[CS 7643] Classifying Useful Tweets during Disaster

Jinwoo Jeong
jjeong342@gatech.edu

Jade Kim
jkim4163@gatech.edu

Abstract

The ability to classify and prioritize tweets during crises is essential for emergency responders who must quickly sort through large volumes of information. Many studies have explored Twitter-based classification, but existing category schemes are often either too broad (informative vs. non-informative) or too granular, making practical prioritization difficult. In this project, we redefine the task as a three-class problem (time_critical, support_and_relief, non_informative) and examine the performance of three deep learning models: a CNN built from scratch, a custom Transformer, and a fine-tuned pretrained Transformer. In this report, we describe how each model was implemented and trained, identify the best-performing model through quantitative evaluation, provide qualitative analysis for deeper insight, and discuss limitations and directions for future work.

1. Introduction

In the event of a disaster, valuable real-time information can be found on social media platforms such as Twitter (now known as X) [14]. Time-critical information, such as reports of injured or trapped people, is essential for minimizing casualties. Information about food, water, medical supplies, and shelter is also important for assisting survivors and supporting relief efforts. However, identifying useful information within the overwhelming stream of online messages is difficult and time-consuming because it is mixed with many irrelevant or emotional reactions that do not contribute to response efforts [3].

Researchers have therefore built systems that analyze social media data during crises. Much of this work has focused on Twitter, using it for early disaster detection, situational awareness, and extracting actionable information [2, 14]. A major line of research involves tweet classification, enabled by large annotated datasets such as CrisisBench [2], CrisisLex [11], CrisisNLP [7], and CrisisMMD [1].

Twitter-based classification studies span a wide range of objectives, but many of them fall under two approaches. The first is informativeness classification, which simply distinguishes useful messages from non-useful ones. This binary classification helps filter noise, but it does not tell responders *what* type of help is needed. The second approach is humanitarian category classification, which assigns tweets to detailed label types (e.g., injuries, missing persons, infrastructure damage). This is more descriptive, but can also create confusion when categories overlap or are too numerous. For example, a tweet describing an injured person trapped beneath collapsed debris fits both "injured" and "infrastructure damage," but standard models force only one choice, which can reduce practical usability [2].

To overcome these limitations, we define a three-class classification task based on how tweets can be used during disaster response. Tweets that require immediate action (e.g., reports of injured people or infrastructure damage) should be classified as the **time-critical** class. Tweets useful for post-rescue aid (e.g., food, shelter, medical supply needs) fall under the **support-and-relief** class. Tweets that do not contain any relevant or actionable information (e.g., random online users' reactions to the disaster) are classified as **non-informative**.

To solve this task, multiple deep learning models with different architectures were trained and fine-tuned to perform this three-way classification. We focus on convolutional neural networks (CNN) and transformer-based architectures as they have previously demonstrated strong performance on crisis datasets [2, 10, 15].

If we can successfully train such a classification system, this system will help responders identify urgent information faster, remove distracting content, and allocate resources more effectively. Time-critical posts could assist 911 dispatchers and search-and-rescue teams, while support-and-relief posts could guide donation planning, shelter distribution, and medical supply delivery. In short, this would make real-time crisis information easier to use, which can directly impact human lives.

1.1. Dataset

The CrisisBench dataset is used for this project [2]. The dataset was created to serve as a standard evaluation bench-

mark for crisis tweet classification, addressing the lack of unified and comparable resources in prior work, which aligns well with the goals of our classification task. By consolidating eight pre-existing crisis-related Twitter datasets (e.g., CrisisLex [11], CrisisNLP [7], CrisisMMD [1]), harmonizing labels, and removing duplicates, it enables fair comparison across models and tasks. The primary goal of the dataset is to support robust, reproducible evaluation for both informativeness and humanitarian classification tasks.

For our project, we use the English portion of the CrisisBench dataset for the humanitarian classification task, consisting of 87,455 labeled tweets. The CrisisBench dataset defines 11 humanitarian classes by unifying semantically similar labels across multiple source datasets, merging variants with equivalent meaning (e.g., "Infrastructure damage" and "Infrastructure and utilities" unified as "Infrastructure and utilities damage") [2]. For our three-way classification task, these 11 classes were grouped into three higher-level categories as shown in Table 1.

