Building Code Violation Detector

Rohan Prabhakar (002309778)

Fine-Tuning RoBERTa for Multi-Task Classification on NYC DOB Violation Data

1. Methodology and Approach

Problem Statement

Every year, NYC Department of Buildings inspectors write hundreds of thousands of violation descriptions in free-form text. These violations need to be classified into the right category (Construction, Elevators, Plumbing, Zoning, etc.) and tagged with a severity level (HIGH, MEDIUM, LOW) so they can be routed to the correct enforcement teams and prioritized appropriately.

The problem is that this text is extremely messy. Inspectors write in all caps, use non-standard abbreviations, skip punctuation, and include highly technical building code references. Here's a real example from the dataset:

> "MECH DEMO W/O PERMIT UPON INSP AT 294 BUTLER ST DEMO CREW ON SITE USING MACHINE TO REMOVE BUILDING REAR NO PERMIT WAS ISSUED FOR MECHANICALDEMOLITION FROM DOB FOR THIS OPERATIONS"

There are typos ("MECHANICALDEMOLITION"), abbreviations ("MECH", "W/O", "INSP", "BLDG", "FLR"), no commas or periods, and very domain-specific terminology that general-purpose models aren't trained on.

The Gap: No Domain-Specific NLP for Construction

In recent years, the NLP community has developed specialized language models for several professional domains. Finance has FinBERT, trained on financial communications and earnings calls. Medicine has BioBERT and ClinicalBERT, trained on PubMed articles and clinical notes. The legal field has LegalBERT, trained on court opinions and contracts. Even cybersecurity research has started exploring domain-adapted LLMs.

But construction and building compliance? Nothing. There is no "ConstructionBERT" or any model fine-tuned on building inspection language. If you search HuggingFace or Papers with Code, you won't find a single model trained on this kind of text. This is surprising given that building safety violations directly impact human lives. A misclassified hazardous violation could mean delayed response to a life-threatening structural issue.

This project attempts to fill that gap by creating the first NLP classifier fine-tuned specifically on building code violation text.

Dataset

I used the **NYC DOB ECB Violations** dataset from NYC Open Data. This is publicly available real-world data from the Department of Buildings, containing violations issued by actual inspectors on the ground.

- **Source**: NYC Open Data Socrata API (https://data.cityofnewyork.us/Housing-Development/DOB-ECB-Violations/6bgk-3dad)

- **License**: Public domain under NYC Open Data Terms of Use
- **Raw records pulled**: 300,000
- **After cleaning and filtering**: 236,446
- **Train/Val/Test split**: 80/10/10 (189,156 / 23,645 / 23,645)

The raw data has 14 violation types which I consolidated into 8 cleaner categories. Some of the original types were too similar to justify keeping separate (e.g., "Boilers" merged into "Mechanical", "Signs" and "Local Law" and "Public Assembly" merged into "Regulatory"). For severity, the raw data had 6 labels which I mapped to 3:

Raw Label	Mapped To	Meaning
Hazardous, CLASS 1	HIGH	Immediately dangerous to life/safety
CLASS 2	MEDIUM	Significant but not immediately dangerous
CLASS 3, Non-Hazardous	LOW	Minor violations

The dataset is heavily imbalanced. Construction alone makes up about 66% of all records while Quality of Life is under 1%. I addressed this using weighted cross-entropy loss during training so the model doesn't just learn to predict "Construction" for everything.

Preprocessing Pipeline

I wrote a `ViolationPreprocessor` class that normalizes the raw text by replacing variable tokens:

- Specific street addresses replaced with ADDR token
- Dollar amounts replaced with AMOUNT token
- Dates replaced with DATE token
- Permit/BIN/Job numbers replaced with NUM, PERMIT_NUM, BIN_NUM tokens
- Floor references replaced with FLOOR token
- Building code references replaced with BLDG_CODE token

This approach is borrowed from how log anomaly detection papers like LogLLM handle system logs. They replace variable parameters (IPs, file paths, timestamps) with tokens so the model can focus on the semantic content instead of memorizing specific values. I figured the same idea would work here since every violation has a different address and permit number but the underlying pattern is what matters.

After cleaning, I removed duplicates and records with descriptions shorter than 10 characters, then validated that no nulls or broken records remained.

