Comparing Few-Shot Learning with LLMs for Efficient Text Classification in Road Maintenance Applications

Varun Kumar Reja[1,*], Ching Yau (Fergus) Mok[1,*], Aritra Pal[1,2] and Ioannis Brilakis[1]

[1] University of Cambridge, UK
[2] Indian Institute of Technology Madras, India

varunreja7@gmail.com

Abstract -

Efficient road maintenance is vital for long-lasting and safe transportation networks, but traditional methods that rely on manual inspection are labour-intensive and error-prone. The integration of Natural Language Processing (NLP) and Large Language Models (LLMs) presents a transformative solution for automating text-based tasks in road maintenance. This study investigates the application of LLM-based text classification models to process unstructured textual data from road maintenance logs, with a focus on resource-constrained scenarios characterised by limited labelled datasets. Two primary approaches were evaluated: traditional fine-tuning and few-shot learning. Using a public dataset from New York City roadwork inspections containing 83 distinct classes, we performed extensive model comparisons. Pre-trained transformer models *Llama* and *BERT* were fine-tuned to achieve baseline performance. Additionally, a few-shot learning method, SetFit, was employed to address data scarcity through efficient task adaptation using minimal labelled examples. Results showed SetFit outperformed fine-tuning in low-resource scenarios, achieving high accuracy and F1 scores with as few as 1-10 examples per class, reducing annotation efforts. This highlights the potential of few-shot learning for real-world deployment. Future work will address scalability, multimodal data fusion, and integration into predictive maintenance.

Keywords -

Few-Shot Learning; Large Language Models; Road Maintenance; Text Classification; SetFit; Natural Language Processing; Transformer Models; Inspection Logs; Infrastructure Management; Low-Resource NLP

1 Introduction

Effective road maintenance is essential for the safety, reliability and longevity of transportation networks. However, increasing traffic volumes and ageing infrastructures present challenges, such as identifying deteriorating assets, ensuring timely intervention, and optimising resource allocation. Inspection logs provide valuable information on road maintenance. However, manually analysing an enormous volume of inspection reports is labour-intensive, time-consuming, and error-prone [1, 2]. These issues are further amplified in a resource-constrained setting where funding and manpower are limited [3].

The advent of artificial intelligence (AI) and natural language processing (NLP) offers the potential for automating road maintenance tasks like analysing inspection data, prioritising repairs, and generating insights [4]. Transformer-based models like BERT and GPT have become state-of-the-art for text processing [5], yet their application in road maintenance remains underexplored, especially in low-resource settings with limited labelled data.

This paper aims to bridge this gap by systematically evaluating the use of transformer-based text classifiers in road maintenance applications, particularly in resource-constrained settings. Specifically, the study focuses on:

1. **Comparative Analysis**: Evaluating the performance of traditional fine-tuning approaches across various pre-trained transformer model architectures and sizes, highlighting their effectiveness in processing road maintenance-related textual data.
2. **Few-Shot Learning**: Assessing the capability of dedicated few-shot models to deliver robust performance with minimal labelled data, catering to the constraints of limited resources often encountered.

The paper is structured as follows: Section 1 introduces the topic, followed by background in Section 2. Section 3 outlines the methodology, while Section 4 discusses the findings, limitations, and future research directions. Finally, Section 5 presents the conclusions.

2 Background

2.1 Challenges in Leveraging Text Data for Road Maintenance

Malihi et al. [6] emphasise the untapped potential of textual records such as inspection and maintenance reports in road maintenance, noting that their unstructured and inconsistent nature limits their utilisation. Unlike numerical or sensor data, text is inherently diverse in format, structure, and terminology. Inspection reports, for example, may describe similar issues in varied ways depending on

*Co-First Authors (Equally Contributed to this work).

the inspector's language and style. This lack of standardisation makes it difficult to extract consistent and actionable insights. Another challenge is the volume of data generated across extensive road networks. Maintenance operations often produce large quantities of text data over time, including daily logs, contractor reports, and communications. Manually sifting through this information to identify critical issues is time-consuming and prone to human error. The ambiguity and context-dependence of language further complicate the processing of text data. Terms like "minor crack" or "urgent repair" may vary in meaning depending on the context, location, or maintenance standard, leading to inconsistent interpretation. Similarly, implicit details, such as the severity of a reported defect, may not be readily apparent without supplementary information [5].

