Sentiment Classification of Consumer Reviews of Amazon Products Under Severe Class Imbalance: A Reproducible Comparison of Convolutional Neural Networks and Sparse Linear Baselines

Shamik Palit[1]*, Preetha V. K.[2], Dhanya Sudarsan[3], Annie Edwis Threse F.[4], Jimy[5], Santhuana Sedhumadhavan[6]
Associate Professor, Faculty of Natural Sciences-Department of Computing Science, Software Engineering-Data Science and Artificial Intelligence, University of Stirling RAK Campus, Ras Al Khaimah, United Arab Emirates[1, 2, 3, 4]
Lecturer, Faculty of Natural Sciences-Department of Computing Science, Software Engineering-Data Science and Artificial Intelligence, University of Stirling RAK Campus, Ras Al Khaimah, United Arab Emirates[5, 6]

Abstract*—Sentiment models that determine the sentiment of online reviews, i.e., programs that can automatically distinguish whether a user's review is praising or criticizing a product, often need to process data with an extremely large gap between positive and negative samples. The positive and negative samples mentioned here are pre-labeled positive reviews and negative reviews. If the training data for a model has far more positive reviews than negative reviews, with an especially stark imbalance, a major problem emerges: even if the model cannot identify negative reviews at all, it can still achieve a decent overall accuracy rate by random guessing. The small proportion of negative reviews is easily overlooked by conventional performance evaluation systems, and the model's poor ability to identify negative reviews often goes undetected. For this study, we specifically selected a dataset that fits this extreme imbalance scenario, the Consumer Reviews of Amazon Products dataset, to test the real-world performance of two commonly used sentiment recognition models—CNNs and sparse linear classifiers. This dataset contains 32,252 positive reviews and only 810 negative reviews, with an extremely disparate number of positive and negative reviews, which is perfectly suited to measure the true capability of models under extreme sample imbalance. Instead of randomly splitting the data, we conducted a stratified split at ratios of 70%, 15%, and 15%. In each of the three resulting subsets, the ratio of positive to negative reviews is consistent with that of the full dataset, eliminating the risk of an unexpected increase or decrease in the number of negative reviews in any single subset. These three parts are the training set, which is used to teach the model to learn patterns; the validation set, which is used to adjust the model's parameters during training; and the test set, which is used for the final assessment of the model's capabilities. At no point did the model gain prior access to the content of the test set. We tested a total of four distinct models: The first is the most basic reference, a baseline model that only predicts the majority class. This model learns nothing and uniformly predicts all reviews as the more numerous positive reviews, to anchor the minimum passing threshold for the models. The second model uses TF-IDF to convert text into features that the model can interpret, then uses a logistic regression algorithm to make judgments. The third model also extracts features with TF-IDF but uses a linear support vector machine algorithm to make judgments. The final model is a CNN. In the final assessment of all models using the reserved test set that the models had never encountered before, the linear support vector machine achieved the most outstanding balanced performance. The macro-averaged F1 score used here is a fair metric specifically designed for sample imbalance scenarios, which will not inflate the total score simply because one class has more samples, and can truly reflect the model's comprehensive ability to identify both positive and negative reviews. Its macro-averaged F1 score reached 0.832, and the logistic regression model that followed closely behind had a nearly comparable score. In contrast, the CNNs performed far worse, with a macro-averaged F1 score of only 0.620, which is significantly lower than the first two types of sparse linear models. To rule out the possibility that the single test result was a coincidence, we additionally conducted multiple rounds of stability verification, and repeated tests all produced consistent results: sparse linear models consistently held an advantage. These conclusions prove that CNNs are not the optimal choice for all comment sentiment classification tasks with imbalanced samples, and also remind relevant researchers that they must attach importance to three matters: they must use evaluation metrics adapted to sample imbalance scenarios, adjust the classification threshold based on the performance of the validation set, and list high-performing sparse linear models as a baseline reference that must be included for comparison.

Keywords*—Sentiment analysis; amazon products; consumer reviews; convolutional neural network; support vector machine; class imbalance; reproducibility; customer analytics

I. Introduction

The reviews left by users hide a great deal of useful information directly related to products. To unearth this information, one must first filter out negative reviews from dissatisfied users among the overwhelming volume of routine positive reviews; after all, the vast majority of reviews praise products, and voices of dissatisfaction are mixed in among them. Without separating them first, it is impossible to obtain truly valuable reference information. Related work on opinion mining and sentiment analysis establishes online reviews as a useful source of consumer preference signals [22, 26]. Consumer-review sentiment analysis also supports e-commerce decision-making and recommendation applications [7, 15].

Conducting sentiment classification on reviews is inherently a task with imbalanced sample sizes. There are far

*Corresponding author.

too many positive reviews and too few negative reviews, creating a huge gap in their numbers. More critically, letting a serious user complaint slip by carries a far greater cost than mistakenly flagging an irrelevant review as a problem. The former would cause you to miss user demands that genuinely require attention, while the latter only requires a small amount of extra effort to verify; the losses from these two scenarios are completely incomparable in magnitude. Similar skewed distributions are a recognised challenge in deep-learning classification, where the majority class can dominate learning and evaluation [16].

The earliest sentiment classification models could only operate after manually organized input features were provided. Back then, to make a model understand whether a piece of review was a compliment or a complaint, people had to first list and organize features such as "complimentary words" and "critical words" one by one, feed them to the model, and only then could the model perform its work. Later, deep learning technology fully automated the process of extracting valid information from unstructured text, eliminating the need for people to manually organize these features; machines could dig out usable information from large volumes of user reviews on their own. Relying on these advantages, convolutional neural networks gradually gained widespread adoption. They can capture local sentiment expressions in text and generate compact data representations, with related research supported by [1]. This progression from lexicon-based approaches to supervised methods is documented in foundational sentiment-analysis research and later surveys [2, 22, 26].

In recent years, increasingly complex neural network architectures have emerged one after another, but three elements have always remained the foundation of rigorous research in this field: constructing rigorous datasets, preparing mature baseline models, and adopting performance evaluation metrics that can distinguish between different class weights. Even with newer, more impressive model structures, these three core tasks cannot be skipped; missing any one of them would undermine the credibility of the research. Reproducible reporting and transparent model documentation are also important for assessing dataset limitations and model behaviour [9, 21].

The original manuscript that triggered this study once tested a CNN model using Amazon product review data, but that article had two unavoidable problems: first, its experimental methods could not be fully replicated; other researchers following the procedures it described could not produce consistent results; second, it lacked supporting standardized comparative evaluation metrics, making it impossible to compare its model with other models in the same field to determine their relative performance. Prior work has compared text-based consumer-review sentiment methods and deep-learning approaches in e-commerce settings [7, 8, 29].

This study addresses these gaps: we conducted a fully transparent analysis of Amazon product reviews, with all details documented. Throughout the process, we did not manually adjust the proportional distribution of reviews across different classes, preserving the natural class distribution in the original dataset. At the same time, we conducted comparative tests between the compact CNN for Amazon model and traditional linear baseline models, completing the work that the original manuscript failed to accomplish.

The study is guided by three research questions:

(RQ1) How well do compact single- and multi-kernel CNNs classify rating-derived review sentiment under severe class imbalance?

(RQ2) Do CNNs outperform TF-IDF logistic regression and linear SVM when evaluation includes macro-F1, MCC, and negative-class PR-AUC?

(RQ3) How robust are the strongest models when tested on products not represented in training?

The contribution is empirical rather than architectural novelty. It provides an auditable dataset report, a threshold-aware evaluation protocol, three-seed CNN estimates, a product-held-out robustness analysis, and a practical interpretation of false-positive and false-negative review classifications. These elements convert a conceptual CNN study into a defensible journal-style investigation. The comparison is also motivated by evidence that model complexity and performance must be assessed against suitable baselines and task-specific data [7, 13, 26].

