Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 6, December 2026, pp. 5514~5531
ISSN: 2302-9285, DOI: 10.11591/eei.v15i6.12782

A leakage-audited comparison of neural architectures for fake news detection: a cross-dataset evaluation

Mohammad Alkhazaleh[1], Mahmoud Baklizi[2], Musab Alzghoul[1], Loui Al Sardy[3]
[1]Department of Computer Science, Faculty of Information Technology, Isra University, Amman, Jordan
[2]Department of Computer Science, Faculty of Information Technology, University of Petra, Amman, Jordan
[3]Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany

Article Info

Article history:

Received Mar 11, 2026
Revised Sep 11, 2026
Accepted Oct 4, 2026

Keywords:

Cross-dataset evaluation
Data leakage
Fake news detection
Negative results
Reproducibility
Text classification

ABSTRACT

Reported accuracies for fake news detection span roughly 60% to 99%, too wide a spread to be explained by architecture. We trace much of it to leakage. Auditing a corpus built from word embedding over linguistic features for fake (WELFake), LIAR, FakeNewsNet, and a published COVID-19 dataset, we find five publisher artifacts, chiefly wire-service datelines and image-credit lines, in 31,701 of the 96,626 documents collected (32.8%); they predict the label at about 99.7% purity by string matching alone, and fake articles are exact duplicates fifteen times more often than real ones. They survive preprocessing and are not removed by source-aware splitting, because they sit inside the article. After mitigating both channels and partitioning 87,408 deduplicated articles by source, we compare seven architectures under three seeds each and detect no robust separation: the spread between their macro-averaged accuracies (2.16 points) is smaller than the 2.63 expected from seed variation alone, an absence of detected separation rather than evidence of equivalence. A term frequency-inverse document frequency (TF-IDF) logistic regression (LR) fitted in 77 seconds reaches 88.05% against 89.30% for the best neural model. WELFake supplies 69.6 of those 89.3 points, and on LIAR every run gives an area under the curve (AUC) of 0.57-0.59. Predictions, splits and an audit tool are released.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Mohammad Alkhazaleh
Department of Computer Science, Faculty of Information Technology, Isra University
Amman, Jordan
Email: m.alkhazaleh@iu.edu.jo

1. INTRODUCTION

Digital platforms have made it possible for any citizen to publish to a global audience in real time, and the same openness has enabled an unprecedented diffusion of fabricated content, a phenomenon information scientists describe as an infodemic running in parallel with legitimate information streams [1]. Large-scale analysis of rumour cascades has shown that false stories travel farther, faster, and more broadly than true ones: falsehoods reached 1,500 people roughly six times faster than accurate reports, and false political news spread more widely than any other category [2]. Manual fact-checking does not scale to this volume, which has motivated a large literature on automated detection [3]–[5].

That literature has a measurement problem. Reported accuracies on public benchmarks span roughly 60% to 99%. Studies on word embedding over linguistic features for fake (WELFake) routinely report figures above 97% [6], [7]; studies on LIAR rarely exceed 70% [8], [9]; and the same architecture can produce both, as FakeStack does at 98.25% and 75.58% respectively [6]. A spread of nearly forty points is not plausibly explained by architectural choices, and it makes the comparative claims in this field difficult to interpret. If

two papers report 92% and 97% on overlapping data, a reader currently has no way to tell whether the difference reflects a better model, an easier corpus, or an evaluation artefact.

This paper takes the measurement problem as its subject. We identify three leakage channels that inflate reported performance, quantify each on a combined four-corpus dataset, and then ask what remains of architectural differences once they are mitigated. The three channels are distinct and require different remedies, and none of the three is eliminated entirely: subsection 3.2 states the limits of each. Publisher artifacts, such as wire-service datelines and image-credit lines, live inside the article text and survive conventional preprocessing. Duplicate articles cross the partition boundary and, in the corpora we examine, do so far more often for one class than the other. Outlet and speaker fingerprints allow a model to identify the source rather than the content. Only the third is addressed by the source-aware splitting that has begun to appear in this literature, and it is, on our measurements, the least consequential of the three.

Our results are negative with respect to architecture and positive with respect to method. Across seven architectures trained under three seeds each we detected no robust separation under this protocol; the hybrid convolutional neural network-bidirectional long short-term memory (CNN-BiLSTM) model with additive attention, which an earlier version of this work proposed as a contribution, ranks sixth of seven. We report this in full, including the retraction of our own claim, because a negative result obtained under a controlled protocol is more informative for this field than another architecture reporting a gain that will not reproduce.

The specific contributions of this work are as follows:

- A quantitative audit of publisher-artifact leakage. Five patterns appear in 31,701 of the 96,626 documents collected (32.8%; the audit precedes deduplication) and separate the classes at approximately 99.7% purity by string matching alone. A simple calculation puts their effect at four to seven points of any accuracy obtained on WELFake without scrubbing. The audit script is released separately and can be run on other corpora.
- Evidence that duplication in WELFake is heavily skewed by class. Exact duplicates make up 22.3% of its fake articles but only 1.4% of its real ones, and under a random split this roughly fifteen-fold difference shows up mostly as inflated recall.
- A source-aware split in two tiers. Where a corpus provides an outlet or speaker field we group by it, and elsewhere we block near-duplicates with MinHash; how well this works, and where it falls short, is measured and reported.
- A comparison of seven architectures, each trained with three seeds under identical settings. The differences between architectures turn out to be smaller than those caused by changing the seed, and which model comes first depends on the metric used.
- A linear reference point: a term frequency-inverse document frequency (TF-IDF) logistic regression (LR) evaluated on the identical partition, which the best neural architecture exceeds by 1.25 points and the previously proposed hybrid by 0.64, at a small fraction of the training cost.
- A set of reporting rules derived from these measurements, together with the release of per-article predictions and split assignments, so that the results tables in this paper can be regenerated and checked without access to a graphics processing unit (GPU).

The remainder of this paper is organised as follows: section 2 reviews the related literature. Section 3 sets out the evaluation protocol and the architectures compared. Section 4 describes the corpus, the leakage audit and the preprocessing pipeline. Section 5 gives the experimental setup. Results are presented in section 6, limitations and possible extensions in section 7, and section 8 closes the paper.

2. RELATED WORK

Below we trace how computational fake news detection has developed. Because our contribution concerns evaluation, we pay particular attention to how each strand of work tests its models.

2.1. Foundational approaches and benchmark datasets

Early detectors were built on linguistic analysis, using features such as writing style, sentiment polarity and indicators of source credibility [10]. Among the first benchmarks was LIAR [8], which contains 12,836 political statements annotated manually. Social context was added by FakeNewsNet [11] and WELFake [12] combined 72,134 articles drawn from four sources. A large share of later work relies on these three corpora, and two features of how they were constructed are important for the analysis in this paper. WELFake was assembled from sources that overlap with one another, including a wire service, whereas LIAR consists of short claims with an average length below twenty tokens. Studies reporting results on these corpora rarely discuss either feature.

2.2. Deep learning for fake news detection

2.2.1. Convolutional neural network

CNNs extract local n-gram patterns effectively. Yang *et al.* [13] found that filters of several widths respond to phrasing typical of fabricated news, and Ivancova *et al.* [14] trained CNN and long short-term memory (LSTM) models on Slovak political news. CNNs are often criticised for missing long-range dependencies, which, it is argued, should hurt them on long documents. In the form in which it is usually made, this claim is not borne out by our measurements in section 6.

2.2.2. Long short-term memory

LSTMs handle sequences by means of gated recurrence [15]. Rajalaxmi *et al.* [9] obtained 99.65% on ISOT and, after a hyperparameter search, 45.23% on LIAR, so the same model differs by fifty-four points between two corpora. Nithya and Sahayadhas [16] combined evolutionary feature selection with an LSTM; Dutta *et al.* [17] reported 75.34% on a COVID-19 corpus; Jaybhaye *et al.* [18] reported 94% on a Kaggle corpus. The pattern across these studies is not that some architectures are better, but that the corpus determines the result far more than the model does.

2.2.3. Bidirectional long short-term memory approaches

Bidirectional processing lets a model use both past and future context [19]. Islam *et al.* [20] reached 84% with a Bi-LSTM on 2,977 articles. Truică *et al.* [21] reported that cross-lingual BiLSTM models fell to F1 around 0.28 against 0.53 monolingual, and Vo and Do [22] observed the same for Vietnamese. These cross-lingual collapses are the closest the literature comes to the kind of cross-dataset performance collapse reported in subsection 6.4.

2.3. Hybrid and ensemble architectures

Hybrid CNN-LSTM structures have been widely investigated [6], [23]. Keya *et al.* [6] proposed FakeStack, a tri-layer bidirectional encoder representations from transformers (BERT)-CNN-LSTM model reporting 99.74% on one English dataset, 98.25% on WELFake and 75.58% on LIAR. Akilandeswari *et al.* [23] combined CNN-LSTM with particle swarm optimisation for cybersecurity, and Huang *et al.* [24] applied a CNN-LSTM-attention hybrid to wind-speed forecasting. What these studies share, and what motivates the present work, is that the hybrid is compared against baselines trained by the same authors under the same protocol, with a single run and no variance reported. Subsection 6.4 shows that a single run is not sufficient to distinguish these architectures.