Mapped class	Original CrisisBench class
time_critical	affected_individual
	caution_and_advice
	displaced_and_evacuations
	infrastructure_and_utilities_damage
	injured_or_dead_people
	missing_and_found_people
support_and_relief	requests_or_needs
	donation_and_volunteering
	response_efforts
non_informative	not_humanitarian
	sympathy_and_support

Table 1: Label Mapping

All tweets were preprocessed prior to modeling. URLs were removed because they do not provide meaningful semantic information [13]. Twitter-specific tokens such as hashtags (e.g., #PrayForParis), usernames, and retweet markers (e.g., RT @BreakingNews) were removed. We also removed symbols, emoticons, invisible and non-ASCII characters, and punctuation, following the preprocessing strategy described in the CrisisBench paper [2]. All text was lowercased to reduce vocabulary dimensionality while maintaining semantic validity [6]. Finally, 110 tweets were discarded because they contained empty strings or duplicated content, resulting in 87,345 remaining samples. Following the original data splits provided by CrisisBench, the final dataset consisted of 61,089 training samples, 8,921 validation samples, and 17,335 test samples. These subsets roughly account for 70%, 10%, and 20% of the total dataset, respectively.

2. Approach

For the experiments, we use CNN and transformer-based models, as they have repeatedly shown strong performance in disaster tweet classification and broader natural language processing tasks [2, 10, 9, 15]. Our objective is to compare different architectures and identify the most effective model for our three-way classification task. To achieve this, we fine-tuned each model by experimenting with hyperparameters, selecting the configuration that yielded the best validation performance. All the problems we have encountered throughout the project are described in Section 5.

2.1. CNN

We implemented the CNN model from scratch using PyTorch. The implemented model begins with an embedding layer initialized with pretrained GloVe word vectors, which are fed into parallel 1D convolutional layers with kernel sizes of 3, 4, and 5 to capture linguistic features across different n-gram spans (see Figure 3 in the appendix). Each convolutional branch is activated with ReLU and followed by max-over-time pooling, after which the resulting feature representations are concatenated and regularized through dropout. A final fully connected layer maps the combined feature vector to the three output categories producing the logits used for classification.

The original CrisisBench paper [2] and its public GitHub repository[1] served as references for our CNN implementation. We did not reuse their source code, since it was written using Keras while our pipeline was developed in PyTorch, and we aimed to introduce several architectural modifications. In particular, unlike the original work that used word2vec embeddings, we used pretrained GloVe vectors which tend to produce more semantically structured word representations by capturing global co-occurrence statistics [12]. We expected this to help the model learn relationships between rare or domain-specific terms often found in crisis-related tweets, providing a stronger initialization for downstream classification.

Overall, this architecture is likely to be successful because it follows the widely adopted TextCNN architecture [8], which has proven effective for short, informal, and often noisy text such as tweets. By applying multiple convolutional filters with different receptive field sizes, the model can detect key local patterns and capture short but meaningful phrases (e.g., mentions of injuries or resource needs) that signal urgent or relevant information, even when the surrounding text is inconsistent or unstructured.

[1] https://github.com/firojalam/crisis_datasets_benchmarks

2.2. Transformers

Transformer-based models were also included to establish a stronger baseline beyond CNNs. Due to their ability to capture long-range dependencies through self-attention, transformer architectures have achieved state-of-the-art results across many NLP tasks, including Twitter classification [2, 10, 9, 15]. While CNNs primarily detect local n-gram features with fixed windows, Transformers model token interactions across the entire sequence, which is advantageous when humanitarian meaning emerges from dispersed cues such as locations, casualties, or resource needs. Since the original CrisisBench reported that pretrained Transformers outperform CNNs on the original 11-way task, similar benefits can be expected in our simplified 3-class setting [2].

We first trained a custom encoder-only Transformer classifier from scratch, without relying on pretrained linguistic representations. Raw tweets were tokenized into subword IDs and embedded via learnable token and positional matrices, whose size is determined by the vocabulary, sequence length, and hidden dimension. These embeddings pass through stacked Transformer encoder blocks, each comprising multi-head self-attention, a position-wise feed-forward layer, and two layer-normalization steps. Each block applies residual connections around both the attention sublayer and the feed-forward sublayer, following the standard Transformer design. The resulting contextual representations are passed through an additional layer-normalization step, then aggregated via masked mean pooling over valid tokens and fed into a two-layer feed-forward classifier with dropout (see Figure 4 in the appendix). As this is a sentence-level task, no decoder is required. The encoder alone suffices for semantic extraction. This configuration allows us to evaluate how effectively a Transformer learns crisis-specific patterns purely through supervised training, independent of large-scale pretraining.