Additionally, there is a lack of annotated datasets specific to road maintenance applications. Developing effective machine learning models often requires large, labelled datasets to train algorithms. In road maintenance, creating such datasets is challenging due to the domain-specific nature of the text and the expertise required to accurately label data. This echoes [7], which raised concerns about the lack of annotated and high-quality data in the Architecture, Engineering, and Construction (AEC) domain.

These challenges underscore the need for innovative methods to efficiently classify, interpret, and utilise textual data in road maintenance.

2.2 NLP as an Opportunity in Road Maintenance

NLP, a field that enabling machines to comprehend natural language, is transformative for extracting insights from unstructured maintenance data like logs and inspection reports. Machine learning models can identify patterns in defect descriptions and prioritise tasks based on urgency. Advanced transformer-based NLP architectures like LLMs excel in handling language's contextual nuances. Khalil et al. [4] highlight the significant advancements of LLMs, such as GPT-3, in reasoning tasks, further enhanced by techniques like "Chain of Thought" prompting. In construction, [8] designed an automated system for construction specification review using NLP techniques such as named entity recognition (NER) with Bi-LSTM and document embedding through Doc2Vec to streamline processes and improve construction risk management. Recognising the difficulty in many AEC applications in collecting sufficient training data, [9] proposed a few-shot NER model for information extraction from bridge inspection texts.

2.3 LLM-Based Text Classification Approaches

LLMs have extensive capabilities in classifying texts due to their ability to accurately encode sentences [10]. They use the transformer architecture [11], a neural network leveraging the attention mechanism [12] to model dependency, to process long inputs at once without loss of information as is the case with recursion in approaches such as Recurrent Neural Network. LLMs utilise pre-trained models that possess task-agnostic knowledge through training on vast amounts of general text, akin to reading a million books. To adapt LLMs for downstream tasks such as text classification, architecture-specific fine-tuning using a task-specific dataset is performed. The following sections elaborate on different architectures and their associated fine-tuning techniques.

2.3.1 Traditionally Fine-tuned Models

The most common LLM-based text classifier, shown in Figure 1, appends a small task-specific classification head to produce class probabilities or logits. It typically consists of an uninitialised fully connected layer with neurons equal to the number of classes. Pre-trained models vary in size: smaller models (100M parameters), often termed Pre-trained Language Models (e.g., *BERT*), and larger models (7B+ parameters), such as *Llama*. For instance, BERT was pre-trained on 800M words from BooksCorpus and 2,500M words from English Wikipedia [13]. In this work, both 100M and 7B models are referred to as LLMs. Fine-tuning adapts the classifier to a specific task by updating pre-trained model parameters and initialising the classification head in a single step, using a cross-entropy loss function on model output logits.

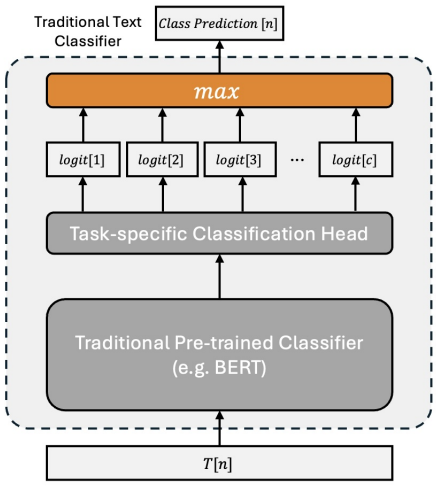

Figure 1. Architecture of a traditional transformer-based text classifier.

2.3.2 Few-shot Learning - SetFit

Few-shot learning refers to adapting models to a task using only a small number of examples (or shots). This is particularly useful for resource-constrained applications where large-scale annotated data is unavailable. A popular transformer-based few-shot learning technique is Sentence

Transformer Fine-Tuning (SetFit) [14].The classifier embeds text using Sentence-BERT (SBERT), a BERT variant pre-trained on pairwise similarity tasks, and adds a lightweight classification head to generate class scores. This setup produces semantically rich embeddings suited for classification.