2.4. Attention mechanisms and interpretability

Attention is used both for performance and for interpretability [25], [26]. Aloshban [27] reported 75% with a self-attention model over claims and supporting articles. Mohapatra *et al.* [7] combined BiLSTM with self-attention and reported 98.65% on a single dataset. Tseng *et al.* [25] introduced KAHAN, a knowledge-aware hierarchical attention network reaching 92.4% precision on FakeNewsNet. We include both an additive and a scaled dot-product attention variant in our comparison, and find that the choice between them changes the result by about a point in the direction opposite to the one usually assumed.

2.5. Transformer-based models

Pre-trained transformers such as BERT [28] and robustly optimized BERT approach (RoBERTa) [29] have advanced the state of the art across natural language processing (NLP), and Jing *et al.* [30] applied a multimodal transformer to this task. Almost all of the transformer results in this area, though, come from random splits within a single dataset, so any comparison with lighter models carries over whatever leakage that split allows. Tashtoush *et al.* [31] built a framework aimed specifically at COVID-19 content, which shows how much performance depends on the domain. Transformers are not part of our experiments, a limitation taken up again in subsection 7.1. The leakage effects measured below are, however, properties of the data, so they bear on transformer evaluations just as much.

2.6. Summary of research gaps

The gaps we see in earlier work, and our response to each, are listed in Table 1. The first three rows concern leakage that enters through the corpus rather than through the model, which is why it persists whichever architecture is trained on it. The last three concern what is reported, and together they determine whether a reader can tell one published result from another.

Table 1. Summary of research gaps and contributions

Research gap	Prior work	This study
Leakage inside the text not measured	Publisher artifacts stay in the text	Five artifacts quantified: 32.8% of the corpus at 99.7% purity
Duplicate leakage not measured	Deduplication seldom mentioned	9,133 exact duplicates dropped; 15× gap between classes measured
Source leakage	Random splits	Source-aware split in two tiers, with its effect measured
Single-run comparisons	Single run, variance not given	Three seeds for each model; differences set against seed noise
Pooled metrics on unbalanced corpora	A single headline number	Pooled, macro and per-dataset results, each with its majority baseline
Results cannot be checked	Only summary tables	Predictions per article and split assignments made public

3. PROPOSED METHOD

3.1. Problem formulation and scope

For each input document x, the model has to predict a label y in {0, 1}, where a value of 0 marks legitimate content and 1 marks fabricated content. Formally:

$$f(x;\theta) \rightarrow y \quad (1)$$

where f denotes a neural classifier with parameters theta, trained by minimising binary cross-entropy. We are less interested in which f gives the best score than in how much of the measured gap between candidate f models is still there after the evaluation has been cleaned. For this reason the protocol itself is what we study, and the architectures serve as the means of probing it.

3.2. Evaluation protocol

The protocol is made up of three parts, and each one targets a different leakage channel. We describe them below in the same order in which they are applied to the corpus.

The first channel is closed by publisher-artifact scrubbing. Before tokenisation, we strip out any text matched by five regular expressions that target wire-service datelines, image-credit lines and embedded social-media URLs. The coverage and class purity of each pattern are given in subsection 4.2. We chose a blunt removal on purpose. Some genuine stylistic signal is lost along with the artifacts, and there is no clean way to tell the two apart, so the results obtained after scrubbing should be taken as a conservative estimate.

The second channel is closed by exact-duplicate removal. Duplicates are detected on the text that is left after scrubbing. Subsection 4.2 gives the counts and, more importantly, shows how unevenly they fall across the two classes.

The third channel is addressed by two-tier source-aware splitting. Each article is given a single group identifier, and a group is never split across partitions. Tier 1 uses an explicit source where the corpus carries one: the registered domain of the article URL for FakeNewsNet, and the speaker field for LIAR, which has no outlet but whose speakers are the entities whose stylistic fingerprints a model could memorise. For corpora that have no source field (WELFake and COVID-19), Tier 2 takes over. Here the group is a near-duplicate cluster found with MinHash and locality-sensitive hashing on 5-gram shingles; we use 64 permutations arranged as 16 bands of 4 rows, which corresponds to a Jaccard threshold of roughly 0.5, and the clusters are closed transitively. This blocks syndicated copies and templated output of a single outlet.

We note two honest limitations of the third component. Temporal ordering is not applied because none of the four corpora carries a usable per-article timestamp. And Tier 2 is weak precisely where the corpus is largest: WELFake yields 61,296 groups over 63,327 articles, so after exact duplicates are removed the split approximates a random one on 72% of the data. In short, source-aware splitting deals with source leakage only when the source can be identified. For WELFake, it is mainly the artifact scrubbing that keeps the evaluation honest.

3.3. Model architectures

3.3.1. Convolutional baseline

The CNN extracts multi-scale n-gram features through three parallel convolutional branches (kernel sizes 3, 4, and 5; 128 filters each) over 300-dimensional global vectors for word representation (GloVe) embeddings [32]. Each branch is globally max-pooled, and the resulting 384-dimensional vector passes through dropout (p=0.5), a dense layer (128 units, rectified linear unit (ReLU)) and a sigmoid unit. Figure 1 shows the architecture.

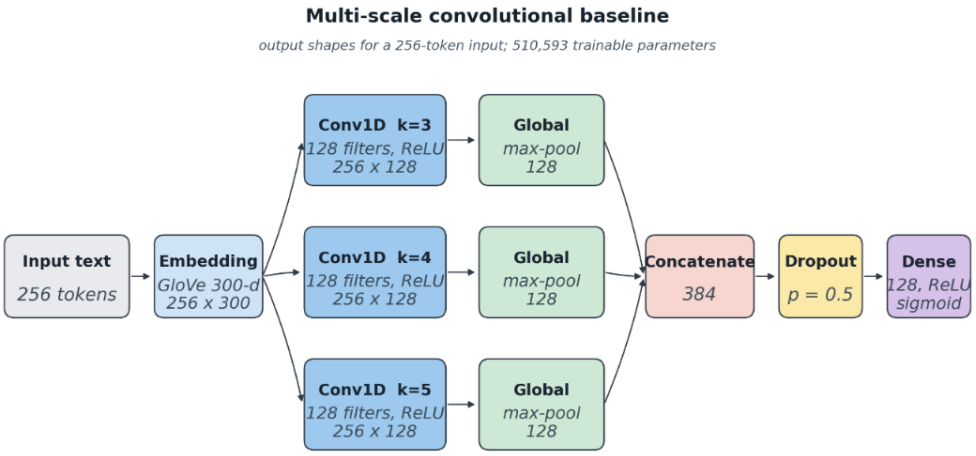


Figure 1. Multi-scale convolutional baseline, with output shapes given for a 256-token input

3.3.2. Recurrent baseline

The BiLSTM feeds the embedded sequence through a bidirectional recurrent layer with 128 units per direction, followed by dropout (p=0.3), a dense layer (64 units, ReLU) and a sigmoid unit. Figure 2 shows the architecture.

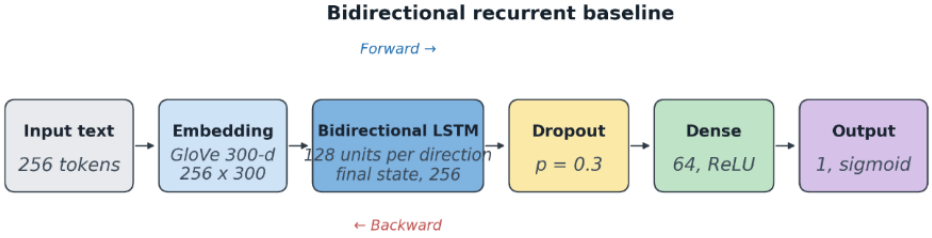


Figure 2. Bidirectional recurrent baseline

3.3.3. Hybrid variants

Four hybrid variants are compared. In all of them the convolutional branches use 'same' padding and are concatenated along the channel axis with the temporal axis preserved, so the recurrent layer receives a sequence of local feature vectors rather than a pooled summary. The variants are: CNN with BiLSTM and no attention; CNN with BiLSTM and scaled dot-product attention; CNN with a unidirectional LSTM and additive attention; and CNN with BiLSTM and additive attention, which is the configuration an earlier version of this work proposed. Additive attention over the fused hidden states $h_1 \ldots h_T$ is computed as (2)-(4):

$$e_i = W_a \tanh(W_u h_i + b_u) + b_a \quad (2)$$

$$\alpha_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)} \quad (3)$$

$$c = \sum_i \alpha_i h_i \quad (4)$$

where W_u and W_a are learnable projections, b_u and b_a are biases, the coefficients α_i are the normalised weights, and c is the context vector, which passes to a dense layer (128 units, ReLU), dropout (p=0.5) and a sigmoid unit. The dot-product variant replaces this with a single learned query scaled by the square root of the hidden dimension. Attention is masked so that padded positions receive no weight. Table 2 gives the layer-by-layer specification of the additive variant; the model has 1,085,570 trainable parameters and an 18,000,000-parameter frozen embedding matrix, 19,085,570 in total. Figure 3 shows the architecture.