To examine how transfer learning improves tweet classification, we also fine-tune a pretrained Transformer model, DeBERTa-v3 [4], which is the most recent variant of the DeBERTa family [5]. Although the model was not included in the original CrisisBench study, it is recognized for strong contextual representation through disentangled attention and improved pretraining efficiency. By evaluating both the custom Transformer and the pretrained DeBERTa-v3 model, we can directly compare learning from scratch with transfer learning, analyzing whether crisis-specific representations can emerge purely from supervised data and how much additional performance gain large-scale pretraining provides in our three-way task.

2.3. Model Evaluation

All models were optimized using AdamW with cross-entropy loss, consistent with standard multi-class text classification practice. However, the dataset is strongly imbalanced. Approximately 65% of samples are labeled *non_informative*, while each minority class accounts for only about 18% (Table 2). Under such imbalance, accuracy alone becomes a misleading objective because a classifier favoring the majority label may appear to perform well despite poor minority-class recognition. To avoid this bias, we adopted macro **F1 score** as the primary training and model-selection metric.

Training was monitored using validation macro F1-score, and we identified the optimal checkpoint as the epoch immediately before the drop in validation score. Although training was not halted early, this selection strategy allowed us to evaluate each model at its most stable generalization point rather than at the peak of training performance, which may reflect overfitting. For every architecture, we tested multiple hyperparameter configurations and compared results using this validation checkpoint to ensure a fair and capacity-aware assessment.

After identifying the best configuration for each model, evaluation will not rely on a single metric. Because each label carries distinct operational implications in disaster response, we perform a detailed class-wise confusion analysis, breaking predictions into true positives, false positives, false negatives, and true negatives per class. Beyond quantitative counts, we additionally examine misclassified samples qualitatively to identify recurring linguistic patterns, sources of ambiguity, and failure modes that numerical metrics alone cannot fully reveal. This approach allows us to conduct an in-depth analysis of where the models are likely to succeed and where they are likely to fail when applied to real-world crisis situations.

Class	Count	Percent
time_critical	15,363	17.5889%
support_and_relief	15,390	17.6198%
non_informative	56,592	64.7913%
Total	87,345	100%

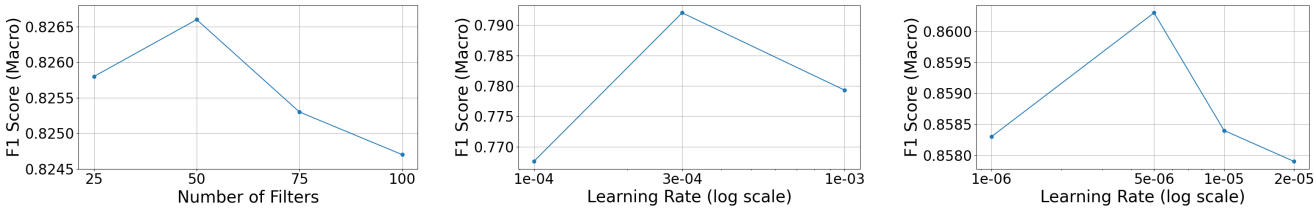
Table 2: Class distribution for CrisisBench dataset

3. Training and Model Selection

3.1. CNN

The CNN model contains multiple adjustable hyperparameters, including the number of convolution filters, embedding dimension, kernel sizes, dropout probability, and vocabulary size. The model with the highest validation F1 score had a learning rate of 1×10^{-3}, 50 filters per kernel, batch size of 64, 50-dimensional GloVe embeddings, dropout of 0.5, a vocabulary size of 5,000, and a maximum input length of 64 tokens.

With this configuration, the learnable parameters can be quantified as follows. The 50-dimensional GloVe embed-



(a) CNN (b) Custom Transformer (c) DeBERTa

Figure 1: Validation curves for all three models (see Table 5 in the appendix for details of each configuration)

ding matrix accounts for $5000 \times 50 = 250000$ trainable weights. The three convolutional layers collectively add $\sum_{fs\in\{3,4,5\}} 50 \cdot (50 \cdot fs + 1) = 30150$ parameters, since each filter learns $50 \times fs$ weights plus one bias term. After max-pooling, the resulting feature vector of dimension $3 \times 50 = 150$ is passed to a linear classifier for three-way prediction, contributing $(150 \cdot 3) + 3 = 453$ additional weights. As dropout contains no trainable components, the final model contains 280,603 learnable parameters under the best configuration.