II. Related Work

A. Sentiment Analysis and Online Reviews

What sentiment analysis does is very simple to describe: it extracts the hidden preferences from the text people write—whether it is like or dislike, approval or dissatisfaction. This technology did not look like it did today from the start. In its earliest days, it judged emotions relying on pre-organized lexicons, and later gradually developed the supervised machine learning path of training models with annotated data. Early related research can be found in [2]. Nowadays, online user reviews are particularly suitable for this type of research. Every piece of review text written casually by users is paired with a corresponding star rating, which perfectly matches the text content with quantitative sentiment indicators. However, these star ratings are unreliable; they cannot accurately align with the real positive or negative tendencies hidden in the text, and instead introduce a lot of interference, so they can only be used as inaccurate proxy indicators. Broader accounts of opinion mining describe the development of sentiment analysis from lexical approaches to supervised learning and modern neural methods [22, 26].

B. CNNs for Text Classification

Convolutional neural networks for processing text, that is, Text CNNs, use three basic core components: dense numeric vectors converted from text, one-dimensional filters, and pooling operations. This model, which does not have a complex structure, can achieve performance comparable to many more complex models in sentence classification tasks, with quite outstanding competitiveness [1]. This result is achievable because the model can extract information through the sequential order of words and can also perform convolutional processing at the character level. The verification of this related performance is mentioned in multiple previous studies [3].

There is also another type of model with a multi-kernel design, which can capture the patterns hidden in texts of different lengths. In addition, convolutional neural networks themselves have high operating efficiency and can process large-scale data fast enough, making them an ideal choice for sorting massive user reviews. Character-level and word-order convolutional methods have also been explored for text categorisation, supporting CNNs as a relevant comparison family [28, 3].

C. Transformer and Hybrid Models

Transformer models can handle contextual dependencies in sequence data; that is, they can capture the associations between words at different positions in a piece of text [4].

In e-commerce research, such models are often used in combination with recurrent layers and convolutional layers [5]. The model structures produced by such combinations are relatively complex. If we want to compare such structurally complex models with small, simple models, we must first control many variables—such as data splitting methods, classification thresholds, and available computing resources; otherwise, the comparison results will not be credible. Recent e-commerce studies have explored deep-learning and transformer-based approaches to review sentiment, including hybrid CNN–BiLSTM designs [8, 23]. Transformer architectures are based on attention mechanisms [27], while pretrained language models such as RoBERTa provide another contextual representation approach [19].

D. Class Imbalance and Evaluation

In user-submitted review datasets, the proportion of positive reviews is often extremely high, leading to an overall skewed data distribution. In this case, using only accuracy to measure a model's performance can easily be misleading; even if a model only classifies all samples into the most frequent category, it can still calculate a very high accuracy relying on the natural skew of the data, which cannot reflect its real ability to identify minority classes. Surveys of deep learning under class imbalance emphasise that the majority class can obscure minority-class errors unless the evaluation is designed carefully [16].

Metrics such as Macro-F1, MCC, and precision-recall are more suitable for evaluating the recognition performance of small minority classes, a conclusion drawn from [6]. MCC is a useful complementary measure for binary classification [10], while precision–recall analysis is especially informative when the positive and negative classes are highly imbalanced [6].

In addition, the model's classification threshold cannot be directly fixed at 0.5; it must be tuned using data from the validation set to select the most appropriate value. Threshold selection should be treated as a validation-stage decision rather than assumed to be optimal at 0.5; probability calibration is also a relevant consideration for neural classifiers [14].

III. Materials and Methods

A. Dataset Provenance and Audit

The study used the public Consumer Reviews of Amazon Products dataset (1429_1.csv) from Kaggle and GitHub, containing reviews of Amazon-branded products. The file's SHA-256 checksum is 53c7098985159f463b1591762f5093f4adbe8e9fc010ca5d90a6ffb0908c95b8. The dataset is the publicly documented Datafiniti Consumer Reviews of Amazon Products collection [11]; the audit details and checksum recorded here are intended to support transparent reuse [9]. Table I summarises the raw-data audit, exclusions, final class counts, and product coverage used in the experiments.

TABLE I. Raw-Data Audit and Label-Construction Summary

Audit item	Value	Operational definition
Raw records	34,660	Rows in the downloaded CSV
Variables	21	Product, reviewer, rating, text, date, and source fields
Missing ratings	33	Removed before label construction
Missing review text	1	Removed before modelling
Three-star reviews	1,499	Excluded as ambiguous for binary polarity
Exact duplicates removed	65	Duplicates based on cleaned title-plus-body text
Final binary corpus	33,062	Used for splitting and evaluation
Positive class	32,252 (97.55%)	Ratings 4-5
Negative class	810 (2.45%)	Ratings 1-2
Unique product IDs	39	Used in product-held-out robustness analysis

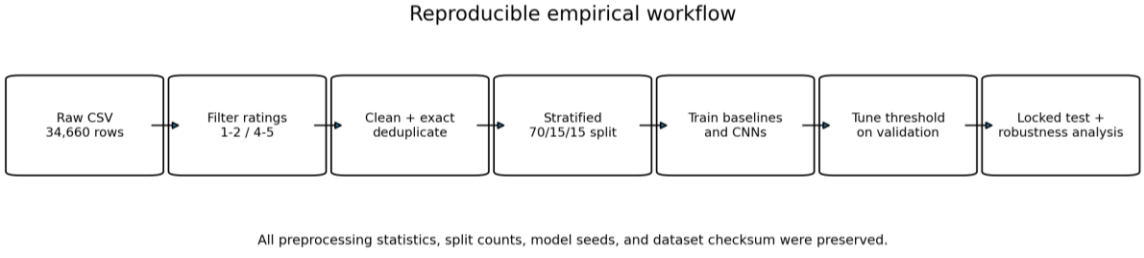

Fig. 1. End-to-end empirical workflow used in the revised study.

B. Label Construction and Text Preprocessing

All user star ratings were first converted into a binary classification; that is, only two categories, positive and negative, were used for processing. 1-2 star negative reviews were classified as negative and uniformly labeled as 0; 4-5 star positive reviews were classified as positive and uniformly labeled as 1; all neutral reviews with 3 stars were excluded from the dataset to be used and did not participate in any subsequent calculations. Next, the text content of each review was processed: the title and body of each review were first concatenated into a single block, then all uppercase letters in the concatenated text were converted to lowercase, after which HTML code, web links embedded in the text, and completely duplicate review entries were removed. During the cleaning process, only basic punctuation marks were retained, as well as negation words that can fully convey emotions and cannot be arbitrarily deleted, such as words like "not" and "never", which were kept untouched. After organizing all usable reviews in this way, the average length of each review was 34 words. These texts were then input into the CNN model, and all sequences input into the model had to be uniformly adjusted to 120 tokens; excessively long sequences had their excess parts truncated from the end, while overly short sequences were padded with placeholder tokens to reach the required length. The vocabulary list, which had a total of 2860 tokens, was generated solely using the split training dataset, and no content from other datasets was used, to avoid information leakage from the test data that would interfere with the model's performance. The rating-to-polarity mapping is an operational proxy rather than a human-annotated sentiment label, so its limitations should be considered when interpreting results [9, 21]. The end-to-end processing and evaluation stages are outlined in Fig. 1, while Table II reports the exact partition counts.