Table 2. Layer specification of the CNN-BiLSTM model with additive attention

Layer	Configuration	Output shape	Parameters
Input	Integer token ids	256	0
Embedding	GloVe 840B, 300-d, frozen	256×300	18,000,000 (frozen)
Conv1D x3	Kernels 3/4/5, 128 filters, ReLU, and same padding	256×128 each	461,184
Concatenate	Channel axis and time preserved	256×384	0
Bidirectional LSTM	128 units per direction and sequences returned	256×256	525,312
Additive attention	Wu 256×256, Wa 256×1, and masked softmax	256 weights	66,049
Context vector	Alpha-weighted sum	256	0
Dense+dropout	128 units, ReLU, L2=0.001, and p=0.5	128	32,896
Output	1 unit and sigmoid	1	129
Total	Excluding the frozen embedding matrix	-	1,085,570 trainable

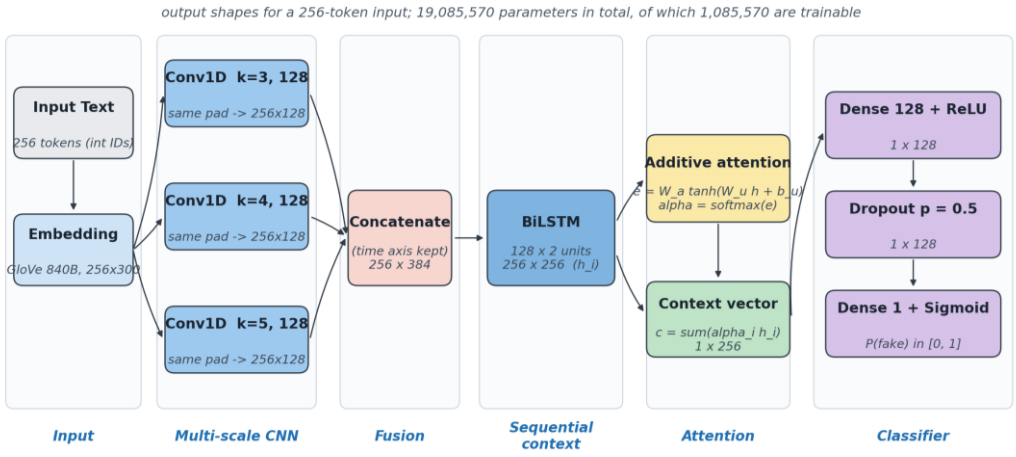


Figure 3. CNN-BiLSTM architecture with additive attention

3.4. End-to-end pipeline

The protocol runs in eight steps, shown in Figure 4. The four corpora are collected (96,626 documents); publisher artifacts are audited and scrubbed; exact duplicates are removed, leaving 87,408 articles; every article is assigned a source group under the two-tier scheme; the corpus is partitioned 70/15/15 by group with the no-crossing property asserted rather than assumed; each architecture is trained at three seeds; per-article predictions are written to disk; and the results tables in section 6 are generated from those files rather than transcribed from training logs. The last two steps are part of the protocol rather than implementation detail: tables generated from released predictions can be checked by a reader, whereas tables transcribed from training logs cannot.

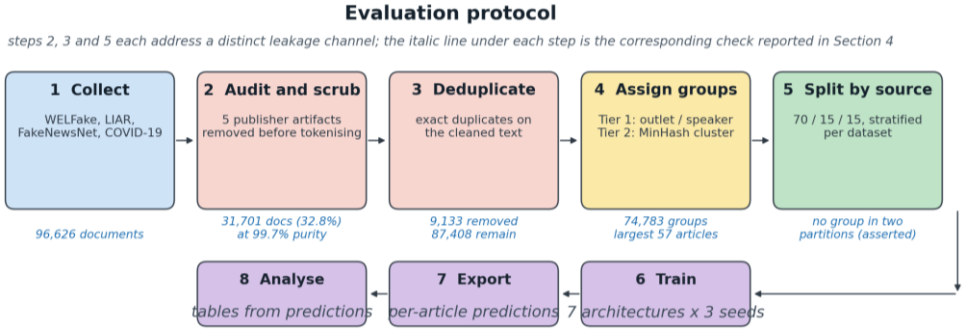


Figure 4. The evaluation protocol, in which steps 2, 3, and 5 address the three leakage channels and the figure below each step is the corresponding check reported in sections 4 and 5

3.5. Regularisation

Four strategies are applied identically to every architecture: dropout (p=0.3-0.5) after the embedding and dense layers; L2 weight regularization with lambda=0.001; early stopping with patience of eight epochs; and learning-rate reduction by a factor of 0.5 on plateau with patience of three.

3.6. Evaluation metrics and reporting rules

We report accuracy, precision, recall, F1-score, specificity and area under the curve-receiver operating characteristic (AUC-ROC) [33], [34], with 95% bootstrap confidence intervals over 1,000 stratified resamples [35]. Four reporting rules follow from the measurements in section 6 and are applied throughout. First, every headline figure is accompanied by the macro average over the four datasets and by the per-dataset breakdown, because a pooled metric on an unbalanced corpus reports its composition. Second, each per-dataset accuracy is shown next to the majority baseline for that dataset, because on its own an accuracy figure can hide a model that has learned nothing on that slice. Third, alongside bootstrap intervals we give the standard deviation over three random seeds. Bootstrap intervals capture the sampling variance of the test set but say nothing about variance from training, and in our experiments the latter is larger. Fourth, effect sizes for differences of proportions are given as Cohen's h rather than Cohen's d, and paired comparisons use McNemar's test [33] with Bonferroni correction alongside Welch t-tests on the seed-level values.

Reuse; both tools are separable from this study. src/leakage_audit.py takes any CSV with a text and a label column and reports, per pattern, the documents matched, their class purity and the accuracy a string matcher would reach; src/analyze.py rebuilds the results tables from released prediction files. Neither needs a GPU, our corpora, our architectures or our splits, so the audit can be run on a new corpus before any accuracy is reported for it.

4. DATASETS AND PREPROCESSING

4.1. Corpus

The corpus combines four public sources spanning the political, general-news and health domains [8]–[12], [31]. Table 3 gives its composition after cleaning and deduplication; the rows sum to the combined row, and the class balance and dataset shares quoted elsewhere in this paper are derived from it. Figure 5 presents the corpus and its publisher artifacts in two panels; Figure 5(a) shows this composition graphically.

Table 3. Composition of the corpus after cleaning and removal of exact duplicates

Dataset	Articles	Real	Fake	Mean tokens	Domain
WELFake [12]	63,327	34,522	28,805	562	General news
LIAR [8]	12,751	7,121	5,630	18	Political
COVID-19 [31]	10,379	5,507	4,872	27	Health
FakeNewsNet [11]	951	537	414	10	Political
Combined	87,408	47,687	39,721	413	Mixed

Three features of this corpus affect all of the later results. WELFake alone accounts for 72.5% of the articles, so any pooled metric predominantly reports general-news performance. Average length varies from 10 tokens (FakeNewsNet) to 562 (WELFake), a ratio of fifty-six, and a single input window has to serve headline-length claims as well as complete articles. The class distribution is close to balanced overall (54.6% real, 45.4% fake), which means accuracy is a meaningful summary statistic and no result below can be attributed to majority-class bias. Two deviations from the corpus described in an earlier version of this work should be noted. For COVID-19 we use the published Constraint@AAAI2021 dataset [31] instead of a curated set of articles. Its items are social-media posts, hence the short average length. FakeNewsNet is used at title level only, because the public release distributes identifiers and URLs rather than article bodies and many links are no longer live; its 143-article test slice is too small to support a per-dataset claim and is always reported with its seed standard deviation.

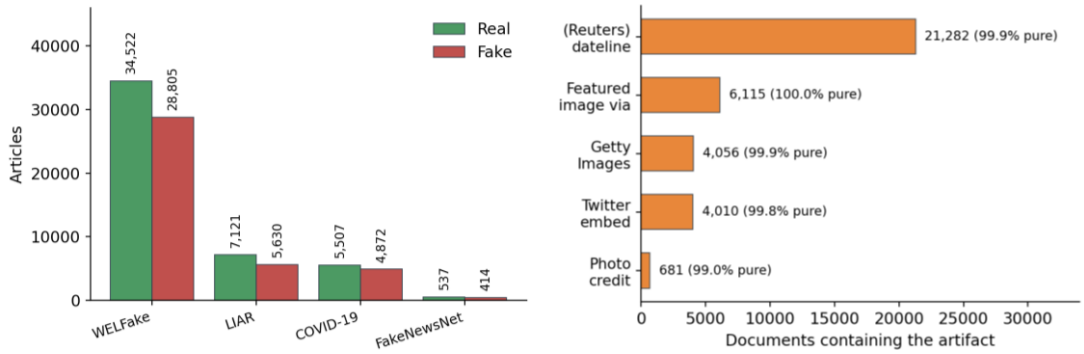
4.2. Preprocessing and leakage audit

The pipeline comprises eight steps applied identically to all four corpora, with the same fitted tokenizer, so that no preprocessing difference confounds the comparison: HTML removal; URL and e-mail stripping; publisher-artifact scrubbing; punctuation normalization; lower-casing; word-level tokenization; stop-word removal using the NLTK list; and padding or truncation to 256 tokens. The tokenizer is fitted on the training partition only.