Changing the number of convolution filters had the most notable impact on model capacity. As the number of filters increases, the parameter count in the convolution layers scales proportionally according to $\sum_{fs\in\{3,4,5\}} nf \cdot (50 \cdot fs + 1)$. For configurations with 25, 50, 75, and 100 filters, this corresponds to approximately 15075, 30150, 45225, and 60300 learnable parameters, respectively. As shown in Figure 1a, varying the number of filters resulted in notable changes in validation performance. Too few filters constrain the model's ability to capture diverse n-gram patterns and semantic cues, whereas too many inflate capacity and promote overfitting relative to dataset size. In this setting, an intermediate configuration of 50 filters per kernel produced the best generalization.

3.2. Custom Transformer

The Transformer model includes several hyperparameters, such as the hidden embedding dimension, number of attention heads, number of encoder layers, feed-forward network width, maximum input sequence length, dropout probability, weight decay, and learning rate. The best configuration used a hidden size of 256, four attention heads, two encoder layers, a feed-forward dimension of 512, a learning rate of 3×10^{-4}, weight decay of 0.01, a dropout rate of 0.3, a vocabulary size of 30,000, and a maximum token length of 64.

With this configuration, the token embedding layer contains $30{,}000 \times 256 = 7{,}680{,}000$ parameters and the positional embedding layer contains $64 \times 256 = 16{,}384$ parameters. Each encoder layer includes $3 \times (256 \times 256 + 256) + (256 \times 256 + 256) = 263{,}168$ parameters for the query, key, value, and output projection matrices. The feed-forward subnetwork contributes $(256 \times 512 + 512) + (512 \times 256 + 256) = 262{,}912$ parameters, and the two layer-normalization modules add $2 \times (256 + 256) = 1{,}024$ parameters. With two encoder layers, the encoder contributes $1{,}054{,}208$ learnable parameters in total. In addition, the final layer-normalization module contains 512 parameters, and the classifier head consists of a $256 \times 256 + 256 = 65{,}792$ parameter linear layer followed by a $256 \times 3 + 3 = 771$ parameter output layer. Summing all components results in a total of $8{,}817{,}667$ learnable parameters.

Changing the learning rate had the strongest impact on validation performance. Figure 1b illustrates how validation performance varies as the learning rate changes while other hyperparameters remain fixed. Validation F1 increases steadily up to 3×10^{-4}, where the model achieves its peak score, and declines beyond this point as larger step sizes are introduced. This trend aligns with optimization theory, where excessively high learning rates tend to overshoot local minima, destabilize updates and ultimately reduce generalization performance.

3.3. DeBERTa

Unlike our lightweight encoder-only Transformer, DeBERTa-v3-base is a substantially larger pre-trained model. It is built on 128,000-token vocabulary with approximately 98,000,000 parameters in the embedding layer. It contains 12 Transformer encoder layers with a hidden size of 768 with the backbone alone accounting for about 86,000,000 parameters. This results in roughly 184,000,000 learnable parameters in total [4]. Since we do not modify the internal architecture, we simply reuse the predefined DeBERTa classification head and adapt only its output layer for our three-way label space. Thus, fine-tuning primarily updates the existing pretrained representations and a relatively small number of task-specific parameters on top.

Hyperparameter tuning was performed while keeping the pretrained architecture fixed, modifying only learning rate, weight decay, and batch size. The best validation performance was achieved at 5×10^{-6} with weight decay 0.01 and batch size 16. Changing the learning rate had the strongest impact on validation performance. Figure 1c shows the vali-

Model	Accuracy	Precision	Recall	F1	Class	TP	FP	FN	TN
Custom Transformer	0.8545	0.8204	0.7807	0.7980	time_critical	1,999	521	1,041	13,774
					support_and_relief	2,274	630	766	13,665
					non_informative	10,539	1,372	716	4,708
CNN	0.8694	0.8276	0.8237	0.8255	time_critical	2,335	766	705	13,529
					support_and_relief	2,380	543	660	13,752
					non_informative	10,356	955	899	5,125
DeBERTa	0.8877	0.8434	0.8625	0.8524	time_critical	2,484	724	556	13,571
					support_and_relief	2,600	601	440	13,694
					non_informative	10,303	623	952	5,457

Table 3: Confusion-matrix counts and overall classification performance metrics for all three models evaluated on the test set. True label counts for each class: time_critical = 3,040, support_and_relief = 3,040, non_informative = 11,255.

dation performances for learning rates of 1×10^{-6}, 5×10^{-6}, 1×10^{-5}, and 2×10^{-5} while keeping other configuration fixed. We see that the optimal learning rate value is 5×10^{-6}, which is a lot smaller compared to the learning rates chosen for other models (CNN: 1×10^{-3}, Custom Transformer: 3×10^{-4}). This indicates that a slight increase in the learning rate could make drastic changes in the model's internal representations and destabilize the learning process due to the extensive amount of parameters in the pretrained model.