SetFit is fine-tuned in a two-step process. First, the SBERT encoder is fine-tuned to produce accurate embeddings through contrastive learning by providing a pair of texts and training the model to determine whether they belong to the same class. This is followed by training the entire model (SBERT plus the classification head) to perform the classification task through cross-entropy. The contrastive learning adopted in the embedding training step is what allows SetFit to be data efficient in that it can train the model to explicitly separate the embeddings of texts from different classes.

The literature highlights the potential of NLP and LLMs in extracting insights from road maintenance texts, yet challenges remain due to unstructured data, inconsistency, and limited annotated datasets. While LLMs and few-shot learning methods like SetFit show promise, their application to domain-specific classification and automated maintenance prioritization is underexplored. This study addresses these gaps by evaluating LLM-based classification approaches for structured information extraction.

3 Research Methodology

3.1 Dataset

A public dataset on roadwork inspection comments was used to validate the proposed approach. It is the "Street Construction Inspections and Corrective Action Requests" from the New York City (NYC) Open Data [15], which is a collection of inspection comments on roadwork performed by non-governmental third parties that record instances of road code violations. Examples of inspection comments and their associated violated road code are listed in Table 1. From the dataset, the free text inspection comments come from the "DetailsofViolation" column and the associated violated road codes from "NOVCodeDescription".

The dataset underwent a series of cleaning processes and random subsampling to create few-shot datasets that can simulate data constraints. Figure 2 shows the steps taken to generate the datasets used in this work.

The raw dataset was first cleaned, starting by inspecting for irregularities. It was found that there were road codes that contained duplicate meanings that essentially refer to the same sets of violations. Table 2 outlines such examples that were manually merged. Following this, codes with fewer than 100 examples were excluded to ensure sufficient testing examples for representative conclusions to be drawn. This resulting dataset has a total of 400,000

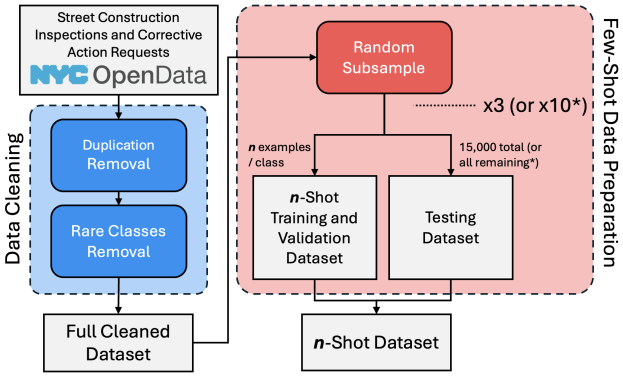

Figure 2. Illustration of the data preparation process for an *n*-shot dataset. (*further experiments with smaller models were performed using more datasets per shot and more testing examples per dataset.)

texts that belong to 83 classes.

The full, cleaned dataset was then randomly subsampled to form few-shot datasets. *Training datasets* were generated by subsampling 1, 2, 3, 4, 5, and 10 examples per class (shots), each shot was subsampled from all available data three times to create three statistically independent draws which helps mitigate the instability that can occur with small training datasets [14]. Each *Training datasets* is coupled with a corresponding *Validation dataset* and *Testing dataset*. *Validation datasets* were subsampled for roughly 1/3 of the shot to maintain the ratio of training to validation data common in standard machine learning datasets (minimum of 1 example per class), while the remaining unsampled data were used as a potential testing pool. For *Testing datasets*, 15,000 examples were subsampled (similar order of magnitude to relevant works [8]) to reduce computational costs, particularly for models with 7 billion parameters.

Note that to assess the impact of these subsampling operations, a further set of experiments were conducted with only smaller 100 million parameters models: First, few-shot scenarios used the full remaining unsampled testing pool (as opposed to 15,000) and ten individual datasets per shot (as opposed to three). Second, an all-data training scenario was introduced using the full dataset with an 8/1/1 split for the training, validation, and testing sets.

3.2 Simulated Task

The task simulated in this case study involves training a transformer-based text classifier using limited data to classify roadwork inspection comments into the violated road code. Each inspection comment records the details of an instance of a code violation, which in this dataset, can belong to one of 83 road codes. The simulated task highlights two key challenges: limited training examples and a large number of classes, reflecting a complex technical text classification scenario.