C. Train, Validation, and Test Design

All duplicate reviews were first deleted, and the cleaned corpus was split into three non-overlapping groups using stratified random sampling. The first group was the training set, accounting for 70% of the total, with a total of 23143 reviews, which was the core data used for the model to learn classification rules. The second group was the validation set, accounting for 15%, with 4959 reviews; the third group was a dedicated held-out test set, also accounting for 15%, with 4960 reviews. This set of data never participated in any training process throughout and was only used to ultimately test the model's real performance.

The validation set had two functions: first, to help the model terminate ineffective training in advance—if the model failed to make progress on the validation set across consecutive training rounds, training was stopped in time to avoid wasting computing power and causing overfitting; second, to screen the classification decision threshold, that is, to set a standard for how high the classification probability output by the model must be to count as a positive review, and how low it must be to count as a negative review. This threshold was first adjusted to the most suitable state on the validation set.

In the original reviews, the number of positive reviews was far larger than that of negative reviews, and the large gap in the quantity of the two types of samples would affect the model's learning effect. To balance this quantity gap, when training the CNN model, we actively reduced the number of positive reviews used, adjusting the ratio of positive to negative reviews used during training to 4:1. However, the original proportion of positive and negative samples was retained in both the validation set and the test set without any modifications, so that the model's real capability could be tested with data closest to real-world scenarios.

TABLE II. DATASET PARTITIONS AND CLASS COUNTS

Partition	Total	Negative	Positive	Purpose
Training	23,143	567	22,576	Model fitting and TF-IDF vocabulary
Validation	4,959	121	4,838	Early stopping and threshold selection
Test	4,960	122	4,838	Locked final evaluation
CNN training subset	2,835	567	2,268	All negative training examples plus four sampled positives per negative

As for the sparse linear baseline comparison model, we used the complete, unmodified full training set, without actively deleting any positive reviews. Instead, the quantity gap between the two types of samples was balanced by assigning different weights to samples of different categories.

D. Models

A total of five models' classification effects were tested in this study, starting with the most basic ones. The first was the majority baseline, which served as a benchmark comparison. The rule of this model was very simple: it directly predicted all input samples as belonging to the positive category, to compare its performance with the more complex models that followed. The next two were traditional machine learning models, namely Logistic Regression and linear SVM. These two models adopted the same text preprocessing rules: both used TF-IDF vectors to convert text into numerical features recognizable by the model, with the extraction scope covering only unigrams of single words and bigrams of two-word combinations, retaining a maximum of 50000 feature values. At the same time, balanced weights were added to samples of different categories to solve the problem of uneven numbers of positive and negative samples in the data. The fourth was a single-kernel CNN model, whose specific configuration was: text was converted into vectors processable by the model using 64-dimensional word embeddings; 64 convolutional filters with a width of 5 were set; all text features first underwent global max pooling, then were connected to a fully connected layer with 64 units; a dropout mechanism was added in between to prevent the model from overfitting; and finally, the classification result was output

through the softmax function. The last was a multi-kernel CNN model (Fig. 2), which set up three completely parallel convolutional branches, with the convolutional widths of the three branches being 3, 4, and 5, respectively. Each branch was equipped with 32 independent convolutional filters. After each branch processed its own features, it first completed the pooling operation; then the pooling outputs of all branches were concatenated and sent to the subsequent classification layer to complete the classification. The CNN implementations use dropout as a regularisation method [25] and AdamW-style decoupled weight decay [20], while Adam is a widely used adaptive optimisation method for neural networks [18]. The full settings, including feature extraction, CNN dimensions, optimisation, and random seeds, are summarised in Table III to make the comparison easier to reproduce [18, 20, 25].

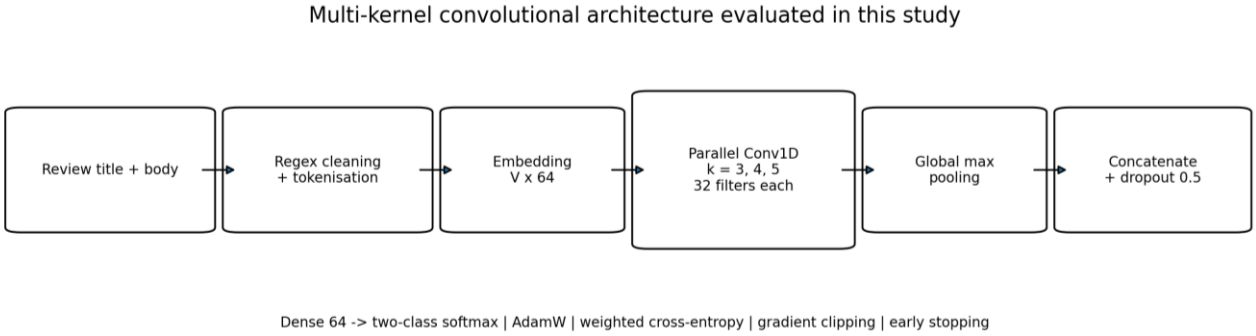

Fig. 2. Multi-kernel CNN architecture implemented for the raw-data experiment.

TABLE III. MODEL AND TRAINING CONFIGURATION

Component	Configuration	Control
TF-IDF	1-2 grams; max 50,000; min df 2; sublinear TF	Training text only
Logistic regression	Balanced class weights; liblinear; max 2,000 iterations	Validation-tuned threshold
Linear SVM	Balanced class weights; linear margin classifier	Validation-tuned decision threshold
Embedding	64 dimensions; trainable; padding index 0	CNNs
Single-kernel CNN	64 filters; width 5; global max pooling	207,874 parameters
Multi-kernel CNN	32 filters each for widths 3, 4, 5; concatenation	214,050 parameters
CNN optimiser	AdamW; learning rate 0.001; weight decay 0.0001	Gradient norm clipped at 1.0
CNN training	Batch 256; maximum 8 epochs; patience 2; dropout 0.5	Seeds 13, 42, 97
Sequence length	120 tokens	Covers more than 95% of reviews without truncation

E. Threshold Selection and Metrics

We first optimized the classification decision threshold on the validation set to maximize the macro-F1 metric, then applied this set of tuned threshold parameters to the separately held-out test set. All performance results of the CNN model were aggregated after running with three different random seeds. The evaluation metrics used included accuracy, macro-F1, weighted F1, MCC, ROC-AUC, negative PR-AUC, as well as metrics for each individual category. We paid particular attention to the performance of the negative class, which had a small sample size; that is, the minority class. We also conducted a leave-one-product-out analysis, splitting the data by product ID: the model was trained on data from 28 product IDs and tested on data from 11 remaining product IDs that had never appeared in training.

IV. RESULTS

A. Exploratory Data Analysis

The dataset was highly imbalanced with a 97.55% positive prevalence. Consequently, as per Fig. 3, the majority baseline reached 97.54% accuracy but had zero negative recall, a 0.494 macro-F1, and 0.000 MCC. This illustrates why accuracy alone is insufficient for imbalanced classification, consistent with prior analyses of MCC and precision–recall evaluation [10, 6].

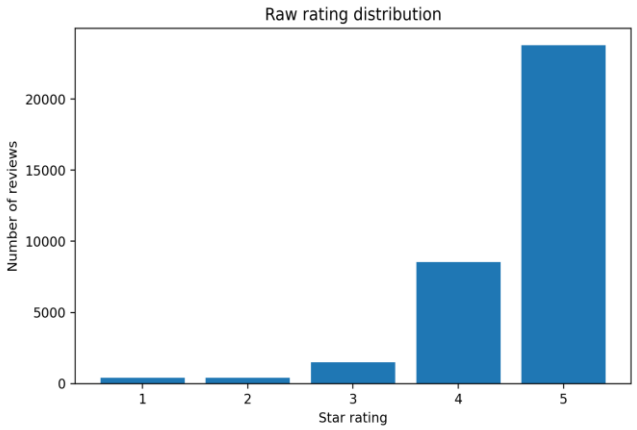

Fig. 3. Distribution of star ratings in the raw 34,660-record dataset.