The third step deserves its own treatment, because it is the channel this paper argues is both the largest and the least discussed. WELFake inherits a Reuters-sourced portion in which the wire dateline survives every other step listed above: lower-casing, punctuation removal and stop-word filtering all leave the token intact. Table 4 reports, for each pattern, how many documents contain it and what share of those documents belong to a single class, which we call its purity. Figure 5(b) shows the same figures.

Table 4. Publisher artifacts measured before scrubbing, where purity is the share of matching documents belonging to a single class

Artifact pattern	Documents	Share of corpus (%)	Purity (%)
Wire-service dateline	21,282	22.03	99.9
Image-credit phrase	6,115	6.33	100.0
Stock-photo agency name	4,056	4.20	99.9
Embedded social-media URL	4,010	4.15	99.8
Photo-credit line	681	0.70	99.0
Any of the above	31,701	32.80	~99.7



(a) (b)

Figure 5. The assembled corpus, showing; (a) its composition by dataset and class and (b) the publisher artifacts, their coverage, and their class purity

The magnitudes are not marginal. The audit is run before deduplication, on the 96,626 documents as collected. The Reuters dateline appears in 21,282 of them, 22.03%, and 99.9% are real. The phrase introducing an image credit appears in 6,115 documents, and 100.0% of them are fake, with no exceptions. Taken together the five patterns cover 31,701 documents, 32.8% of the collected corpus, separable at approximately 99.7% purity by string matching with no learning at all.

The consequence for reported accuracy follows arithmetically. If a model scores 99.7% on that 32.8% and X on the remainder, its headline figure is 0.328×0.997+0.672×X. Reaching 92% therefore requires only 88.5% genuine skill on the artifact-free portion, and 88% true skill produces a 91.8% headline. By this estimate, four to seven points of any accuracy reported on unscrubbed WELFake come from the five patterns alone. Splitting by source cannot fix that, because the artifact is part of the article text and is carried into the training, validation or test partition along with it.

Next came exact-duplicate removal, run after scrubbing. In total 9,133 documents (9.5%) were dropped at this stage and 87,408 articles remained; these are used in every experiment below. The dropped documents were mostly fake. Inside WELFake the duplicate rate was 22.3% for fake articles (8,252 of 37,057) but only 1.4% for real ones (506 of 35,028), about fifteen times lower. Under a random split, copies of fake articles end up on both sides of the split far more often than copies of real ones. Recall gets the biggest boost, and recall is the metric papers in this field most often lead with. WELFake's origin explains the pattern: four overlapping sources were merged into one. We verified that scrubbing did not create these duplicates: recomputing the count without scrubbing gives 9,132, a difference of one document, so the duplication is a property of the corpus rather than an artefact of our pipeline.

We see the imbalance as a data-curation failure at least as much as a modelling one. A corpus assembled from overlapping sources should be deduplicated before the classes are balanced, not after, and the duplicate rate should be reported per class rather than in aggregate: an overall figure of 9.5% conceals the 22.3% against 1.4% split that does the damage. We suggest that future releases report both, and publish the grouping key used, so that a reader can tell whether a reported split is source-aware in fact and not only in name.

Documents shorter than three tokens after cleaning were discarded. The 256-token window covers the whole of LIAR, COVID-19 and FakeNewsNet, and roughly 46% of the mean WELFake article; subsection 7.1 records this as a limitation.

4.3. Data and code availability

Per-article predictions for every architecture and seed, the split assignments, the preprocessing and analysis code, and the artifact-audit tool are deposited at https://github.com/moh1984/fake-news-leakage-audit.

The prediction files carry the article identifier, dataset, source group, true label, and predicted probability, allowing the reported results to be regenerated in seconds without a GPU and enabling independent computation of any additional statistics. The remaining corpus-based classifier results can also be regenerated using the released notebook. The source corpora are available from their original providers [8]–[12], [31] and are not redistributed.

5. EXPERIMENTAL SETUP

5.1. Partitioning

Groups are assigned to partitions greedily, largest first, with each group placed in whichever partition is furthest below its target share, breaking near-ties in favor of the partition whose class balance the group would improve. We run this assignment separately for each dataset, which keeps every corpus represented in proportion across the three partitions.

This produces 61,185 training articles (70.00%), 13,112 validation articles (15.00%) and 13,111 test articles (14.99%), drawn from 74,783 distinct source groups. Fake articles make up 45.44%, 45.45% and 45.44% of the three partitions, and each dataset is split 15/70/15 with a deviation of at most 0.1 points. The code checks explicitly that no group is shared between partitions, and because the assignment of every article is released, anyone can confirm this. No single Tier-2 cluster dominates the split; the biggest has 57 articles, or 0.08% of the corpus.

5.2. Training configuration

The configuration is given in Table 5. Any setting that the architectures have in common was kept the same for all of them, so that differences in the results can be put down to the architecture and not to more or less careful tuning. For early stopping we track the validation area under the curve (AUC) instead of the validation loss. The hybrid models quickly become overconfident; this pushes cross-entropy up even though their ranking of examples does not get worse, and a loss-based rule would therefore stop them because of poor calibration, not poor discrimination. According to the released training logs the rule was applied in the same way everywhere: in each of the 21 runs, training ended exactly eight epochs after the best epoch. The mean number of epochs needed to reach the best epoch ranges from 4.3 (recurrent baseline) to 26.7 (convolutional baseline).

Table 5. Hyperparameter configuration, with the same values used for all seven architectures and branch-specific rows applying only to models that contain that branch

Hyperparameter	Value
Embeddings	GloVe 840B, 300-d, frozen (18,000,000 parameters)
Vocabulary	60,000 (tokeniser fitted on training data only)
Sequence length	256 tokens, padded at the end
Convolutional branch	128 filters for each kernel size (3, 4, and 5), ReLU, and 'same' padding
Recurrent branch	128 units in each direction
Attention	Additive (Wu 256×256) or scaled dot-product, masked over padding
Dense head	128 units, ReLU, L2=0.001, and dropout p=0.5
Optimizer	Adam and initial learning rate 0.001
Batch size	64
Epoch budget	50, early stopping on validation AUC, and patience 8
Learning-rate schedule	factor 0.5 on plateau, patience 3, and floor 1e-6
Random seeds	42, 1337, and 2024

5.3. Computational environment

Experiments were executed on a single NVIDIA Tesla T4 GPU (16 GB) in a Kaggle notebook environment, running TensorFlow 2.20 and Python 3.11. A single device was used deliberately: under data-parallel execution the per-article latency below would measure inter-device coordination rather than model cost. Total training time for the 21 runs was 4.10 GPU-hours, a mean of 11.7 minutes per run; per architecture, summed over three seeds, 0.35 h for the recurrent baseline, 0.40 h for the convolutional baseline, 0.41 h for the recurrent model with attention, 0.54 h for the unidirectional hybrid, 0.66 h for the hybrid without attention, 0.84 h for the dot-product hybrid and 0.90 h for the additive hybrid. Mean single-article inference latency ranges from 3.43 ms to 9.48 ms.

6. RESULTS AND DISCUSSION

6.1. Architecture comparison

All metrics for the seven architectures appear in Table 6 as means over three seeds, each with its standard deviation across seeds. In Figure 6 the pooled and macro-averaged accuracies are plotted, with error bars for the seed variation.

On pooled accuracy the two single-branch models take the first two places, and the four models that pair a convolutional branch with a recurrent one are all in the bottom half. The order changes somewhat on the macro average, although the overall picture stays the same. There the recurrent model with attention (79.12) and the plain recurrent model (79.00) come first, and again the four convolutional-recurrent hybrids are at the bottom. From best to worst, the seven models differ by 1.17 points in pooled accuracy and 2.16 points in macro average. These numbers can be compared with the mean seed standard deviations of 0.33 and 1.05 points. Seven values drawn from one distribution would be expected to span about 2.5 standard deviations, that is 0.82 and 2.63 points. The macro-averaged range is therefore smaller than what seed variation alone would produce.

The ranking is also metric-dependent: no architecture leads on more than two of the columns in Table 6. On the macro average no pairwise difference reaches significance under a Welch test on the seed-level values. With three seeds the design has limited power to detect small differences, so this is an absence of detected separation rather than evidence of equivalence. The only significant differences are on pooled accuracy, between the previously proposed hybrid and the two single-branch baselines, with effect sizes an order of magnitude below the conventional threshold (subsection 6.2). The clearest observed separation is in operating-point behavior rather than in mean accuracy, which makes the choice between these architectures a decision about the relative cost of the two error types.

The CNN-BiLSTM model with additive attention, which an earlier version of this work proposed as its contribution, ranks sixth of seven on pooled accuracy and sixth on the macro average. It does not lead on any metric in Table 6. In the earlier draft it was reported to beat the two baselines by 3.9 and 6.8 points, and neither gain could be reproduced with the protocol of section 3.