Table 1. Examples of pairs of free text comments describing instances of code violation (textual inputs) and static code descriptions (class descriptions)

Free Text Inspection Comments	Associated Road Code Violated
A/T/P/O I observed that the respondent has failed to remove the ramping material from the roadway after the completion of the final restoration or prior to the expiration of the permit as per subsection (vii). Respondent was notified to fix this condition via CAR [#20184520373] issued on 12/31/18.	Failure to remove plating and decking after final restoration and prior to DOT permit expiration.
Respondent failed to display required signs at work site with permit number,work type, start date ,completion date.No one on site at time of inspection. If not admitting the charge, you MUST APPEAR IN PERSON.	Identifying signs improperly displayed or missing
A/T/P/O I observed respondent identified by "BIC 3179" markings on Commercial Refuse Container, having a container in the roadway without any visible reflective markings capable of producing illumination or warning glow when struck by vehicle headlamps. No one onsite.	Commercial refuse container without proper reflective markings on all four sides

Table 2. Examples of road codes with duplicating definitions.

Code 1	Code 2	Reason for duplication
Commercial refuse container/debris obstructing sidewalks, gutters, crosswalks or driveway	Commercial refuse container/debris obstructing sidewalks, gutters, crosswalks, bike lanes or driveway	Semantically identical: Code 1 is strictly contained within code 2
Failure to post âSteel Plates Aheadâ or âRaise Plowâ sign; failure to countersink plates flush with roadway	Failure to post “Steel Plates Ahead” or “Raise Plow” sign; failure to countersink plates flush with roadway	Character encoding error: The “â” is a result of a character encoding error originating from the publisher

Limited data refers to training models with a small number of violation examples per road code (number of shots n). Various ns were experimented with to simulate different levels of data constraints. 10 was selected as the maximum, considering that it would amount to 830 total examples with 83 classes, which can already be difficult to replicate in many resource-limited real-world engineering applications. This will provide insight into how well different models perform under different data availabilities.

A unique characteristic of the employed dataset is the large number of classes. Using natural language to infer one of 83 road codes that are being violated can be challenging due to the complexity of the classification task. However, this is a common characteristic in many road maintenance and AEC applications that are tightly regulated and stringent. For example, the dataset used records instances of road code violations which correspond to regulations on roadwork, having an impact on both safety for road users and disruptions caused.

3.3 Models

The two types of transformer-based models outlined in Section 2.3 were experimented with. Traditionally, fine-tuned models represent standard transformer-based text classifiers, which typically require a larger amount of training data to be performant, while SetFit is an example of a technique that specialises in tasks where there is a limited number of training data, like road maintenance.

Traditionally Fine-tuned For traditionally fine-tuned models that were explored in Section 2.3.1, different pre-trained models were experimented with, which have different numbers of parameters. They mostly belong to two groups: those with roughly 100 million parameters and 7 billion parameters. Table 3 lists the pre-trained models used as traditionally fine-tuned text classifiers along with hyperparameters that affect model weights updates and convergence. The 100 million parameters model was the three best-performing models of this size in a comparison study [16], and they can be run on a wider range of hardware (tested with an M1 MacBook Pro with 32GB of unified memory). The 7 billion models are also widely quoted in literature [17][18] and exhibit high performance at the expense of requiring much more computational power, including GPU and VRAM.

Table 3. Pre-trained models used and their associated training hyperparameters (search range in [ ])

Model	Hyperparameters		
	No. params	Learning rate	Batch size
bert-base-uncased	110M	4e-5$_{[1e-6,1e-4]}$	16$_{[4,32]}$
roberta-base	125M	2e-5$_{[1e-6,1e-4]}$	16$_{[4,32]}$
xlnet-base-cased	110M	2e-5$_{[1e-6,1e-4]}$	16$_{[4,32]}$
Llama-2-7b-hf	6.74B	2e-4$_{[1e-3,1e-5]}$	3
Meta-Llama-3-8B	8.03B	2e-4$_{[1e-3,1e-5]}$	3
Mistral-7B-v0.1	7.24B	2e-4$_{[1e-3,1e-5]}$	3

Few-shot Additionally, SetFit, a transformer-based few-shot technique discussed in Section 2.3.2, was also experimented with to evaluate performance improvements in a low-resource workflow brought by dedicated few-shot techniques. The experimental design was modeled after the methodology described in the study by [14], with the 300 million parameter *paraphrase-multilingual-mpnet-base-v2* used as the pre-trained SBERT model and hyperparameters affecting model weights updates and convergence for the two training phases [14] listed in Table 4. SetFit has not been focused on in works relating to classifying AEC texts such as road inspection comments, and

its performance can be validated against more traditional techniques in various data-constrained settings.