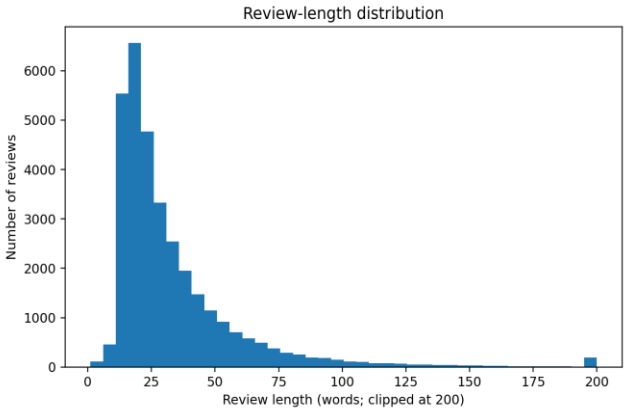

Fig. 4. Review-length distribution after cleaning; values are clipped at 200 words for visualisation.

As shown in Fig. 4, Review lengths were right-skewed (median ≤ 24 words, 95th percentile = 84 words). A 120-token sequence length retained most text while limiting computational costs and the impact of unusually long reviews.

B. Main Test-Set Performance

Fig. 5 displays Accuracy, F1, and MCC. We recorded the performance of all tested models with scores based on unified dimensions; each entry first clearly marks the model name and the number of actual model runs, then lists the values of five core metrics used to evaluate classification capability: accuracy, macro-F1, Matthews Correlation Coefficient (MCC), ROC-AUC, and the negative class precision-recall AUC specifically used to measure the ability to recognize negative classes. The use of macro-F1 and MCC alongside accuracy follows recommendations to report class-sensitive and correlation-based measures rather than accuracy alone [10, 16].

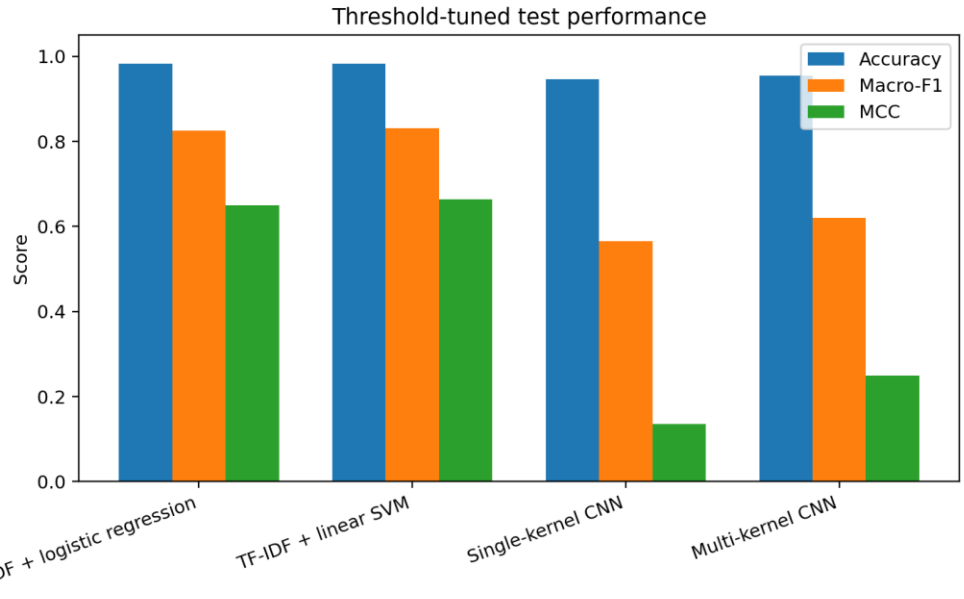

Fig. 5. Accuracy, macro-F1, and MCC after validation-based threshold selection.

The first model tested was the most basic majority baseline model; this model was only run once, and it achieved an accuracy of 97.54%, which seemed very high. However, other metrics that measure classification balance immediately exposed its nature—it only blindly predicted the class that accounted for the absolute majority in quantity and had no real capability at all. Its macro-F1 was 0.494, MCC was 0.000, ROC-AUC was 0.500, and even though its negative class PR-AUC was only 0.025, a dismal value.

Next tested was the TF-IDF model paired with logistic regression; this model was also only run once, and its performance in all aspects improved by a large margin. Its accuracy rose to 98.27%, and the remaining metrics also increased accordingly: macro-F1 0.825, MCC 0.650, ROC-AUC 0.970, negative class PR-AUC 0.687. All sensitive metrics used to test whether a model can recognize minority samples left the previous baseline model that could only guess the majority class far behind.

After that was the TF-IDF model paired with linear SVM; it was also only run once, and its scores were even better than the previous logistic regression model. It achieved an accuracy of 98.33%, with the remaining metrics being macro-F1 0.832, MCC 0.664, ROC-AUC 0.969, negative class PR-AUC 0.742, ranking first overall among all tested models.

Then came the testing of two CNN models; unlike the previous models, both were run three times with independent training random seeds, each with different initial conditions, and their final performance was much weaker. All metrics were formatted as the average of the three runs plus the standard deviation, to reflect the fluctuation range of the results. First was the Single-kernel CNN; its accuracy was 94.56±0.88%, macro-F1 0.565±0.047, MCC 0.135±0.094, ROC-AUC 0.786±0.071, negative class PR-AUC 0.101±0.060, and the standard deviation listed indeed confirms that the results from the three runs with different seeds consistently showed fluctuations.

After that was the Multi-Kernel CNN, it also calculated the average of the results from three random seeds. Compared with the previous Single-kernel CNN, the gap narrowed significantly, but it still lagged far behind the previous two sparse linear models. Its scores were accuracy 95.53±1.01%, macro-F1 0.620±0.007, MCC 0.249±0.019, ROC-AUC 0.866±0.011, negative class PR-AUC 0.218±0.035.

All these results are summarized in Table IV. Every set of data in Table IV was measured on a fixed closed test set after the model's classification decision threshold was adjusted. All values for CNN models retain the average plus standard deviation format calculated from three independent training seeds, which is consistent with the actual testing situation. In addition, there is a Fig. 5 that specifically plots three core performance metrics—namely accuracy, macro-F1, and MCC. All models included in the figure went through the same set of threshold screening processes based on the validation set, so their scores can be directly compared horizontally, with no issue of inconsistent standards.

TABLE IV. THRESHOLD-TUNED PERFORMANCE ON THE LOCKED TEST SET. CNN VALUES ARE MEAN ± STANDARD DEVIATION ACROSS THREE SEEDS

Model	Runs	Accuracy	Macro-F1	MCC	ROC-AUC	Neg. PR-AUC
Majority baseline	1	97.54%	0.494	0.000	0.500	0.025
TF-IDF + logistic regression	1	98.27%	0.825	0.650	0.970	0.687
TF-IDF + linear SVM	1	98.33%	0.832	0.664	0.969	0.742
Single-kernel CNN	3	94.56 ± 0.88%	0.565 ± 0.047	0.135 ± 0.094	0.786 ± 0.071	0.101 ± 0.060
Multi-kernel CNN	3	95.53 ± 1.01%	0.620 ± 0.007	0.249 ± 0.019	0.866 ± 0.011	0.218 ± 0.035

When all the evaluation results of the fixed test set are compared together, the overall performance of the TF-IDF plus linear SVM model is the best among all models, followed closely by the TF-IDF plus logistic regression model; the gap between all metrics of these two models is very small, and they are neck and neck. Both models belong to the sparse linear architecture, and they outperformed the two types of CNN models by a large margin on all balance- and minority-sensitive metrics designed to prevent the model from biasing toward the majority class that dominates in quantity, with an obvious gap.