Table 6. Performance on the combined test set (N=13,111), given as the mean and standard deviation over three seeds, with AUC between 0.966 and 0.971 for all seven architectures

Architecture	Accuracy	Macro avg.	Recall	Specificity	F1
CNN	89.30±0.16	77.86±0.82	88.60±0.07	89.89±0.35	88.28±0.15
BiLSTM	89.27±0.17	79.00±0.73	85.77±2.28	92.18±1.61	87.89±0.44
BiLSTM+additive attn. [7]	89.02±0.47	79.12±1.19	86.50±3.28	91.12±2.88	87.74±0.62
CNN+BiLSTM	88.13±0.27	76.96±1.07	86.26±6.20	89.68±4.70	86.81±1.08
CNN+BiLSTM+dot-product attn.	88.98±0.79	77.57±1.82	88.52±3.41	89.36±4.23	87.96±0.41
CNN+UniLSTM+additive attn.	88.70±0.26	77.57±1.02	85.10±1.97	91.70±1.25	87.24±0.50
CNN+BiLSTM+additive attn.	88.69±0.17	77.43±0.71	87.50±3.50	89.69±3.22	87.55±0.28

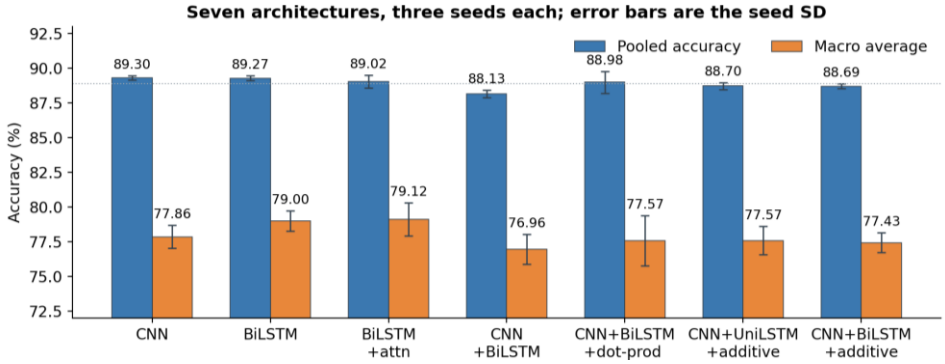


Figure 6. Pooled and macro-averaged accuracy for the seven architectures, with error bars giving the standard deviation over three random seeds

6.2. Statistical analysis

With 1,000 stratified bootstrap resamples, the accuracies of all seven architectures fall within about one point of each other. Since each interval overlaps all the others, the intervals alone do not separate any pair. The paired McNemar test is more sensitive because it compares predictions on the same articles, and it does find differences. It is computed on a single seed, so the margins below are the seed-42 differences and are

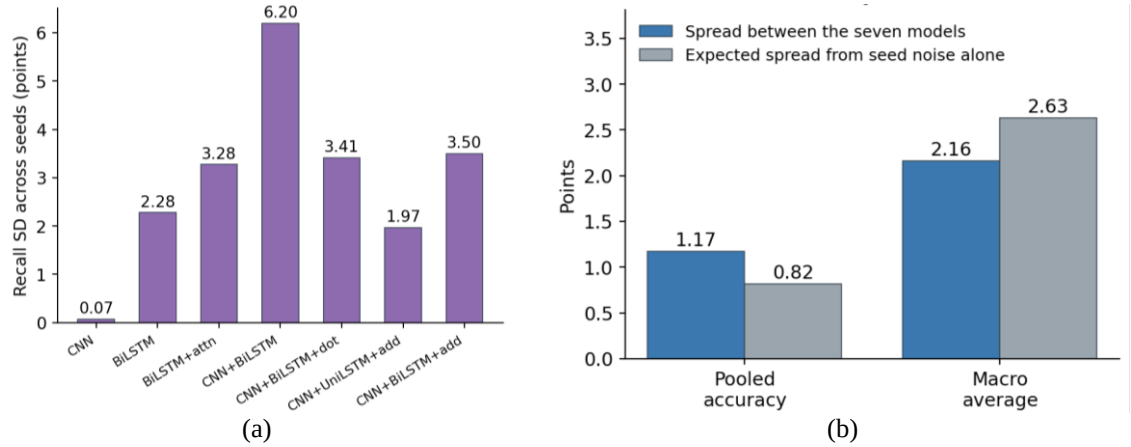
slightly larger than the three-seed means in Table 6 (0.61 and 0.58 points). Taking the CNN-BiLSTM model with additive attention as the reference, it loses to the convolutional baseline by 0.76 points (chi-square 8.68, Bonferroni-corrected p=0.019), to the recurrent baseline by 0.69 points (chi-square 8.54, p=0.021) and to the dot-product attention variant by 1.04 points (chi-square 19.68, p=0.000055). All p-values are Bonferroni-corrected for the six comparisons against the reference. Welch tests on the seed-level accuracies agree for the first two (p=0.010 and p=0.014) but not the third (p=0.604): the dot-product advantage is consistent within a seed but not across seeds.

These differences are statistically significant and practically negligible. The corresponding Cohen's h values are 0.024, 0.022, and 0.033, against a conventional threshold of 0.20 for a small effect. With 13,111 test articles the test has enough power to detect differences far below the level at which anyone would choose one model over another, which is precisely why an effect size appropriate to proportions must be reported alongside the p-value.

The clearest loss for the architecture proposed in the earlier draft, at p=0.000055, is to the same hybrid with standard scaled dot-product attention in place of the additive formulation. The additive mechanism carries 66,049 additional parameters and an explicit projection of the fused states, and performs about a point worse.

6.3. Seed stability

Mean accuracy conceals a difference between these architectures that is larger than any difference in accuracy, and that would matter more in deployment. Across the three seeds, the recall of the convolutional baseline changes by no more than 0.07 points. The CNN-BiLSTM model without attention varies by 6.20 points in recall and 4.70 in specificity, meaning its operating point moves by twelve points between runs that differ only in initialization. The additive-attention hybrid varies by 3.50 and 3.22. Figure 7 illustrates the performance differences and seed sensitivity across the seven architectures. Figure 7(a) shows the recall standard deviation across the three seeds, while Figure 7(b) compares the observed performance spread between architectures with the expected spread from seed variation.



(a) (b)

Figure 7. Seed sensitivity of the seven architectures, showing; (a) the recall standard deviation of each architecture over the three seeds and (b) the spread between architectures compared with that expected from seed variation

For a classifier deployed with a fixed decision threshold, this variance is passed straight through. Train it twice with the same recipe and the two resulting systems can have quite different false-positive rates. The ordering follows architectural complexity without exception: the single-branch models vary least, the composite ones most. Our interpretation is that the recurrent layer, when it receives sequences of convolutional features, converges to quite different solutions depending on initialisation. On stability grounds and independently of accuracy, the simplest architecture is to be preferred by a wide margin.

Figure 8 gives the seed-42 confusion matrix of the convolutional baseline. Real articles are recognised correctly in 6,441 of 7,153 cases (specificity 90.05%), fabricated ones in 5,276 of 5,958 cases (recall 88.55%); there are 712 false positives and 682 false negatives. Because the two error counts are almost equal, we find no evidence of a systematic preference for either class.

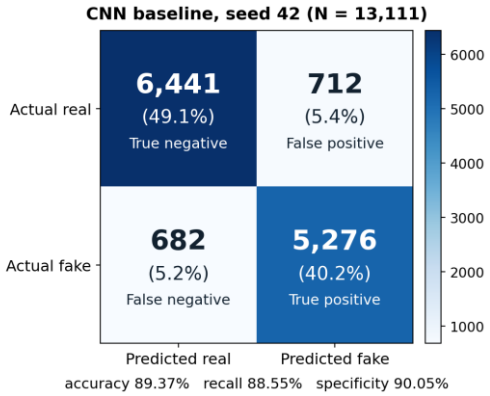


Figure 8. Seed-42 confusion matrix of the convolutional baseline, combined test set (N=13,111)

6.4. What the pooled figure is measuring

Table 7 breaks the pooled accuracy of the convolutional baseline down by dataset and lists, next to each slice, the accuracy of a majority-class predictor (the corresponding plot for all seven architectures is Figure 9). Read this way, most of the numbers given earlier in this section change their meaning.

