection in the wild: A browser extension for twitter," in *Proc. NLP Open Source Software Workshop 2023*. Singapore: ACL, 2023, pp. 107–113.

[14] D. Warman, M. Cole, and Y. Li, "Real-time fake news detection with explainable bert in a chrome extension," *Journal of Web Engineering*, vol. 22, no. 4, pp. 811–830, 2023.

[15] L. Hasimi and A. Poniszewska-Maranda, "Browser extension for detection of fake news and disinformation," in *Information Systems*, ser. Lecture Notes in Business Information Processing. Springer, 2023, pp. 209–220.

[16] J. Wang, M. Luo, and R. Patel, "Efficient transformers for edge deployment," *Computers & Electrical Engineering*, vol. 116, p. 108267, 2024.

[17] W. Chen, R. Zhang, and L. Zhao, "Hybrid BERT–GPT architecture for misinformation detection," *Journal of Information Security and Applications*, vol. 73, p. 103625, 2024.

[18] J. Yin, M. Gao, and K. Shu, "Break: Rationale-augmented fake news detection with fully connected semantics graph," in *Proc. The Web Conf. 2025 (WWW '25)*. Montréal, Canada: ACM, 2025, pp. 2114–2123.

[19] M. Gao, Y. Huang, and K. Shu, "Birds of a feather: Multimodal retrieval for fake news detection," in *Proc. IEEE ICDE 2025*. Berlin, Germany: IEEE, 2025, pp. 1022–1033.

[20] W. Li, Q. Xu, and Y. Zhang, "Multi-element evidence retrieval for robust fake news detection," in *Proc. ACM SIGKDD 2025*. San Francisco, USA: ACM, 2025, pp. 1456–1465.

[21] R. Gupta, P. Singh, and A. Varma, "Deep learning for fake news detection: A comprehensive survey," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–38, 2024.

[22] X. Zhou and Y. Liu, "An overview of fake news detection: From a new perspective," *Information Sciences*, vol. 676, pp. 1–28, 2024.

[23] S. Dadkhah, X. Zhang, A. G. Weismann, A. Firouzi, and A. A. Ghorbani, "TruthSeeker: The Largest Social Media Ground-Truth Dataset for Real/Fake Content," *IEEE Trans. Computational Social Systems*, 2023.

[24] H. Ahmed, I. Traore, and S. Saad, "ISOT Fake News Dataset," 2018, university of Victoria. [Online]. Available: https://onlineacademiccommunity.uvic.ca/isot/

[25] A. Zubiaga, M. Liakata, R. Procter, G. W. S. Hoi, and P. Tolmie, "Analysing how people orient to and spread rumours in social media by looking at conversational threads," in *Proc. ACM Web Science*, 2016, pp. 16–26.

[26] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "FakeNewsNet: A data repository with news content, social context, and spatio-temporal information for studying fake news on social media," *Big Data*, vol. 8, no. 3, pp. 171–188, 2020, the "Social" split contains 433 k labelled tweets (200 k fake) linked to PolitiFact and GossipCop articles.

[27] W. Y. Wang, ""Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection," in *Proceedings of ACL*, 2017.

[28] M. Taşcı and Y. Bayrakdar Yılmaz, "Developing a web browser extension to prevent the spread of fake news," *International Journal of Engineering and Computer Science*, vol. 13, no. 10, pp. 26 576–26 588, 2024. [Online]. Available: https://www.ijecs.in/index.php/ijecs/article/view/4921

[29] X. Yang, Y. Wang, X. Zhang, S. Wang, and K.-Y. Lam, "A macro- and micro-hierarchical transfer framework for cross-domain fake news detection," in *Proceedings of the Web Conference (WWW)*, 2025.

[30] D. Berrios, F. Hoffmann, and J. Kreutzer, "Llms have a mosaic memory: Accounting for fuzzy duplicates in deduplication," 2025. [Online]. Available: https://arxiv.org/abs/2405.15523

[31] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th International Conference on Machine Learning (ICML)*. Montreal, Canada: ACM, 2009, pp. 41–48.

[32] G. Karthik, K. S. F. Ahmad, and A. P. Sathe, "Hybrid optimisation-driven fake news detection using reinforced transformer models," *Scientific Reports*, vol. 15, p. 14782, 2025.

[33] F. Pedregosa, G. Varoquaux, and A. t. Gramfort, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] X. Chen, F. Barbieri, and J. Camacho-Collados, "Tweeteval 2.0: Strong baselines and unified benchmarks for social-media nlp," in *Findings of EMNLP*, 2024.

[35] S. Nakatani, "Language detection library for java," GitHub repository, 2010, accessed 31 Jul 2025. [Online]. Available: https://github.com/shuyo/language-detection

[36] I. Beltagy, A. Cohan, and K. Lo, "SciBERT: A pretrained language model for scientific text," in *Proceedings of EMNLP-IJCNLP*. Hong Kong, China: ACL, 2019, pp. 3615–3620.

[37] Y. Kara, F. Durmus, and D. Yuret, "FAKER: A benchmark dataset for fake news detection," *PeerJ Computer Science*, vol. 6, p. e279, 2020.

[38] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[39] S. Raza, D. Paulen-Patterson, and C. Ding, "Fake news detection: Comparative evaluation of BERT-like models and large language models with generative AI-annotated data," *arXiv*, 2024.

[40] M. Chen, S. Wang, and R. Zhang, "Edgebert: Efficient transformer inference on commodity edge devices," in *Proceedings of AAAI*, 2024.

[41] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv*, 2019.

[42] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, and Q. Liu, "Tinybert: Distilling bert for natural language understanding," in *Proc. EMNLP 2020*, 2020, pp. 4163–4174.