3.4. Learning Curves

Figure 2 shows the learning curves for each model under their best-performing hyperparameter settings, with the red vertical line marking the epoch of peak validation performance. Validation F1-score reached a near-saturated level as early as the first epoch and showed minimal improvement afterward, even as training performance continued to rise. This suggests that later epochs contributed more to fitting the training data than to improving generalization. This may be due to limited dataset size (61,089 samples) and the short, low-variability nature of tweet-length inputs.

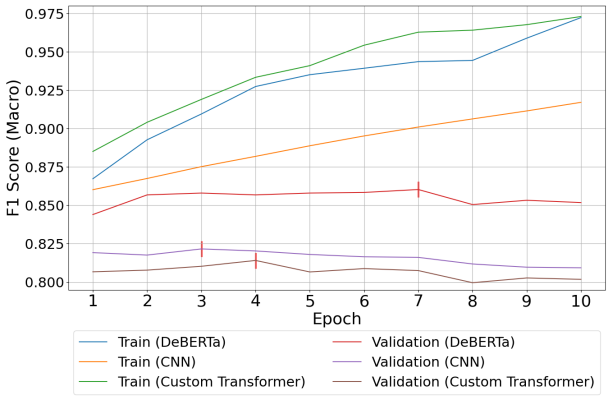


Figure 2: Learning curves

This behavior aligns with known properties of CNNs and Transformer encoders. Both models can acquire discriminative features from sentence-level inputs rapidly, with CNNs capturing local n-gram patterns efficiently and Transformers modeling global context in a single attention step. Because the task involves short, self-contained text rather than long-context discourse, extensive training was likely unnecessary, and most useful features appear to have been learned within the first few epochs.

4. Results

To evaluate whether our models can successfully classify tweets into the target labels, we analyzed their performance on the test set. First, we analyzed the performance of each of the three models quantitatively to choose the best one. Since the classification of time_critical and support_and_relief tweets is more important than identifying non_informative ones, we conducted our analysis with particular attention to these two classes. Subsequently, we manually examined misclassified tweets from the selected model to gain a deeper qualitative understanding.

4.1. Quantitative Analysis

Table 3 shows a clear performance gradient across accuracy, precision, recall, and F1. The custom Transformer performed the lowest, the CNN showed moderate improvement, and the pretrained Transformer achieved the best overall performance.

A similar pattern appears in the confusion matrix. For both time_critical and support_and_relief classes, true positive counts rise across the three models in the same order, indicating stronger sensitivity to minority classes. This is particularly notable given the strong class imbalance, with non_informative tweets forming the majority of the dataset. Even under this imbalance, the pretrained model did not default to majority prediction. In contrast, true positive and false positive counts for non_informative tweets gradually declined from the custom Transformer to pretrained Transformer, indicating that weaker models tended to default to the majority label when uncertain rather than making a careful distinction.

Taken together, these results provide strong quantitative

evidence that finetuned pretrained Transformer is the most effective model for this task. It demonstrates superior ability to identify critical crisis-related information while reducing unnecessary majority-class misclassification.

It is important to note that this performance gain cannot be explained by model size alone. Although the pretrained Transformer has more learnable parameters than the other two models, the CNN still outperformed the custom Transformer despite having fewer parameters. This suggests that architectural design and representational strength play a more decisive role than capacity alone. In this context, models trained from scratch likely struggled to learn stable decision boundaries under limited training data, whereas DeBERTa-v3 benefited from large-scale pretraining and transferred linguistic knowledge. This reduced the effect of data scarcity and enabled better generalization to minority humanitarian labels, leading to the highest macro F1 and the most reliable predictions.

4.2. Qualitative Analysis

We conducted qualitative analysis by manually evaluating the classified tweets using our best model, the fine-tuned pretrained Transformer. We especially focused on the misclassified cases between the most important class (time_critical) and the least important class (non_informative). This allows us to highlight the worst-case scenarios in which urgent information is dismissed or non-actionable content is treated as critical.