Table 7. Per-dataset results of the convolutional baseline, in which contribution is the dataset accuracy multiplied by its test-set share, in points of the pooled accuracy

Dataset	Share of test set (%)	Accuracy	Majority baseline	Lift	AUC
WELFake	72.5	96.06±0.13	54.51	+41.54	0.993
COVID-19	11.9	91.63±0.58	53.05	+38.58	0.974
FakeNewsNet	1.1	68.30±2.46	56.64	+11.66	0.829
LIAR	14.6	55.44±0.82	55.86	-0.42	0.575
Pooled	100	89.30±0.16	54.56	+34.75	0.971

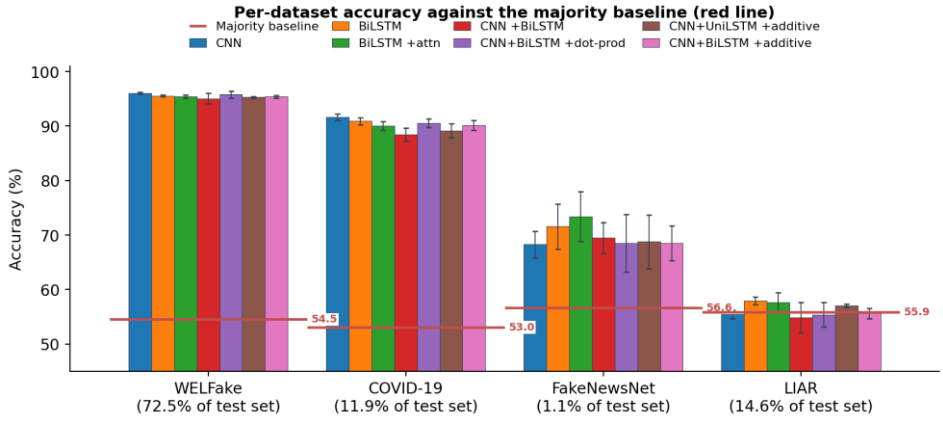


Figure 9. Per-dataset accuracy for all seven architectures against the majority baseline of each slice (red line), with error bars giving the standard deviation over three seeds

Of the 89.30 pooled points, WELFake is responsible for 69.59. If WELFake is left out the accuracy is 71.55%, and the unweighted mean of the four per-dataset accuracies is 77.86%; both should be set against 89.30%. A large part of the headline number is due to the way the test set is composed, and not to the classifier.

For LIAR the situation is worse. On this dataset the baseline's accuracy is 55.44%, which is lower than the 55.86% a majority-class predictor would get (lift of minus 0.42 points); the AUC is 0.575. The other six architectures behave in the same way. Over all 21 runs the LIAR AUC never leaves the range 0.57 to 0.59, and its standard deviation is close to zero. We conclude that, in our setup, none of the models discriminates usefully on LIAR, and with so little variation between runs neither the choice of seed nor the tuning can be the

cause. The reason becomes visible in the predicted probabilities, which for LIAR are concentrated near 0.39 (standard deviation 0.16) while for WELFake they are spread over the full interval (standard deviation 0.48). With eighteen tokens of claim and nothing else to work from, the classifier hedges, and the hedge ends in the majority class.

Published accuracies for binary LIAR lie at roughly 62% to 70%, not far from what we find. An earlier version of this paper, by contrast, reported 88.4% on the same slice, 33 points above our current figure. The same earlier version gave 91.8% for the political domain, which did not agree with its own per-dataset table, and with the new figures that inconsistency disappears.

This has two consequences in practice. The first concerns pooled metrics: when a corpus is composed like ours, they are hardly sensitive to modest architectural differences. About 72.5% of the test set is already saturated at around 96% and varies by no more than 0.13 points between seeds, so in practice the comparison is decided on the remaining quarter. Second, a study reporting 98% on WELFake and a study reporting 60% on LIAR are not in disagreement. They are measuring different things, and the field's forty-point spread is substantially explained by which corpus was chosen rather than by which model was built.

The same decomposition holds for every architecture in Table 6. Figure 9 shows that the between-architecture differences on each slice are smaller than the seed error bars in almost every case, and that on FakeNewsNet, with 143 test articles, no ordering can be read at all.

6.5. Ablation

Table 8 presents the ablation path from the complete model. Each row removes or replaces one component while holding the splits, seeds and every other setting fixed, and reports the change in macro-averaged accuracy together with a Welch test on the three seed-level values. The first row is the variant Reviewer feedback identified as missing from our earlier draft: the recurrent branch with attention but no convolution, which corresponds to the published architecture of Mohapatra *et al.* [7] and which the earlier version claimed was the worst of the set.

Table 8. Ablation of the CNN-BiLSTM model with additive attention over three seeds per row, where a positive delta means the ablated model performs better than the complete one

Ablation	Accuracy	Macro avg.	Δ macro	Welch p	Recall SD
Remove the CNN branch, keep attention [7]	89.02±0.47	79.12±1.19	+1.69	0.118	3.28
Remove the CNN branch and attention (BiLSTM)	89.27±0.17	79.00±0.73	+1.57	0.057	2.28
Remove the recurrent branch (CNN only)	89.30±0.16	77.86±0.82	+0.42	0.536	0.07
Replace additive with dot-product attn.	88.98±0.79	77.57±1.82	+0.14	0.911	3.41
Unidirectional LSTM instead of BiLSTM	88.70±0.26	77.57±1.02	+0.14	0.858	1.97
Complete model	88.69±0.17	77.43±0.71	-	-	3.50
Remove attention	88.13±0.27	76.96±1.07	-0.47	0.565	6.20

In five of the six ablations the reduced model does better than the complete one. The biggest gain (+1.69 macro points) is obtained when the convolutional branch is removed and attention is kept. Our earlier draft had reported a loss of 3.1 points for exactly this ablation and had named the convolutional branch as the most important component, so the direction of the effect has flipped while its size is similar. We do not want to make much of this reversal. None of the rows in Table 8 is significant (p-values between 0.057 and 0.911), and under the present protocol the experiment gives no reliable evidence that any of the components helps macro-averaged accuracy.

Still, the results are not random in shape. On the macro average, every configuration that has both branches scores below both single-branch models, and the one configuration without a convolutional branch is the best of all seven. One possible explanation is that feeding convolutional outputs as a sequence into the recurrent layer makes the representation worse instead of richer, and that attention makes up for part of this damage. That would also explain why removing attention is the only ablation with a cost. We offer this as the most economical account of Table 8 rather than as a demonstrated mechanism; the differences involved are not significant.

6.6. Comparison with reported results

Table 9 places our figures beside results reported elsewhere. The comparison is not like-for-like and we do not present it as one: the studies differ in corpus, in split procedure and in whether publisher artifacts were removed, and those differences are larger than the differences between the numbers.

Table 9. Reported accuracies under different evaluation conditions

Method	Reported accuracy	Corpus and split	Artifacts removed/seeds
FakeStack [6]	98.25%/75.58%	WELFake/LIAR, random	Not reported/1
Att-BiLSTM [7]	98.65%	single dataset, random	Not reported/1
BERT-base [28]	93-95%	single dataset, random	Not reported/not reported
TRANSFAKE [30]	91-93%	multimodal, random	Not reported/not reported
This work, CNN (pooled)	89.30±0.16	4 datasets, source-aware	Yes/3
This work, CNN (WELFake slice)	96.06±0.13	WELFake, source-aware	Yes/3
This work, CNN (LIAR slice)	55.44±0.82	LIAR, source-aware	Yes/3

Read down the last three columns rather than the second. Our WELFake slice reaches 96.06% and would sit comfortably among the high figures in the table had we reported it alone; our LIAR slice reaches 55.44% and would sit at the bottom. Both come from the same model, the same run and the same training corpus. The 40-point spread that characterizes this literature is reproduced within a single one of our experiments, indicating that the spread is a property of evaluation conditions rather than of model quality.

Two further columns are worth noting because they are almost never populated in this literature. No study we compare against reports whether publisher artifacts were removed, and none reports results over multiple seeds. On our measurements the first is worth four to seven points on WELFake and the second determines whether a one-point difference means anything.

6.7. A linear reference point

The comparisons so far are internal to the neural family. To place them on an absolute scale we trained a TF-IDF LR (unigrams and bigrams, 100,000 features, sublinear term frequency) on the identical partition. The classifier is deterministic, so it carries no seed variance, and it fits in 77 seconds on CPU against 4.10 GPU-hours for the 21 neural runs. Table 10 reports it in two configurations: one model trained on the pooled corpus, and a separate model per dataset.

Table 10. Accuracy of a TF-IDF LR against the neural architectures on the same partition, with the pooled column undefined for per-dataset models

Configuration	WELFake	COVID-19	FakeNewsNet	LIAR	Macro (pooled)
Majority baseline	54.51	53.05	56.64	55.86	55.02 (54.56)
Linear, one pooled model	94.76	88.12	69.23	56.07	77.04 (88.05)
Linear, one model per dataset	95.24	91.78	78.32	59.05	81.10 (n/a)
Best neural on each slice	96.06	91.63	71.56	57.97	79.31 (89.30)
CNN+BiLSTM+additive attn.	95.41	90.15	68.53	55.63	77.43 (88.69)

On the pooled corpus the best neural architecture exceeds the linear model by 1.25 points and the previously proposed hybrid by 0.64. For that margin the neural models require 19.1 million parameters, a 5.6 GB pre-trained embedding file and 4.10 GPU-hours across the 21 runs. Fitting the proposed hybrid once takes about 1,080 s of GPU time against 77 s of CPU time for the linear model, a factor of fourteen; the full campaign of 21 runs is a factor of 192. Neither figure accounts for the accelerator itself, which the linear model does not need. Whether 0.64 accuracy points justifies that difference depends on the deployment.