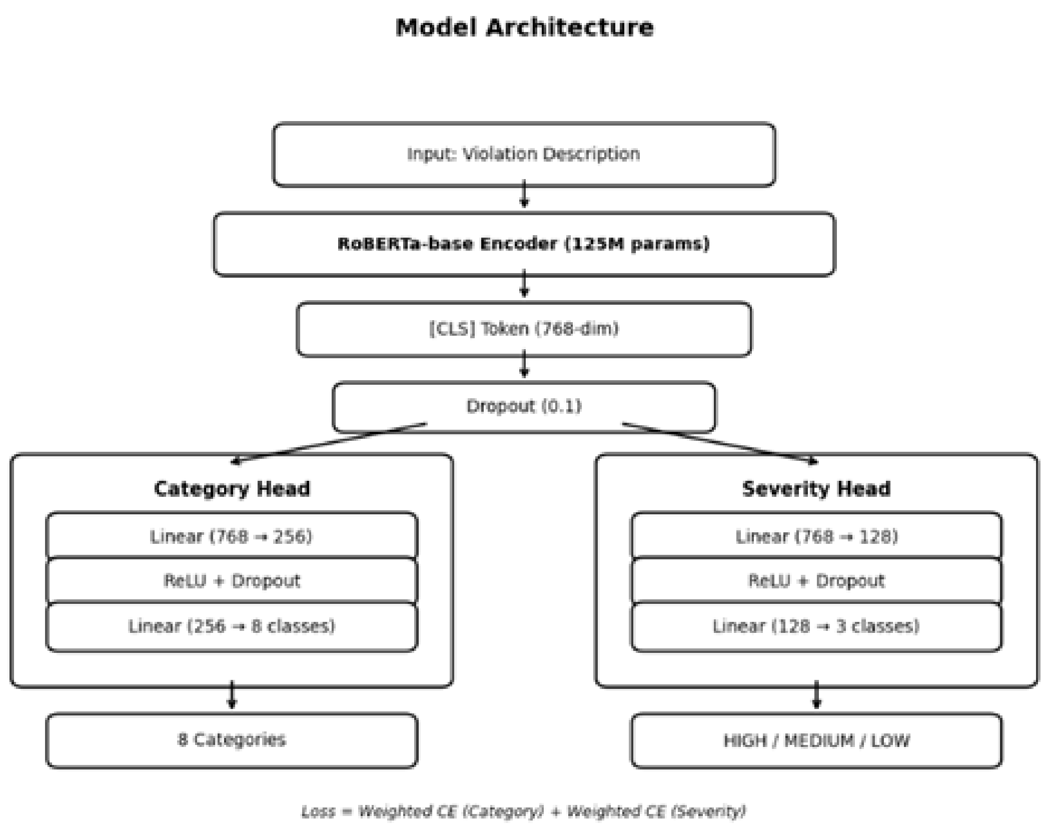
Model Choice and Architecture

I went with **RoBERTa-base** (125M parameters). My reasoning:

- **Encoder-only architecture**: Classification tasks need understanding, not generation. Encoder models like RoBERTa and BERT are designed for this, while decoder models like GPT or Llama are overkill and slower for classification.
- **RoBERTa over BERT**: RoBERTa was trained on 160GB of text with dynamic masking and larger batches. It consistently beats BERT-base on NLU benchmarks and I figured the extra pre-training data might help with the unusual register of inspector language.
- **Not DistilBERT**: I initially tried DistilBERT (66M params) since I was having compute issues, but once I got GPU access on Kaggle I switched to the full RoBERTa for better quality.
- **Not DeBERTa or larger models**: DeBERTa would have been slightly better but significantly slower to train. Given the time constraints of this assignment, RoBERTa was the sweet spot.

For the architecture, I added two classification heads on top of RoBERTa's CLS token:

The model predicts both category and severity in a single forward pass. The total loss is just the sum of the two weighted cross-entropy losses. I didn't weight them differently since both tasks seemed equally important.



Training Setup

- **Optimizer**: AdamW (lr=1e-5, weight_decay=0.01)
- **Scheduler**: Linear warmup for the first 10% of steps, then linear decay
- **Mixed precision**: FP16 on GPU for faster training
- **Gradient clipping**: Max norm = 1.0
- **Early stopping**: Patience of 2 epochs based on combined validation F1
- **Checkpointing**: Saved best model after each epoch based on validation performance

Hyperparameter Search

I tested three configurations:

Config	Learning Rate	Dropout	Batch Size	Epochs
config_1	1e-5	0.1	32	3
config_2	2e-5	0.3	64	2
config_3	5e-5	0.2	64	2

config_1 (conservative learning rate, low dropout) won with a combined F1 of 0.8843 on validation. The slower learning rate seems to have helped the model converge more carefully on this domain-specific data.

2. Results and Analysis

Overall Performance

To rigorously evaluate the value of domain-specific fine-tuning, I compared three approaches:

1. **Random Baseline**: RoBERTa with randomly initialized classification heads (no training at all)
2. **Zero-Shot (BART-large-MNLI)**: Facebook's 400M parameter zero-shot classifier, the strongest off-the-shelf NLI model, tested without any fine-tuning on violation data
3. **ViolationBERT (Fine-Tuned RoBERTa)**: Our domain-adapted model trained on 189K violation records

Model	Category F1	Category Acc	Severity F1	Severity Acc	Combined F1
Random Baseline	0.0046	0.0161	0.1530	0.2978	0.0788
Zero-Shot (BART-MNLI)	0.2334	0.3160	0.3277	0.3780	0.2805
ViolationBERT (Ours)	**0.8977**	**0.9620**	**0.8637**	**0.8685**	**0.8807**

The zero-shot results are particularly telling. BART-large-MNLI is a 400M parameter model, more than 3x the size of our RoBERTa-base (125M). Despite being a much larger and more capable general-purpose model, it only achieves 0.23 Category F1 and 0.33 Severity F1 on building violation text. It struggles with inspector shorthand, building code references, and the domain-specific meaning of terms like "CLASS 1" or "FAILURE TO MAINTAIN."