Table 4. SetFit training hyperparameters for the two fine-tuning stages [14]. The first value corresponds to the SBERT encoder, and the second to the classification head. During the *Embedding* phase, no classification head is involved (N/A).

Hyperparameters	Fine-tuning	
	Embedding	Classifier
Learning rate	2e-5, N/A	1e-5, 1e-2
Batch size	16, N/A	16, 2
No. epochs	1, N/A	1, 16

3.4 Implementation Details

The models in this study were trained using the Transformers and SetFit libraries on an RTX 4080 (16GB VRAM). Hyperparameters were selected on validation set performance. Training lasted up to 20 epochs with the best model being chosen automatically on validation results.

3.5 Metrics

Accuracy and $f1$ score are the reported metrics in this study and are defined by the following equations:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad (1)$$

$$f1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (2)$$

Accuracy values reported are micro-averaged, meaning that it assigns equal importance to each testing example. It is the overall number of correct classifications over all classifications. $f1$ values reported are macro-averaged, where equal importance is assigned to each class, meaning that performances in underrepresented classes in testing examples are taken into account in the overall metric values, which is useful in class imbalanced applications.

4 Results and Discussion

This section outlines the key results. Table 5 lists averaged *accuracy* and $f1$ values of all evaluated models for each few-shot scenario tested, including 100M and 7B traditionally fine-tuned models and SetFit.

4.1 SetFit Outperforms all Traditionally Fine-tuned Models

Classification performance is critical in road maintenance as errors can compromise safety and reliability. For instance, misclassifying "Identifying signs improperly displayed or missing" can lead to possible accidents due to the lack of warning for works. As a classifier specifically designed for resource-constrained settings, SetFit outperformed all traditionally fine-tuned and structured transformer-based classifiers in every few-shot scenario. In a 10-shot setting, SetFit achieved an $f1$ score of 0.539 and an accuracy of almost 70%, while the best traditionally fine-tuned model attained an $f1$ of 0.524 and an accuracy of 66%, respectively. This trend becomes more apparent as the number of shots is reduced. At a 1-shot setting, SetFit outperformed the best performing traditionally-fine-tuned model by more than 3 times in terms of $f1$ score, achieving 0.232 vs 0.074 with just 1 violation example per road code. Figure 3 plots the results of SetFit against three 7B models. Despite utilising 20 times more parameters, the 7B traditionally fine-tuned models were not able to outperform SetFit. This signifies that computationally efficient models can outperform state-of-the-art text classifiers in a resource-constraint domain-specific setting if they are effectively used, such as the case of SetFit. The contrastive learning stage of SetFit allows the text encoder to effectively distinguish between classes, making efficient use of model parameters. This demonstrates that the use of SetFit in a resource constrained setting in road maintenance is more suitable than traditionally fine-tuned techniques.

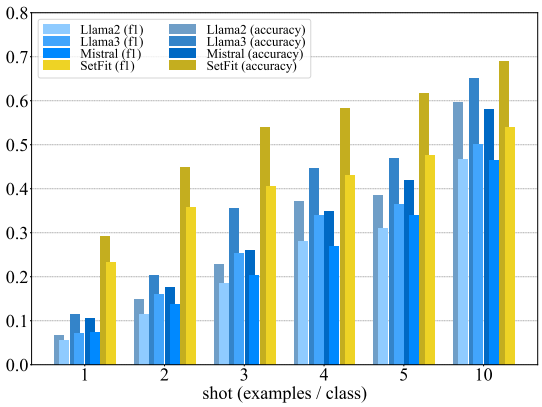

Figure 3. *Accuracy* and $f1$ scores for 7 billion parameter traditional models and SetFit.

4.2 Model Sizes Have Little Impact on Few-Shot Performance

Looking at the results for traditionally fine-tuned models with 100M and 7B parameters, the model's performances do not correlate with size. For instance, by looking at metric values in Table 5, the 100M *bert-base-uncased* model outperformed all 7B models in a 10-shot setting.