The per-dataset rows are less equivocal. Fitting a separate linear model to each corpus gives a macro average of 81.10%, which exceeds the macro average of the best neural result on each slice (79.31%) and every individual architecture in Table 6, including the best of the seven at 79.12%. On LIAR the linear model reaches 59.05% against 57.97% for the best neural architecture, and on FakeNewsNet 78.32% against 71.56%. The neural models retain an advantage only on WELFake, by 0.82 points. Two conclusions follow. First, the weakness on LIAR reported in subsection 6.4 is not a limitation of neural capacity: a bag-of-bigrams classifier extracts more from those eighteen-token claims than any architecture we trained, which locates the ceiling in the data. Second, pooling the corpora is itself harmful, since per-dataset linear models beat the pooled linear model on all four slices.

We report this comparison not to argue that neural architectures are useless for this task, but because it is the reference point that makes the other numbers interpretable. A 1.17-point spread between seven architectures is difficult to assess in isolation; against a 1.25-point total margin over a classifier from the 1950s, it is easier to judge what is at stake.

7. LIMITATIONS AND FUTURE WORK

7.1. Limitations

The linguistic and modal scope is narrow: all experiments use English text only, with no social context, propagation signal or imagery. Given the sharp drops in performance reported for other languages in [21], [22], we cannot assume that our findings carry over to them.

The study includes no transformer baseline, since we did not fine-tune BERT [28], RoBERTa [29] or a distilled variant [36] under this protocol, and the comparison in Table 9 therefore quotes transformer figures obtained under other conditions. This is the most significant gap in the present study. We note that the leakage measurements in subsection 4.2 apply to transformer evaluations with equal force, since the artifacts are properties of the corpus rather than of the model, so a transformer evaluated on unscrubbed WELFake inherits the same four to seven points.

Truncation limits the input, because the 256-token window covers the whole of three corpora but only about 46% of the mean WELFake article. A sensitivity analysis at longer windows was not run, and it is possible that the recurrent and hybrid models are penalized more by truncation than the convolutional baseline, which pools globally.

Source isolation is weak on the dominant corpus. On WELFake the tier-2 groups contain 1.03 articles on average, so once exact duplicates are gone, the split of this corpus, which makes up 72% of the data, is close to random. Source leakage is controlled for corpora with a source field, but for WELFake we rely on artifact scrubbing.

Scrubbing is blunt. When an image-credit line is deleted, some real stylistic information goes with the shortcut, and we have no way of keeping one without the other. The post-scrubbing results are thus best seen as a conservative estimate and not as an upper limit.

A negative result is not proof of absence. What we show is that, on this corpus, with this protocol and three seeds, the seven architectures cannot be told apart reliably. Three seeds give little power to detect small effects, so failing to find a significant difference does not mean the models are equivalent. They might well separate on a larger or better-balanced corpus, or if a different family of architectures were included, and our conclusion is limited to the conditions tested.

7.2. Future directions

Four directions follow directly from the measurements reported above. Each addresses a question that the present evidence could not settle, and each is framed so that the answer would be measurable under the same leakage-audited protocol. The first is the most consequential, since the absence of a transformer baseline is the clearest gap in this study.

- Evaluating transformers under the same protocol (scrubbed artifacts, several seeds) to see whether the advantage reported for them holds up.
- Leave-one-dataset-out experiments, where one corpus is kept completely out of training. The experiments reported here measure in-distribution performance on a pooled multi-corpus dataset, which is why this paper describes itself as a cross-dataset rather than a cross-domain evaluation; genuine domain transfer remains unmeasured and, given the LIAR result, is unlikely to be strong.
- Extension of the artifact audit to other widely used corpora, particularly ISOT and its derivatives, where the same wire-service provenance is present by construction.
- Evidence-grounded verification, since the failure mode identified here is that models learn stylistic rather than epistemic signal, which is what limits transfer to slices such as LIAR where style is uninformative.

8. CONCLUSION

This paper set out to explain why reported accuracies for fake news detection span forty points, and to determine what remains of architectural differences once the explanation is removed. On a combined corpus of 87,408 articles we identified and quantified three leakage channels: publisher artifacts covering 32.8% of the 96,626 documents collected before deduplication, at 99.7% class purity, a fifteen-fold class asymmetry in exact duplication, and outlet-level fingerprints. Only the third is addressed by the source-aware splitting now beginning to appear in this literature, and it is the least consequential of the three.

Under the three-part leakage-mitigation protocol, seven architectures trained under three seeds each show no robust separation: the macro-averaged spread of 2.16 points is smaller than the 2.63 expected from seed variation alone, and the few significant differences on pooled accuracy carry Cohen's h below 0.04. The hybrid CNN-BiLSTM model with additive attention that an earlier version of this work proposed ranks sixth of seven, and we retract the 3.9 and 6.8 point gains that version claimed; the ablation it reported as worst in fact ranks first on the macro average. A TF-IDF LR fitted in 77 seconds on CPU reaches 88.05% against the

best neural 89.30%. What does separate the neural architectures is stability: recall reproduces to 0.07 points across seeds for the convolutional baseline and varies by 6.20 for the composite model.

The per-dataset decomposition explains the field's spread directly. WELFake supplies 69.59 of our 89.30 pooled accuracy points, while on LIAR every architecture and seed returns an AUC between 0.57 and 0.59. A study reporting 98% on WELFake and one reporting 60% on LIAR are not in disagreement: the same model of ours produces 96.06% and 55.44% on those slices in a single run. Architectural gains of one to two points reported on pooled corpora of this kind are therefore not measurable with the evidence usually provided, and effort spent auditing a corpus returns more than effort spent adding components to a model. What the study leaves behind is reusable independently of its conclusions: an artifact audit that runs on any labelled corpus, a set of reporting rules, and per-article predictions from which every results table here can be rebuilt. The significance of this work lies in that material and in the protocol, not in any architecture outperforming another.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Isra University for its institutional support of this research, which was carried out without any research grant or contract.

FUNDING INFORMATION

This research received no research grant or contract from any funding agency in the public, commercial, or not-for-profit sectors.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Mohammad Alkhazaleh	✓	✓	✓	✓	✓	✓		✓	✓		✓		✓	
Mahmoud Baklizi	✓	✓		✓	✓	✓	✓			✓		✓		
Musab Alzghoul	✓		✓	✓	✓		✓	✓		✓	✓			
Loui Al Sardy		✓		✓		✓				✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare no conflicts of interest.

DATA AVAILABILITY

The code, split assignments, per-article predictions and training logs generated by this study are available at https://github.com/moh1984/fake-news-leakage-audit. The four source corpora are publicly available from their original providers at the locations below; they are not redistributed there.

- LIAR [8]: https://www.cs.ucsb.edu/~william/data/liar_dataset.zip.
- FakeNewsNet [11]: https://github.com/KaiDMML/FakeNewsNet.
- WELFake [12]: https://www.kaggle.com/datasets/saurabhshahane/fake-news-classification.
- COVID-19, Constraint@AAAI2021 [31]: https://github.com/diptamath/covid_fake_news.

REFERENCES

[1] World Health Organization, "Infodemic," *Geneva, Switzerland*, 2020. [Online]. Available: https://www.who.int/health-topics/infodemic. (accessed: Nov. 10, 2026).
[2] S. Vosoughi, D. Roy, and S. Aral, "The spread of true and false news online," *Science*, vol. 359, no. 6380, pp. 1146–1151, Mar. 2018, doi: 10.1126/science.aap9559.

[3] M. R. Islam, S. Liu, X. Wang, and G. Xu, "Deep learning for misinformation detection on online social networks: a survey and new perspectives," *Social Network Analysis and Mining*, vol. 10, no. 1, p. 82, Dec. 2020, doi: 10.1007/s13278-020-00696-x.

[4] T. Alhindi, S. Petridis, and S. Muresan, "Where is your evidence: Improving fact-checking by justification modeling," in *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, 2018, pp. 85–90, doi: 10.18653/v1/W18-5513.

[5] L. Syed, A. Alsaeedi, L. A. Alhuri, and H. R. Aljohani, "Hybrid weakly supervised learning with deep learning technique for detection of fake news from cyber propaganda," *Array*, vol. 19, p. 100309, Sep. 2023, doi: 10.1016/j.array.2023.100309.

[6] A. J. Keya, H. H. Shajeeb, M. S. Rahman, and M. F. Mridha, "FakeStack: Hierarchical Tri-BERT-CNN-LSTM stacked model for effective fake news detection," *PLOS ONE*, vol. 18, no. 12, p. e0294701, Dec. 2023, doi: 10.1371/journal.pone.0294701.

[7] A. Mohapatra, N. Thota, and P. Prakasam, "Fake news detection and classification using hybrid BiLSTM and self-attention model," *Multimedia Tools and Applications*, vol. 81, no. 13, pp. 18503–18519, May 2022, doi: 10.1007/s11042-022-12764-9.

[8] W. Y. Wang, "'Liar, liar pants on fire': A new benchmark dataset for fake news detection," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Jul. 2017, pp. 422–426, doi: 10.18653/v1/P17-2067.