Even though every metric produced by each run of the Multi-kernel CNN was better than that of the simpler Single-kernel CNN, its average performance in recognizing negative classes never reached the lowest threshold set by the linear models. On this specific classification task, the gap between the feature extraction capability realized by deep learning and the manually tuned sparse feature representation capability was never closed.

This series of test results provides clear practical evidence: for the specific corpus classification task used in this study, feature representations learned based on CNNs do not automatically outperform carefully tuned sparse representation methods.

C. Class-Specific Performance

The thresholds used in this set of tests shown in Table V are not arbitrarily set: we first tested on the validation dataset to select the most appropriate critical value, and all scores listed in the table are test set results measured using this screened threshold. Six sets of values measuring classification performance are calculated for each model. The first three sets are used to evaluate the model's ability to identify negative-class samples, and the last three sets are used to evaluate its ability to identify positive-class samples.

TABLE V. CLASS-SPECIFIC TEST PERFORMANCE AFTER VALIDATION-BASED THRESHOLD SELECTION

Model	Neg. precision	Neg. recall	Neg. F1	Pos. precision	Pos. recall	Pos. F1
TF-IDF + logistic regression	0.638	0.680	0.659	0.992	0.990	0.991
TF-IDF + linear SVM	0.649	0.697	0.672	0.992	0.990	0.991
Single-kernel CNN	0.130 ± 0.078	0.202 ± 0.102	0.158 ± 0.089	0.980 ± 0.003	0.964 ± 0.007	0.972 ± 0.005
Multi-kernel CNN	0.232 ± 0.044	0.325 ± 0.084	0.263 ± 0.014	0.983 ± 0.002	0.971 ± 0.012	0.977 ± 0.005

Among all models, the linear SVM achieves the best performance on negative classes, with the most balanced coordination between its two core metrics: Model Neg precision and Model Neg recall. The model ranking next is the logistic regression model combined with TF-IDF, whose negative-class performance is only inferior to that of the linear SVM.

The average Model Neg recall of the Multi-kernel CNN is higher than its average Model Neg precision. This characteristic indicates that it adopts a relatively aggressive alert strategy; it tends to classify more samples into the negative class, just like an alarm system that triggers an alert at the slightest sign of movement. While it rarely misses cases that truly require warning, it also triggers a large number of unnecessary false

alarms. The negative-class precision–recall curves in Fig. 7 are particularly important because the dashed reference line marks the negative-class prevalence; PR analysis is more informative than ROC analysis under severe imbalance [6].

In the corpus we used to train and test the models, the number of positive and negative samples differs greatly, with a severely imbalanced distribution. For this reason, the positive-class F1 values of all tested models exceed 0.97.

Fig. 6 compares ROC behaviour across the sparse baselines and the representative multi-kernel CNN. The curves should be interpreted alongside Table IV and the negative-class PR-AUC results, because ROC-AUC alone can look favourable when the minority class is rare [6, 10].

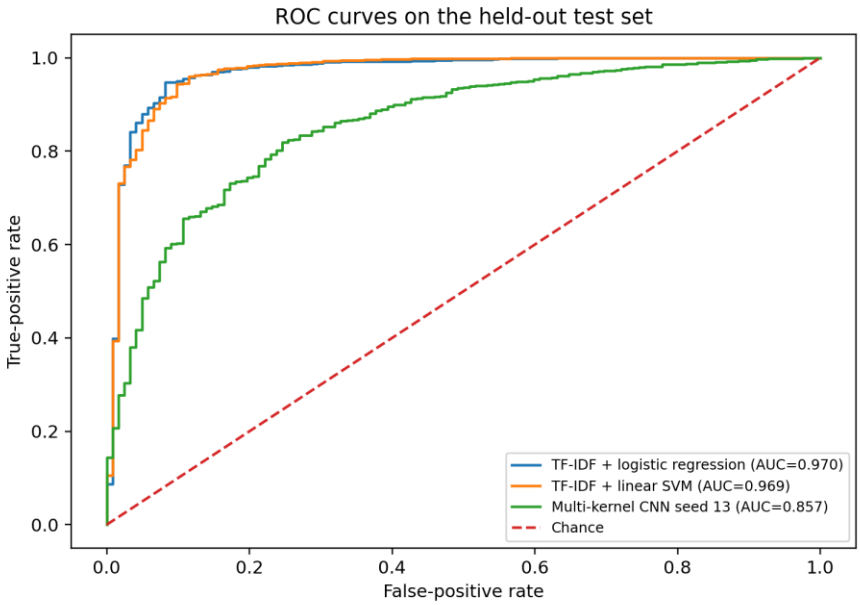

Fig. 6. ROC curves for the sparse baselines and representative multi-kernel CNN.

Fig. 7 focuses directly on detection of negative reviews. The dashed prevalence line provides a baseline for interpreting the precision–recall curve; the linear SVM's stronger negative-class PR-AUC in Table IV indicates better minority-class ranking on this test set [6].

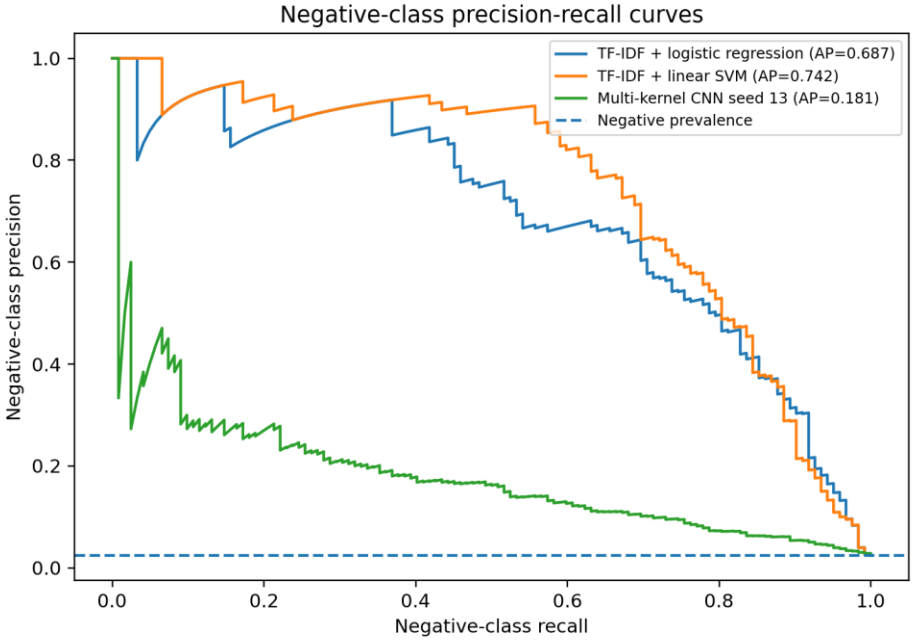

Fig. 7. Negative-class precision-recall curves; the dashed line represents negative prevalence.

D. CNN Stability Across Random Seeds

The single-kernel CNN showed in Table VI substantial seed sensitivity with macro-F1 ranging from 0.536 to 0.619. The multi-kernel CNN was more stable, with macro-F1 between 0.612 and 0.625 and MCC between 0.229 and 0.267. Validation-selected thresholds varied considerably, especially for the multi-kernel architecture (approximately 0.20 to 0.36), confirming that a fixed 0.5 threshold would be inappropriate after weighted and down-sampled training.