This observation indicates that increasing the number of parameters in traditionally fine-tuned models will likely not bridge the gap in performance with SetFit. This can be due to the use of SBERT in SetFit, where the pre-training objective focusing on semantic similarity can potentially better capture the semantics of the inspection comments than the objectives of other models, such as text generation.

Table 5. $F1$ and $accuracy$ results for all tested models. Testing results per shot were calculated with three independent subsampled datasets on a total of 15,000 testing examples.

Shots	Traditionally Fine-Tuned						Few-Shot
	100M			7B			
	BERT	RoBERTa	XLNet	Llama2	Llama3	Mistral	SetFit
	f1 scores						
1	0.031	0.053	0.049	0.056	0.072	0.074	0.232
2	0.107	0.143	0.097	0.114	0.160	0.137	0.357
3	0.195	0.183	0.143	0.186	0.254	0.203	0.405
4	0.288	0.265	0.146	0.281	0.339	0.269	0.430
5	0.372	0.303	0.231	0.311	0.365	0.340	0.475
10	0.524	0.452	0.432	0.467	0.501	0.464	0.539
	accuracy scores						
1	0.041	0.067	0.078	0.066	0.115	0.105	0.291
2	0.147	0.174	0.120	0.148	0.204	0.177	0.449
3	0.262	0.227	0.161	0.228	0.355	0.260	0.539
4	0.347	0.344	0.186	0.372	0.446	0.349	0.582
5	0.502	0.397	0.294	0.386	0.469	0.419	0.617
10	0.660	0.589	0.556	0.596	0.650	0.581	0.690

Table 6. Additional $accuracy$ and $f1$ results for all tested 100M models and SetFit. Testing results per shot were calculated with ten independent subsampled datasets on all available testing data.

Shots	Traditionally Fine-tuned						Few-Shot	
	BERT		RoBERTa		XLNet		SetFit	
	acc.	f1	acc.	f1	acc.	f1	acc.	f1
1	0.058	0.039	0.092	0.074	0.066	0.043	0.304	0.254
2	0.150	0.123	0.195	0.162	0.124	0.108	0.443	0.373
3	0.249	0.199	0.260	0.211	0.163	0.140	0.525	0.434
4	0.345	0.285	0.331	0.270	0.217	0.177	0.576	0.470
5	0.445	0.359	0.410	0.326	0.292	0.240	0.619	0.508
10	0.673	0.552	0.600	0.489	0.547	0.442	0.691	0.579
	Training over all available data							
all	0.954	0.899	0.951	0.890	0.952	0.893	N/A	N/A

4.3 Subsampling Testing Data Did Not Impact Observations

Due to the need to generate few-shot samples and to account for computational constraints, the above experiments were performed with subsampling steps. To verify the observations above and to assess the impact these subsampling operations have, further experimentation was performed by using both a higher number of independent datasets per shot (10 up from 3) and by eliminating subsampling for testing data to utilise all available testing examples (up from 15,000). Results for these experiments are listed in Table 6, which includes all 100M traditionally fine-tuned models and SetFit.

Figure 4 visualises the $accuracy$ and $f1$ values for this additional set of experiments. The same observation as the prior sections remains valid in that SetFit outperforms all other traditionally fine-tuned models. Despite having a general shift in metric values from having a different set of samples, the relative performance has remained consistent.

Figure 4. $Accuracy$ and $f1$ scores in different few-shot settings for 100 million parameter traditional models and SetFit.

4.4 Absolute Performance Low in Extremely Constrained Settings

It is important to note that the absolute performance of all models tested remains moderate in few-shot settings. This is especially apparent given model performances when all data are available (~300,000), where all 100M traditionally fine-tuned models achieved $accuracy$ scores of above 95% and $f1$ of around 0.9. When given only 10 examples per class, the best performing model ($bert-base-uncased$) only retained around 60% of its all-data $f1$ score.

In AEC applications like road maintenance, the tolerance for error is low as tasks often require strict adherence to codes and regulations. For instance, the road inspec-

Table 7. Road codes attaining the highest and lowest classwise $f1$ scores for a 10-shot SetFit.