We began by examining the false negative cases, where our model misclassified time_critical tweets as non_informative. Manual inspection of all 358 instances revealed surprising result that many of these tweets were *mislabeled* within the CrisisBench dataset. Most of the misclassified tweets lacked actionable requests, location-specific information, or any indication of immediate need. Many were not even related to crisis events (e.g., "how many fertilizer plants are there in texas?"). These findings suggest that a substantial portion of what appeared to be model errors were instead labeling issues within the dataset. In other words, the model correctly rejected content that the dataset mislabeled as urgent, indicating that the trained decision boundary was in many cases more reliable than the provided ground truth.

Analysis of the false positive cases, where our model misclassified non_informative tweets as time_critical, also revealed informative patterns. Many of these tweets referred to negative events even though they were labeled as non_informative in the CrisisBench dataset. For example, some tweets were not related to an *ongoing* crisis but contained words associated with harm or danger such as "death" and "murder" (e.g., "Tattingstone suitcase murder: Police appeal over Bernard Oliver death - BBC News"). This pattern suggests that the model tends to overreact to signals of possible danger rather than underreacting to them. On the other hand, some tweets appeared to be *mislabeled* as non_informative despite containing urgent information that could be used to save people's lives (e.g., "Massive fire erupts at #Chevron refinery in #California").

Overall, the qualitative inspection shows that many apparent model errors were caused by issues in the CrisisBench dataset rather than by limitations of the classifier. The model often made judgments consistent with the semantic content of the tweets, rejecting mislabeled urgent posts and elevating non_informative ones when they contained signs of danger. Although the model sometimes overreacted by predicting non_informative tweets labeled as time_critical, this tendency can be beneficial in practical crisis monitoring. Leaving ambiguous tweets for human review is safer than disregarding a potentially urgent message. Taken together, these findings indicate that the fine-tuned pretrained Transformer can capture the intended decision boundary and can be more reliable than the dataset labels, underscoring both the strength of the model and the limitations of the annotations.

5. Challenges and Limitations

Before running the experiments, we anticipated several issues. First, the informal and irregular style of Twitter text seemed likely to complicate training, so we applied various preprocessing methods that were used in prior works. We additionally removed Twitter-specific tokens such as retweet markers and usernames and found that removing elements such as retweets and usernames generally improved performance. Such extensive preprocessing resulted in 110 tweets becoming completely empty after cleaning. Excluding these empty entries led to improvement in model performance.

However, we have failed to address tweets that used hashtags instead of words ("Mexicans shed tears for hundreds killed in #earthquake ..."). Removing hashtags in these tweets sometimes led to loss in significant information ("mexicans shed tears for hundreds killed in ..."), which presumably led to loss in performance. As hashtag removal is a common practice in Twitter analysis, future work could delve into how we can identify hashtags that include meaningful information in the context and analyze the impact of properly using these hashtags.

We also expected the original 11 CrisisBench labels to be too fine-grained for reliable learning, which motivated our three-category mapping for both practical and methodological clarity. However, qualitative analysis revealed that the original CrisisBench dataset may have mislabeled cases, making the ground truth itself unreliable. Future work could delve into analyzing this issue and creating a better benchmark dataset for future research.

Student Name	Contributed Aspects	Details
Jinwoo Jeong	Data Preprocessing, Model Implementation and Training, Analysis	Preprocessed all the Twitter data used in the project. Implemented custom transformer and fine-tuned pretrained transformer model. Focused on writing the Introduction, Approach, Training, Results, Challenges and Limitations sections
Jade Kim	Model Implementation and Training, Analysis	Implemented CNN model. Fine-tuned CNN and custom transformer. Focused on qualitative analysis and decided to focus on misclassification between time_critical and non_informative cases. Created architecture figures.

Table 4: Contributions of team members.

6. Work Division

Summary of contributions is provided in Table 4.