[9] R. R. Rajalaxmi, L. V. N. Prasad, B. Janakiramaiah, C. S. Pavankumar, N. Neelima, and V. E. Sathishkumar, "Optimizing hyperparameters and performance analysis of LSTM model in detecting fake news on social media," *ACM Transactions on Asian and Low-Resource Language Information Processing*, Mar. 2022, doi: 10.1145/3511897.

[10] O. I. Aboulola and M. Umer, "Novel approach for Arabic fake news classification using embedding from large language features with CNN-LSTM ensemble model and explainable AI," *Scientific Reports*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-82111-5.

[11] K. Shu, D. Mahudeswaran, S. Wang, D. Lee, and H. Liu, "FakeNewsNet: a data repository with news content, social context, and spatiotemporal information for studying fake news on social media," *Big Data*, vol. 8, no. 3, pp. 171–188, Jun. 2020, doi: 10.1089/big.2020.0062.

[12] P. K. Verma, P. Agrawal, I. Amorim, and R. Prodan, "WELFake: Word embedding over linguistic features for fake news detection," *IEEE Transactions on Computational Social Systems*, vol. 8, no. 4, pp. 881–893, Aug. 2021, doi: 10.1109/TCSS.2021.3068519.

[13] Y. Yang, L. Zheng, J. Zhang, Q. Cui, Z. Li, and P. S. Yu, "TI-CNN: Convolutional neural networks for fake news detection," *arXiv Preprint*, Jun. 2018, doi: 10.48550/arXiv.1806.00749.

[14] K. Ivancová, M. Sarnovský, and V. Maslej-Krešňáková, "Fake news detection in Slovak language using deep learning techniques," in *2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Jan. 2021, pp. 255–260, doi: 10.1109/SAMI50585.2021.9378650.

[15] H. Padalko, V. Chomko, and D. Chumachenko, "A novel approach to fake news classification using LSTM-based deep learning models," *Frontiers in Big Data*, vol. 6, p. 1320800, Jan. 2024, doi: 10.3389/fdata.2023.1320800.

[16] S. H. Nithya and A. Sahayadhas, "Automated fake news detection by LSTM enabled with optimal feature selection," *Journal of Information & Knowledge Management*, vol. 21, no. 3, p. 2250036, Sep. 2022, doi: 10.1142/S0219649222500368.

[17] R. Dutta, D. R. D. Adhikary, and M. Majumder, "A deep learning model for classification of COVID-19 related fake news," in *Electronic Systems and Intelligent Computing (Lecture Notes in Electrical Engineering)*, Singapore: Springer, 2022, vol. 860, pp. 449–456, doi: 10.1007/978-981-16-9488-2_42.

[18] S. M. Jaybhaye, V. Badade, A. Dodke, A. Holkar, and P. Lokhande, "Fake news detection using LSTM based deep learning approach," in *ITM Web of Conferences*, vol. 56, p. 03005, Aug. 2023, doi: 10.1051/itmconf/20235603005.

[19] V. S. Nirban, T. Shukla, P. S. Purkayastha, N. Kotalwar, and L. Ahsan, "A machine learning perspective on fake news detection: a comparison of leading techniques," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 15, pp. 59–68, 2023.

[20] T. Islam, M. A. Hosen, A. Mony, M. T. Hasan, I. Jahan, and A. Kundu, "A proposed Bi-LSTM method to fake news detection," in *2022 International Conference for Advancement in Technology (ICONAT)*, Jan. 2022, pp. 1–5, doi: 10.1109/ICONAT53423.2022.9725937.

[21] C.-O. Truică, E.-S. Apostol, and A. Paschke, "Awakened at CheckThat! 2022: fake news detection using BiLSTM and sentence transformer," in *CLEF 2022: Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings*, vol. 3180, 2022, pp. 749–757.

[22] D. V. Vo and P. Do, "Detecting Vietnamese fake news," *CTU Journal of Innovation and Sustainable Development*, vol. 15, no. ISDS, pp. 39–46, Oct. 2023, doi: 10.22144/ctujoisd.2023.033.

[23] S. Akilandeswari, S. T R, S. Sumathi, M. Sushith, V. Brinda, and S. Rajan, "A hybrid CNN-LSTM-PSO framework for enhanced cybersecurity threat detection and classification," in *2025 International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI)*, Jan. 2025, pp. 1956–1965, doi: 10.1109/ICMSCI62561.2025.10894126.

[24] Y. Huang *et al.*, "An improved hybrid CNN-LSTM-attention model with Kepler optimization algorithm for wind speed prediction," *Engineering Letters*, vol. 32, no. 10, pp. 1957–1965, 2024.

[25] Y.-W. Tseng, H.-K. Yang, W.-Y. Wang, and W.-C. Peng, "KAHAN: Knowledge-aware hierarchical attention network for fake news detection on social media," in *Companion Proceedings of the Web Conference 2022*, Apr. 2022, pp. 868–875, doi: 10.1145/3487553.3524664.

[26] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv Preprint*, Sep. 2014, doi: 10.48550/arXiv.1409.0473.

[27] N. Aloshban, "ACT: Automatic fake news classification through self-attention," in *12th ACM Conference on Web Science*, Jul. 2020, pp. 115–124, doi: 10.1145/3394231.3397901.

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, vol. 1, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

[29] Y. Liu *et al.*, "RoBERTa: A robustly optimized BERT pretraining approach," *arXiv Preprint*, Jul. 2019, doi: 10.48550/arXiv.1907.11692.

[30] Q. Jing *et al.*, "TRANSFAKE: Multi-task transformer for multimodal enhanced fake news detection," in *2021 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2021, pp. 1–8, doi: 10.1109/IJCNN52387.2021.9533433.

[31] Y. Tashtoush, B. Alrababah, O. Darwish, M. Maabreh, and N. Alsaedi, "A deep learning framework for detection of COVID-19 fake news on social media platforms," *Data*, vol. 7, no. 5, p. 65, May 2022, doi: 10.3390/data7050065.

[32] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543, doi: 10.3115/v1/D14-1162.

[33] Q. McNemar, "Note on the sampling error of the difference between correlated proportions or percentages," *Psychometrika*, vol. 12, no. 2, pp. 153–157, Jun. 1947, doi: 10.1007/BF02295996.

[34] J. Li, "Area under the ROC curve has the most consistent evaluation for binary classification," *PLOS ONE*, vol. 19, no. 12, p.

e0316019, Dec. 2024, doi: 10.1371/journal.pone.0316019.
[35] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. Chapman and Hall/CRC, 1994, doi: 10.1201/9780429246593.
[36] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv Preprint*, Mar. 2020, doi: 10.48550/arXiv.1910.01108.

BIOGRAPHIES OF AUTHORS



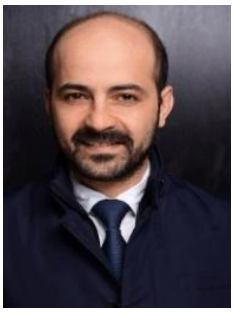
Mohammad Alkhazaleh received the degree in Information Technology and Computing from Arab Open University (AOU), Jordan in 2009. He received the master degree in Computer Science from The Middle East University for the graduate studies (MEU), Jordan in 2011. He received the Ph.D. degree in Computer Engineering from University Malaysia Perlis (UniMAP), Malaysia in 2021. He was a lecturer at Faculty of Applied Studies and Continuing Education at Al-Baha University (2011-2016), and is currently an assistant professor in Department of Computer Sciences at the College of Information Technology at Isra University (2022-present). He can be contacted at email: m.alkhazaleh@iu.edu.jo.



Mahmoud Baklizi 17 years of experience in academia in Jordan. Extensive theoretical and practical expertise in network security, network design, and implementation, with additional proficiency in security, vulnerability assessment, and programming. Several years of managerial experience in the academic sector in Jordan, with strong capabilities in strategic and action plan development. Interdisciplinary and intercultural experience in scientific research, applied research, and research and development within both academic and industrial fields in Jordan and Malaysia. Advanced proficiency in English. He can be contacted at email: mbaklizi@uop.edu.jo.



Musab Alzghoul received his bachelor of science in Computer Science in Don State Technical University (DSTU) Russia in 2005. In 2006, he received his master degree in Computer Science at DSTU, and then in 2009, he received a Ph.D. in Computer Science. He was a member of the Faculty of Information Technology at Zarqa University where he was a lecturer between 2009 and 2012. He was a lecturer in 2012-2023 at the Faculty of Information Technology of Um-AlQura University. He is now an assistant professor in the Department of Computer Sciences at the Isra University where he has been teaching since 2023. He can be contacted at email: musab.alzgool@iu.edu.jo.



Loui Al Sardy is a researcher and head of the Network Security Research Group at the Department of Computer Science (Informatik 7) at Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany. He is interested in research related to network security, intrusion detection using artificial intelligence, network fuzzing, cybersecurity of cyber-physical systems, and smart grid/critical infrastructure systems. He is passionate about building innovative and powerful security solutions with AI, deep learning, and large language models. He is engaged in research collaborations across the globe, academic teaching and supervision of doctoral and graduate researchers. He can be contacted at email: loui.alsardy@fau.de.