Road Code C	Classwise $f1$
Highest	
Failure to use plating and/or decking that is skid-resistant in its entirety on roadway	0.962
Failure to begin emergency work within 2 hours after authorisation	0.946
Construction material/equipment without proper reflective markings	0.942
Lowest	
Failure to restore in kind any parking or regulatory signs or supports	0.085
Failure to maintain a 5-foot pedestrian walkway on sidewalk	0.076
Failure to obtain permit for temporary roadway pavement markings changes	0.058

tion text classification task in this work involves road codes from roadwork licenses which have important implications for safety and disruption mitigation. By only attaining moderate values in few-shot settings, merely using traditionally fine-tuned models, even state-of-the-art LLMs, is not sufficient. Specialised few-shot techniques appear to be a better choice in resource-constrained settings even in a technical many-class application like road maintenance.

4.5 Classwise Performance Imbalance

There is a wide variation in the performance on each of the 83 classes. Figure 5 shows the histogram of the $f1$ scores for individual classes for a SetFit model trained with 10 examples per road code. There is a relatively high standard deviation across classes of 0.262, considering values are bounded between 0 and 1. Most classes exhibit moderately high $f1$ scores of around 0.75 or more, though a few classes with very low scores of below 0.25.

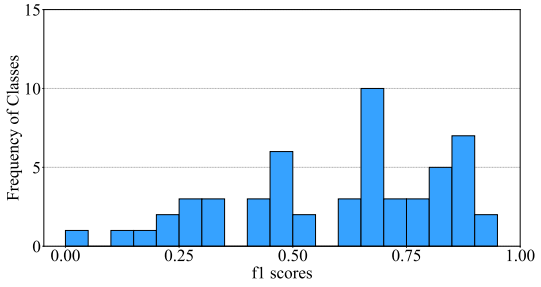

Figure 5. Histogram of $f1$ scores of individual classes for a 10-shot SetFit model.

Table 7 lists the three road codes with the highest and lowest $f1$ scores. The top codes all have high scores of around 0.95, whereas the bottom three have less than 0.1. One potential reason for the latter is that some codes have very specific definitions, such as requiring a "5-foot pedestrian walkway". In addition, there can be potential code duplications, such as the codes *Failure to obtain permit for temporary roadway pavement markings changes* and *Failure to obtain DOT permit for any changes to, or installation of, temporary roadway pavement markings and temporary construction, parking or regulatory signs and supports*, both of which cover very similar violation instances. These can lead to inconsistently annotated data that can both confuse the model during training and lead to a drop in prediction metric values.

To inspect the model's outputs, Table 8 provides examples of the predicted road code P for the inspection comment T with ground truth label L. The first example shows a correct classification where the model inferred that the code *Failure to properly place and ramp plating and decking* relates to examples with ramp elements. However, the second example shows an example of when it fails to infer this relationship for another code, possibly due to the presence of unrelated information from the inspection comments, such as instructions for charge admission.

5 Conclusions

Results indicate that dedicated few-shot techniques can outperform traditionally fine-tuned LLMs when classifying technical road maintenance texts in a resource-constrained setting, in addition to requiring less training data and computational resources than big LLMs with 7B parameters. For example, in 1-shot, the few-shot model SetFit outperformed the traditionally fine-tuned *Llama3* by three times in $f1$. In particular, SBERT used in SetFit appears to be a better pre-trained model choice than many larger models not pre-trained on similarity-based tasks.

However, an unresolved issue of few-shot techniques is that absolute performance is still moderate. With one example per class, the $accuracy$ is only 29%, though that increases to 69% with ten. In road maintenance and many tightly regulated AEC applications where reliability is crucial, this can impact their practicality. For instance, in this study on classifying inspection comments into roadwork licensing codes, there are important safety and disruption implications and necessitate high reliability. This prompts the need to explore alternative, specialised techniques that can tackle the domain-specific nature of the task in a more accurate and data-efficient manner.

6 Acknowledgement

Funded by the European Union

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101034337.

Table 8. Classification examples for inspection comments using a 10-shot SetFit.