References

[1] Firoj Alam, Ferda Ofli, and Muhammad Imran. Crisismmd: Multimodal twitter datasets from natural disasters. In *Proceedings of the international AAAI conference on web and social media*, volume 12, 2018. 1, 2

[2] Firoj Alam, Hassan Sajjad, Muhammad Imran, and Ferda Ofli. Crisisbench: Benchmarking crisis-related social media datasets for humanitarian information processing. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 15 of *ICWSM '21*, pages 923–932, May 2021. 1, 2, 3

[3] Carlos Castillo. *Big crisis data: social media in disasters and time-critical situations*. Cambridge University Press, 2016. 1

[4] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021. 3, 4

[5] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020. 3

[6] Louis Hickman, Stuti Thapa, Louis Tay, Mengyang Cao, and Padmini Srinivasan. Text preprocessing for text mining in organizational research: Review and recommendations. *Organizational Research Methods*, 25(1):114–146, 2022. 2

[7] Muhammad Imran, Prasenjit Mitra, and Carlos Castillo. Twitter as a lifeline: Human-annotated twitter corpora for nlp of crisis-related messages. *arXiv preprint arXiv:1605.05894*, 2016. 1, 2

[8] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014. 2

[9] Dat Nguyen, Kamela Ali Al Mannai, Shafiq Joty, Hassan Sajjad, Muhammad Imran, and Prasenjit Mitra. Robust classification of crisis-related data on social networks using convolutional neural networks. In *Proceedings of the international AAAI conference on web and social media*, volume 11, pages 632–635, 2017. 2, 3

[10] Xiaodong Ning, Lina Yao, Boualem Benatallah, Yihong Zhang, Quan Z Sheng, and Salil S Kanhere. Source-aware crisis-relevant tweet identification and key information summarization. *ACM Transactions on Internet Technology (TOIT)*, 19(3):1–20, 2019. 1, 2, 3

[11] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 376–385, 2014. 1, 2

[12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014. 2

[13] Dwaipayan Roy, Mandar Mitra, and Debasis Ganguly. To clean or not to clean: Document preprocessing and reproducibility. *Journal of Data and Information Quality (JDIQ)*, 10(4):1–25, 2018. 2

[14] Zekiye Tamer, Gülay Demir, Sefer Darıcı, and Dragan Pamučar. Understanding twitter in crisis: a roadmap for public sector decision makers with multi-criteria decision making. *Environment, Development and Sustainability*, pages 1–37, 2025. 1

[15] Matti Wiegmann, Jens Kersten, Friederike Klan, Martin Potthast, and Benno Stein. Analysis of detection models for disaster-related tweets. In *17th Annual International Conference on Information Systems for Crisis Response and Management (ISCRAM 2020)*, pages 872–880, Blacksburg, Virginia, USA, 2020. 1, 2, 3

A. Project Code Repository

The Github repository for our final project is at: https://github.com/bugoverdose/dl-twitter-crisis.

All datasets used for our experiments can be found in the original CrisisNLP dataset Github page, available to download by following these instructions: https://github.com/firojalam/crisis_datasets_benchmarks?tab=readme-ov-file#download. The preprocessed versions are available in our github repository.

B. Validation Curve Supplementary

number of filters	epoch	train F1	validation F1
25	6	0.8490	0.8258
50	4	0.8873	0.8266
75	5	0.8593	0.8253
100	5	0.8593	0.8247

(a) CNN

learning rate	epoch	train F1	validation F1
1e-4	3	0.7992	0.7676
3e-4	3	0.8363	0.7920
1e-3	4	0.8171	0.7793

(b) Custom Transformer

learning rate	epoch	train F1	validation F1
1e-6	16	0.9570	0.8583
5e-6	7	0.9436	0.8603
1e-5	5	0.9675	0.8584
2e-5	3	0.9535	0.8579

(c) DeBERTa

Table 5: Summary of the epoch and performance associated with each point plotted in the validation curves for CNN, Custom Transformer, and DeBERTa models.

C. Model Architectures

Figures 3 and 4 present the flow charts of our CNN and custom Transformer implementations.