Fig. 8 presents the training and validation trajectories for a representative multi-kernel CNN run. These curves are useful for checking convergence and potential overfitting, but they should be read together with the seed-level test results in Table VI rather than treated as evidence of generalisation on their own [21, 24].

TABLE VI. SEED-LEVEL CNN RESULTS ON THE LOCKED TEST SET

CNN	Seed	Threshold	Accuracy	Macro-F1	MCC	Neg. PR-AUC	Neg. F1
Single-kernel CNN	13	0.297	95.54%	0.619	0.243	0.170	0.261
Single-kernel CNN	42	0.497	94.29%	0.536	0.076	0.071	0.102
Single-kernel CNN	97	0.474	93.85%	0.539	0.086	0.062	0.111
Multi-kernel CNN	13	0.356	95.46%	0.612	0.229	0.181	0.247
Multi-kernel CNN	42	0.198	96.57%	0.625	0.250	0.251	0.267
Multi-kernel CNN	97	0.292	94.56%	0.623	0.267	0.222	0.274

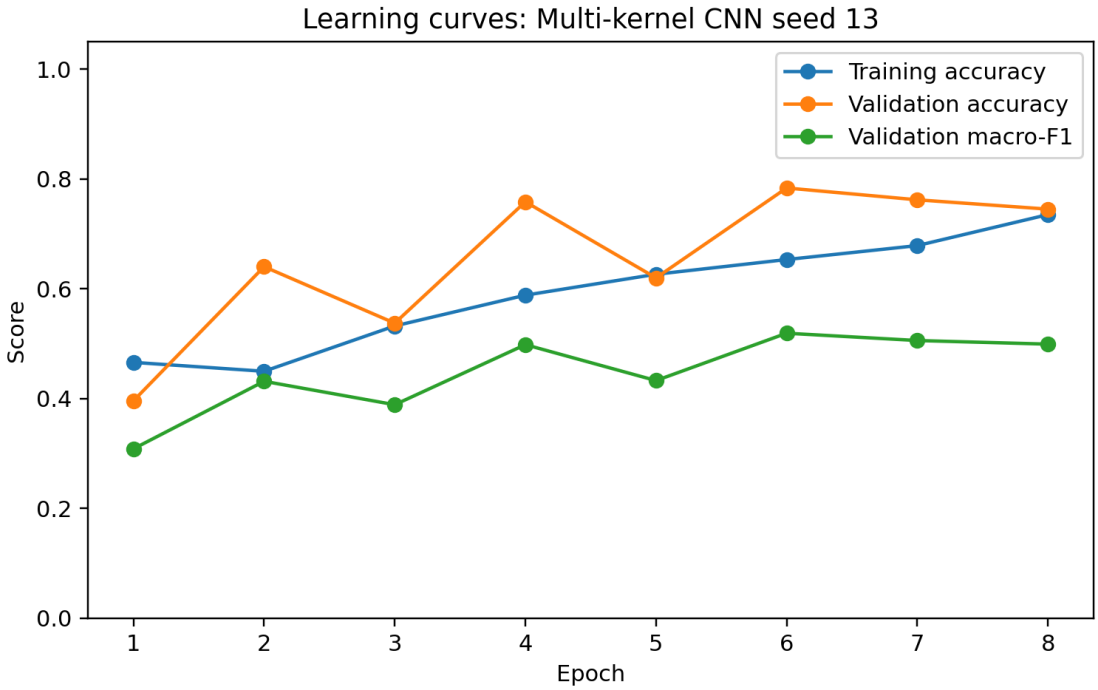

Fig. 8. Training and validation curves for a representative multi-kernel CNN run.

Fig. 9 gives the class-level error counts for the representative threshold-tuned multi-kernel CNN run. It makes false positives and false negatives visible, complementing the aggregate and class-specific metrics in Tables IV–VI.

The seed-level variation also reinforces the need for repeated runs and transparent statistical testing when comparing NLP systems [12].

Comparing these results together as available in Fig. 8 and 9, it can be observed that the Single-kernel CNN is highly sensitive to changes in random seeds. When a different random seed is used to rerun the model, the fluctuation range of all results is extremely large. In contrast, even when tested with different random seeds, the Multi-kernel CNN maintains far more stable performance across all metrics. In addition to differences in model performance, the classification thresholds selected via the validation phase in each test also fluctuate greatly, and the threshold fluctuation of the multi-kernel model is even more pronounced than that of the single-kernel model. This result confirms one point: even after model training is completed via class weighting and downsampling, a fixed value of 0.5 should still not be used as the classification threshold, as a fixed threshold cannot adapt to the classification requirements after this type of training.

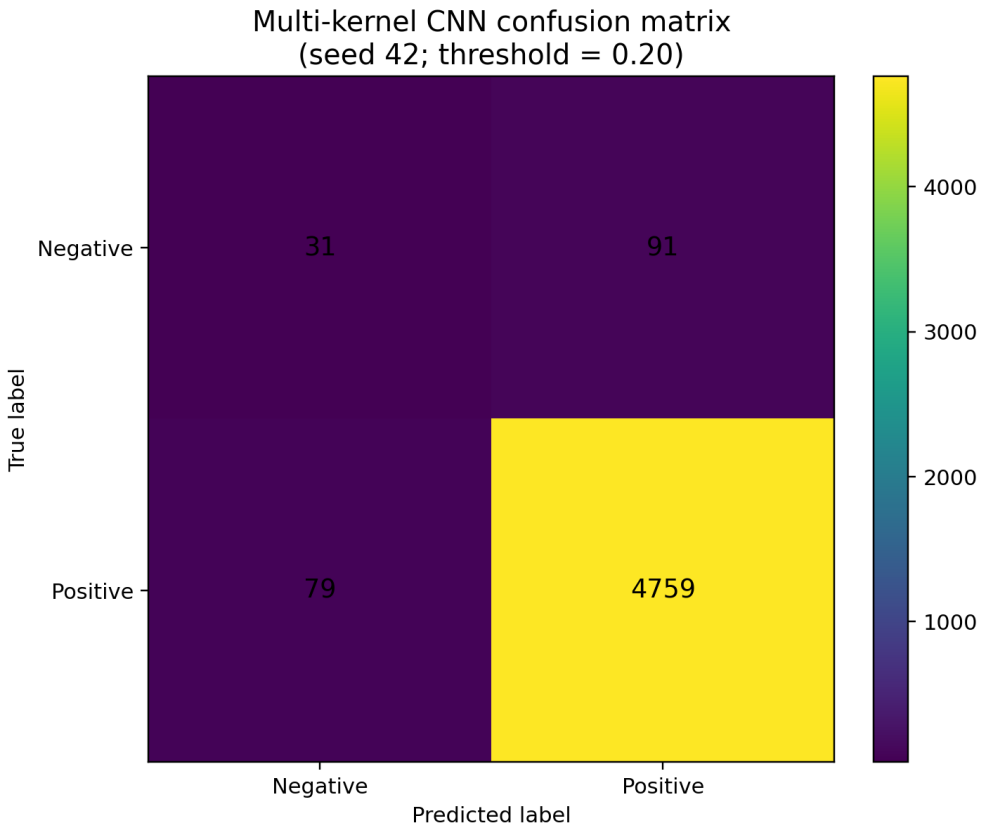

Fig. 9. Confusion matrix for the best representative threshold-tuned multi-kernel CNN run.

E. Product-Held-Out Robustness

This product-held-out evaluation complements random stratified testing by probing generalisation to unseen product domains, a relevant concern in consumer-review modelling [7, 11]. As shown in Table VII, performance decreased under product-held-out evaluation, as expected, but the linear models remained robust. The SVM achieved 98.25% accuracy, macro-F1 of 0.796, MCC of 0.594, and negative PR-AUC of 0.644. Logistic regression achieved 96.82% accuracy, macro-F1 of 0.759, and MCC of 0.546. The reduction relative to random stratified testing indicates that product-specific language contributes to performance, but it does not explain the full advantage of sparse linear features.