Inspection Comment Example T	Prediction of Violated Road Code P	Correct Violated Road Code L	Outcome
I observed respondent failed to spike/ fasten 2 steel plates marked CWP at n/e/c of intersection to prevent movement.	Failure to properly place and ramp plating and decking	Failure to properly place and ramp plating and decking	Correct
A/T/P/O I observed the above respondent failed to repair a 1" gap within their transformer vault on the sidewalk. respondent was previously notified to correct this condition by via (CAR #20164060240) on 8/11/16. If not admitting the charge, you MUST APPEAR IN PERSON.	Failure to repair defective street condition found within an area extending 12 inches outward from the perimeter of the cover/grating	Failure to replace loose, slippery or broken utility maintenance hole (manhole) covers, castings, and other street hardware	Incorrect

References

[1] J. Xu, N. R. Anvo, H. A. Taha, A. M. d'Avigneau, D. Palin, R. Wei, G. Hadjidemetriou, S. Schaefer, L. de Silva, A. Al-Tabbaa, F. Lida, and I. Brilakis. Highway digital twin-enabled autonomous maintenance plant: a perspective. *Data-Centric Engineering*, 5:e24, 2024. doi:10.1017/dce.2024.34.

[2] A. Marie d'Avigneau, L. Potseluyko, N.R. Anvo, H.M.T. Abdalgadir, V.K. Reja, D. Davletshina, P. Lam, L. de Silva, A. Al-Tabbaa, and I. Brilakis. Camhighways: The cambridge highways dataset. *Advanced Engineering Informatics*, 2025. doi:10.1016/j.aei.2024.103036.

[3] D. Davletshina, V. K. Reja, and I. Brilakis. Automating construction of road digital twin geometry using context and location aware segmentation. *Automation in Construction*, 168:105795, 2024. doi:10.1016/j.autcon.2024.105795.

[4] R. A. Khalil, Z. Safelnasr, N. Yemane, M. Kedir, A. Shafiqurrahman, and N. Saeed. Advanced learning technologies for intelligent transportation systems: Prospects and challenges. *IEEE Open Journal of Vehicular Technology*, 5:397–427, 2024. doi:10.1109/OJVT.2024.3369691.

[5] A. Nevatia, S. Saha, S.B. Bhagavatula, and N. Bugalia. A pre-trained language model-based framework for deduplication of construction safety newspaper articles. In *40th ISARC*, 2023.

[6] S. Malihi, L. Potseluyko, A. Mathew, H. Alavi, V. K. Reja, Y. Pan, L. Binni, G. Wang, X. Wang, and I. Brilakis. Review of multimodal data and their applications for road maintenance. *Smart Construction*, 1(2):1–40, 2024.

[7] Michael P. Brundage, Thurston Sexton, Melinda Hodkiewicz, Alden Dima, and Sarah Lukens. Technical language processing: Unlocking maintenance knowledge. *Manufacturing Letters*, 27:42–46, 1 2021. ISSN 2213-8463. doi:10.1016/J.MFGLET.2020.11.001.

[8] S. Moon, G. Lee, and S. Chi. Automated system for construction specification review using natural language processing. *Advanced Engineering Informatics*, 51:101495, 2022. ISSN 1474-0346. doi:https://doi.org/10.1016/j.aei.2021.101495.

[9] Y. Wang, Y. Zhu, W. Xiong, and C. S. Cai. A few-shot word-structure embedded model for bridge inspection reports learning. *Advanced Engineering Informatics*, 62:102664, 2024. ISSN 1474-0346. doi:10.1016/j.aei.2024.102664.

[10] A. Pal, D. Murguia, and C. Middleton. Automatic inference of construction delays through analysis of weekly progress reports using llms. 2024. ISSN 2706-6568. URL http://itc.scix.net/paper/w78-2024-108.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

[12] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2016. URL https://arxiv.org/abs/1409.0473.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[14] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg. Efficient few-shot learning without prompts, 2022.

[15] NYC Open Data : City of New York. Nyc open data. URL https://tinyurl.com/549hjywn. (accessed 2024-11-15).

[16] S. Casola, I. Lauriola, and A. Lavelli. Pretrained transformers: an empirical comparison. *Machine Learning with Applications*, 9:100334, 2022. ISSN 2666-8270. doi:https://doi.org/10.1016/j.mlwa.2022.100334.

[17] Aristides Milios, Siva Reddy, and Dzmitry Bahdanau. In-context learning for text classification with many labels, 2023. URL https://arxiv.org/abs/2309.10954.

[18] Aleksandra Edwards and Jose Camacho-Collados. Language models for text classification: Is in-context learning enough?, 2024. URL https://arxiv.org/abs/2403.17661.