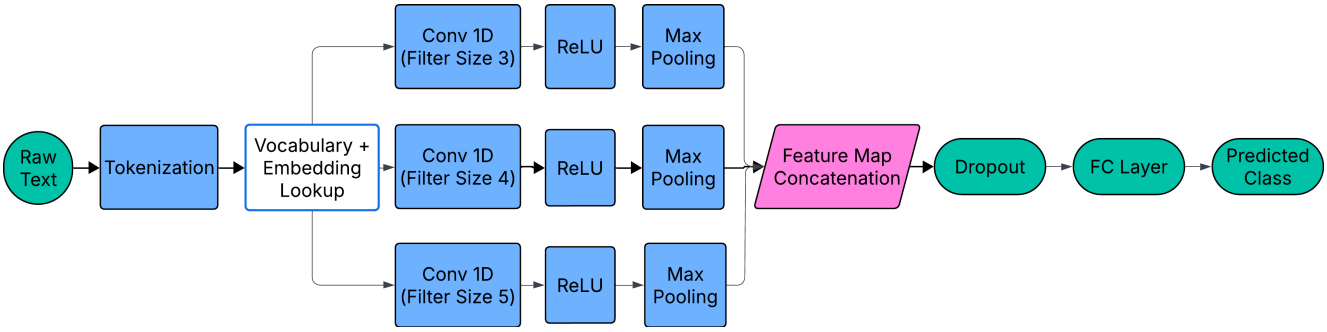


Figure 3: CNN Architecture

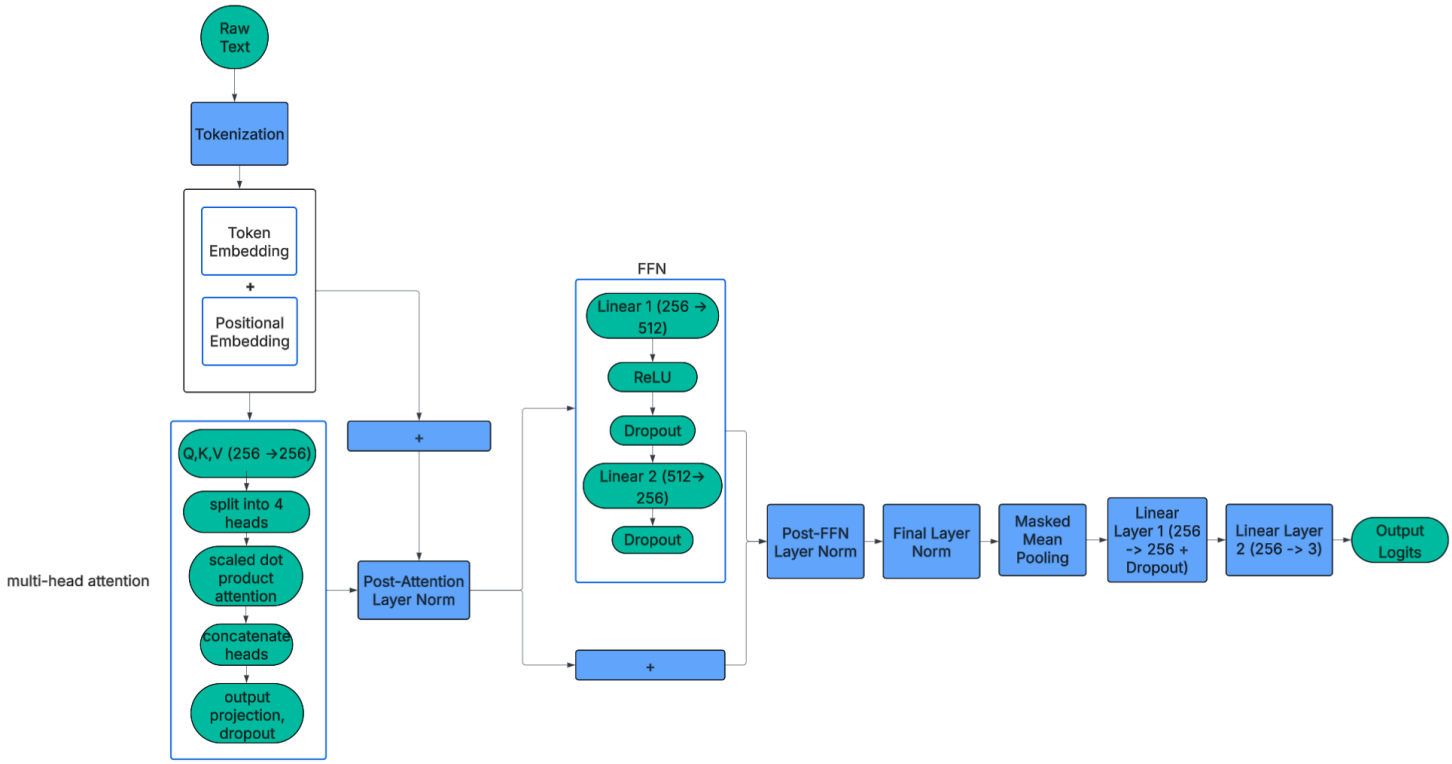


Figure 4: Custom Transformer Architecture