TABLE VII. PRODUCT-HELD-OUT TEST PERFORMANCE ON 11 PRODUCT IDS UNSEEN DURING TRAINING

Model	Accuracy	Macro-F1	MCC	ROC-AUC	Neg. PR-AUC	Neg. F1
TF-IDF + logistic regression	96.82%	0.759	0.546	0.969	0.631	0.535
TF-IDF + linear SVM	98.25%	0.796	0.594	0.971	0.644	0.600

This test is an evaluation that withholds entire products from the training set, meaning all products used for testing were never seen by the model during training. Consistent with expectations, in this test scenario of completely new products, the performance of all models declined, but linear models still produced stable and reliable results. This performance decline illustrates one point: the exclusive terms that only apply to specific products can indeed help models achieve better results in conventional tests, but this advantage cannot fully explain the inherent performance lead of sparse linear features; even after removing the score boost brought by exclusive terms, the advantage of sparse linear features persists.

F. Error Analysis

High-confidence CNN errors were dominated by mixed-polarity positive reviews. Examples included four-star reviews containing phrases such as “limited amount of local storage”, “not what I really needed”, “not as good as an iPad”, or “slow Wi-Fi/browsing”. These local negative phrases activated convolutional filters even though the overall rating was positive. Conversely, brief negative reviews with generic wording offered little lexical evidence for the minority class. The error pattern shows a construct limitation of rating-derived labels: a single review may praise the product overall while criticising one feature.

Most of the high-confidence misclassifications made by the convolutional neural network (CNN) occurred in positive reviews with mixed emotions. These reviews had an overall positive rating but contained local negative complaints; those negative phrases triggered the convolutional filters in the model, directly skewing the model's judgment. The categories and representative patterns are summarised in Table VIII.

TABLE VIII. QUALITATIVE ERROR TAXONOMY DERIVED FROM HIGH-CONFIDENCE MISCLASSIFICATIONS

Error type	Observed pattern	Interpretation
Mixed polarity	Positive rating with a salient complaint	Aspect-level criticism dominates the local CNN representation
Comparative language	"Not as good as..." followed by acceptable overall judgment	Negation and comparison are locally negative but globally qualified
Expectation mismatch	"Seems okay but not what I needed"	Ambiguous sentiment and product-fit concerns
Short generic review	Very few sentiment-bearing tokens	Insufficient evidence for reliable minority detection
Service versus product	Complaint concerns delivery, account, or ecosystem	Rating does not map cleanly to product sentiment

Another opposite scenario is that some negative reviews are very short and general in content, lacking sufficient emotion-laden words to support classification, so the model cannot gather enough basis to assign them to the corresponding minority categories. All these error cases expose the core flaw of using user ratings as classification labels: a review may praise a product overall but then criticize several specific functions, and a single rating cannot accommodate such complex circumstances.

The error taxonomy in Table VIII groups high-confidence CNN misclassifications into mixed-polarity reviews, comparative language, unmet expectations, short generic reviews, and service-versus-product complaints. These categories help explain why rating-derived binary labels can conflict with local wording.

V. DISCUSSION

A. Why the Sparse Linear Models Won

A model's applicability cannot be generalized; it depends on the structure of the text data you use for training and whether that structure aligns with the model's characteristics.

The processing method used in this study groups two consecutive words in the text into a single phrase, then applies the TF-IDF method to calculate weights. This processing can accurately capture key phrase features that distinguish different categories. Even in the entire training set, there were only 810 samples belonging to the negative class—under this condition of an extremely small sample size, linear SVMs still delivered solid performance and did not fail due to insufficient negative samples.

In contrast, CNNs operate differently. To learn useful vector features, they require far more data to support the training process. Furthermore, their final performance is heavily dependent on initial settings; even minor changes to the initial training conditions lead to large fluctuations in the results, making them highly unstable.

B. Effect of Severe Imbalance

Relying on accuracy to evaluate a model's performance can be highly misleading. If one takes no action and simply classifies all samples to be judged into the majority class, failing to identify a single negative review, one can still achieve an accuracy rate of 97.54%. This type of data is what we commonly refer to as a severely imbalanced dataset. Evaluating it only using overall accuracy completely fails to reflect the model's true capability. To evaluate such datasets, one must use metrics specifically designed for individual classes, such as Macro-F1, MCC, and negative PR-AUC. Only these metrics can accurately measure the model's true performance. These observations align with research showing that MCC can reveal performance differences hidden by accuracy or F1 alone [10].

C. Thresholding as Part of the Model

To set a qualified threshold for the model's classification results, one must first test on a reserved validation set to select the most appropriate threshold. The success of this step directly determines whether the model can properly identify the minority class with a small sample size, making it a critical, non-skippable link in the entire modeling workflow. Once the model is deployed for use, one must never set the threshold to the default value of 0.5 arbitrarily; instead, it must be adjusted against the various costs in real business scenarios. Do you prioritize precision because you fear the model misclassifies samples that should not belong to this class, or do you prioritize recall to avoid missing any samples that do belong to this class? All adjustments must align with the actual needs of the business.

D. Relationship to Contemporary Deep Models

Models such as Transformers can indeed capture broader contextual information from text, but they consume far more computational resources and cannot avoid interference caused by dirty data with incorrect annotations. Although transformer models such as BERT and RoBERTa can model contextual dependencies [4, 19], more complex architectures should be compared against strong baselines; hybrid deep-learning approaches have been studied for product-review sentiment [8, 23].

No deep learning architecture inherently outperforms simple baseline models unless we provide it with a suitable amount of data and conduct rigorous, targeted evaluations. Sparse models like linear SVMs, which are fast, logically transparent, and robust, remain the preferred entry-level tool for text classification tasks to this day.

VI. PRACTICAL AND MANAGERIAL IMPLICATIONS

- Independently evaluate the precision and recall of the negative class; when the number of complaint samples

is very small, overall accuracy will only breed unfounded confidence. If you only fixate on the overall accuracy figure, you may easily mistake the model's performance as sufficient, never discovering that its ability to identify minority negative samples is actually very poor, and miss the chance to detect and solve problems.

- The TF-IDF linear model trains quickly, performs excellently on minority class samples, and can explain its classification logic. Even for samples from the small-sized class, it can capture key features to make accurate judgments. More importantly, you can clearly see which text pieces it relies on to reach its classification conclusion, unlike black-box models that only output a result without explaining why.
- Calibrate CNN output probability values and link classification thresholds to the costs of actual workflows. Do not apply a one-size-fits-all approach with the default 0.5 threshold; instead, adjust it to the value most suitable for the current workflow by incorporating the real costs of misclassification and missed classification in business scenarios, balancing business costs and the model's recognition performance.
- For mixed reviews that carry both positive and negative sentiments, do not force a single label on them. Switch to an attribute-based analysis model, or assign them for manual review, to identify which specific functions users have criticized. Forcing binary positive/negative labels on such complexly toned reviews will only obscure the specific problems raised by users and will never help you locate product details that require improvement.
- Continuously monitor the proportion of different categories, vocabulary drift, and the model's performance across various subgroups to ensure the system remains reliable. The ratio of sample categories in your dataset will not stay constant forever, and commonly used vocabulary will also change gradually over time. If you do not regularly track these changes and adjust the model, the system's recognition accuracy will decline before long, and it will fail to operate stably and reliably.

VII. Limitations and Future Research

Limitations include a restricted corpus (Amazon-branded products only), noisy rating-derived labels, lack of transformer testing due to compute limits, a very small minority class, and limited cross-product generalization. Future research should evaluate diverse, multilingual datasets with human annotations, test calibrated transformers, and expand to aspect-level sentiment and temporal drift analysis. Future evaluation could include the multilingual Amazon Reviews Corpus [17] and compare calibrated transformer baselines, including BERT and RoBERTa [4, 19], while following transparent data-statement and model-reporting practices [9, 21].

VIII. Conclusion

Analyzing 33,062 highly imbalanced reviews (97.55% positive), the TF-IDF linear SVM outperformed CNNs across minority-sensitive measures, achieving 98.33% accuracy, 0.832 macro-F1, and 0.742 negative PR-AUC. Credible sentiment research demands transparent data processing, strong baselines, class-sensitive metrics, and validation-based thresholding. Ultimately, model complexity must be driven by evidence rather than assumed superiority.

Declarations

A. Author Contributions

Conceptualisation: Shamik Palit and Preetha V. K.; methodology: Shamik Palit, Dhanya Sudarsan, and Jimy; software and formal analysis: Shamik Palit; validation: Preetha V. K., Dhanya Sudarsan, Jimy, and Santhuana Sedhumadhavan; writing - original draft: Shamik Palit; writing - review and editing: all authors. The final contribution statement should be confirmed by all authors before submission.

B. Funding

No external funding was declared for this study.

C. Conflict of Interest

The authors declare no conflict of interest.

D. Data Availability

The raw Consumer Reviews of Amazon Products dataset is publicly available through Datafiniti on Kaggle. The analysed file was 1429_1.csv. To support reproducibility, the study records the file checksum, preprocessing decisions, split counts, and complete model configuration. Redistribution should follow the source dataset licence and terms. The dataset source is cited as Datafiniti's public Kaggle collection [11]; the reproducibility record in Appendix A provides the checksum, split protocol, seeds, and threshold rule. Table IX consolidates the key reproducibility metadata.

E. Code Availability

Python scripts used for preprocessing, TF-IDF baselines, PyTorch CNN training, threshold selection, evaluation, and figure generation accompany the revised manuscript package.

F. Ethics Statement

The study used an existing public dataset and did not recruit human participants or access private accounts. Reviewer identifiers were not used for modelling, and identifiable review text should not be unnecessarily reproduced in publications or dashboards.

References

[1] Kim, Y. (2014). Convolutional neural networks for sentence classification. Proceedings of EMNLP, 1746-1751. https://doi.org/10.3115/v1/D14-1181

[2] Turney, P. D. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. Proceedings of ACL, 417-424. https://doi.org/10.3115/1073083.1073153

[3] Johnson, R., & Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. Proceedings of NAACL-HLT, 103-112. https://doi.org/10.3115/v1/N15-1011

[4] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT, 4171-4186. https://doi.org/10.18653/v1/N19-1423

[5] Liu, Y., Lu, J., Yang, J., & Mao, F. (2020). Sentiment analysis for e-commerce product reviews by a deep learning model of BERT-BiGRU-Softmax. Mathematical Biosciences and Engineering, 17(6), 7819-7837. https://doi.org/10.3934/mbe.2020398

[6] Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE, 10(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

[7] Alantari, H. J., Currim, I. S., Deng, Y., & Singh, S. (2022). An empirical comparison of machine learning methods for text-based sentiment analysis of online consumer reviews. International Journal of Research in Marketing, 39(1), 1-19. https://doi.org/10.1016/j.ijresmar.2021.10.011

[8] Bellar, O., Baina, A., & Ballafkih, M. (2024). Sentiment analysis: Predicting product reviews for e-commerce recommendations using deep learning and transformers. Mathematics, 12(15), 2403. https://doi.org/10.3390/math12152403

[9] Bender, E. M., & Friedman, B. (2018). Data statements for natural language processing: Toward mitigating system bias and enabling better science. Transactions of the Association for Computational Linguistics, 6, 587-604. https://doi.org/10.1162/tacl_a_00041

[10] Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient over F1 score and accuracy in binary classification evaluation. BMC Genomics, 21, 6. https://doi.org/10.1186/s12864-019-6413-7

[11] Datafiniti. (2017). Consumer Reviews of Amazon Products [Dataset]. Kaggle. https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products

[12] Dror, R., Baumer, G., Shlomov, S., & Reichart, R. (2018). The hitchhiker's guide to testing statistical significance in natural language processing. Proceedings of ACL, 1383-1392. https://doi.org/10.18653/v1/P18-1128

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[14] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. Proceedings of ICML, 1321-1330.

[15] Hou, Y., Li, J., He, Z., Yan, A., Chen, X., & McAuley, J. (2024). Bridging language and items for retrieval and recommendation. arXiv:2403.03952.

[16] Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. Journal of Big Data, 6, 27. https://doi.org/10.1186/s40537-019-0192-5

[17] Keung, P., Lu, Y., Szarvas, G., & Smith, N. A. (2020). The Multilingual Amazon Reviews Corpus. Proceedings of EMNLP, 4563-4568. https://doi.org/10.18653/v1/2020.emnlp-main.369

[18] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations.

[19] Liu, Y., Ott, M., Goyal, N., et al. (2019). RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692.

[20] Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. International Conference on Learning Representations.

[21] Mitchell, M., Wu, S., Zaldivar, A., et al. (2019). Model cards for model reporting. Proceedings of the Conference on Fairness, Accountability, and Transparency, 220-229. https://doi.org/10.1145/3287560.3287596

[22] Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135. https://doi.org/10.1561/1500000011

[23] Rana, M. R. R., Nawaz, A., Ali, T., Alattas, A. S., & AbdElminaam, D. S. (2024). Sentiment analysis of product reviews using transformer-enhanced 1D-CNN and BiLSTM. Cybernetics and Information Technologies, 24(3), 112-131. https://doi.org/10.2478/cait-2024-0028

[24] Ribeiro, M. T., Wu, T., Guestrin, C., & Singh, S. (2020). Beyond accuracy: Behavioral testing of NLP models with CheckList. Proceedings of ACL, 4902-4912. https://doi.org/10.18653/v1/2020.acl-main.442

[25] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15, 1929-1958.

[26] Tan, K. L., Lee, C. P., & Lim, K. M. (2023). A survey of sentiment analysis: Approaches, datasets, and future research. Applied Sciences, 13(7), 4550. https://doi.org/10.3390/app13074550

[27] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

[28] Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. Advances in Neural Information Processing Systems, 28, 649-657.

[29] Zhao, H., Liu, Z., Yao, X., & Yang, Q. (2021). A machine learning-based sentiment analysis of online product reviews with a novel term weighting and feature selection approach. Information Processing & Management, 58(5), 102656. https://doi.org/10.1016/j.ipm.2021.102656.

APPENDIX

A. Reproducibility Record

TABLE IX. REPRODUCIBILITY METADATA FOR THE EMPIRICAL RERUN

Item	Recorded value
Raw file	amazon_reviews_raw.csv (source filename 1429_1.csv)
SHA-256	53c7098985159f463b1591762f5093f4adbe8e9fc010ca5d90a6ffb0908c95b8
Software	Python 3; pandas 2.2.3; scikit-learn 1.8.0; PyTorch 2.10.0+cpu
Random seeds	13, 42, 97 for CNNs; 42 for data splitting and linear baselines
Split	70% train, 15% validation, 15% test; stratified by binary label
Primary threshold rule	Maximise validation macro-F1; apply selected threshold to test once
CNN training sample	All negative training reviews plus four positive reviews per negative
Saved outputs	Split manifest, seed-level metrics, aggregate metrics, model checkpoints, figures, error file