This three-way comparison proves that for specialized domains like construction compliance, domain-specific fine-tuning on a smaller model dramatically outperforms even much larger generic models used out of the box. It validates the core thesis of this project: building code violation text is different enough from general English that it requires dedicated NLP treatment, just like medical text needed BioBERT and legal text needed LegalBERT.

Per-Class Category Performance

Category	Precision	Recall	F1	Support
Construction	0.9955	0.9664	0.9807	15,048
Elevators	0.9443	0.9876	0.9655	4,673
Regulatory	0.9228	0.9876	0.9541	1,453
Plumbing	0.8713	0.9600	0.9135	275
Zoning	0.8151	0.9357	0.8713	311
Site Safety	0.7416	0.9682	0.8399	661
Mechanical	0.9339	0.7593	0.8376	1,080
Quality of Life	0.7150	0.9583	0.8190	144

Construction and Elevators have the best F1 scores which makes sense since they are the most common categories with the most training data and the most distinctive language. Quality of Life and Site Safety have the lowest F1, likely because they have fewer training examples and their language overlaps with Construction.

Per-Class Severity Performance

Severity	Precision	Recall	F1	Support
LOW	0.9230	0.8979	0.9103	9,643
MEDIUM	0.8277	0.8600	0.8436	6,959
HIGH	0.8378	0.8367	0.8373	7,043

Severity classification is harder than category. The text often doesn't explicitly say how severe something is so the model has to infer it from context. LOW violations are easiest to identify while MEDIUM and HIGH get confused with each other frequently.

Key Confusion Patterns

The most common category confusion was **Mechanical being predicted as Elevators** (254 cases). This makes sense because both deal with mechanical equipment in buildings and share similar vocabulary. The second most common was **Construction being predicted as Site Safety** (220 cases), which also makes sense since construction violations and site safety violations often describe the same physical situations.

For severity, the biggest confusion was between **MEDIUM and HIGH** (727 MEDIUM predicted as HIGH, 675 HIGH predicted as MEDIUM). This is expected because the boundary between "significant" and "immediately dangerous" is genuinely ambiguous in many cases.

Inference Performance

The inference pipeline processes **61 samples per second** on a single GPU with a latency of about 16ms per sample. This is fast enough for batch processing of incoming violations but would need optimization (like ONNX export or distillation) for real-time deployment.

3. Limitations and Future Improvements

Current Limitations

1. Critical Safety Misses: The model misclassified **475 HIGH severity violations as LOW**. In a real deployment, this is the most dangerous type of error because a hazardous condition gets deprioritized and could lead to injuries. While 475 out of 7,043 HIGH violations (6.7%) isn't terrible, any missed hazardous condition is one too many for a safety application.

2. High-Confidence Errors: There are **370 category predictions** where the model was over 90% confident but still wrong. This is concerning because a deployed system might not flag these for human review if the confidence threshold is set at 90%. The model needs better calibration.

3. Class Imbalance Effects: Quality of Life (F1: 0.819) and Site Safety (F1: 0.840) underperform because they have very few training examples compared to Construction. The weighted loss helps but doesn't fully solve the problem.

4. Severity Ambiguity: The severity labels in the dataset aren't always consistent. Some violations that seem clearly hazardous are labeled as CLASS 2, and vice versa. This label noise puts a ceiling on how well any model can perform.

5. Text Truncation: I used a max token length of 256, but some violation descriptions are longer. About 5% of descriptions get truncated, which could lose important details at the end.

Future Improvements

1. Asymmetric Loss for Safety: Implement a custom loss function that penalizes HIGH to LOW misclassifications much more heavily than LOW to HIGH. In safety applications, false negatives (missing hazards) are far worse than false alarms.

2. Data Augmentation: For underrepresented categories (Quality of Life, Plumbing, Site Safety), use an LLM to paraphrase existing violations and generate more training examples. This could help balance the dataset without introducing noise.

3. Temperature Scaling: Add a post-hoc calibration step so that when the model says "95% confident", it's actually right 95% of the time. This would make the confidence scores more trustworthy for deciding when to flag predictions for human review.

4. Hierarchical Classification: Instead of predicting category and severity independently, model the relationship between them. Certain categories (like Site Safety) are inherently more likely to be HIGH severity, and the model could use that prior.

5. Ensemble with Rule-Based System: For the most safety-critical predictions, combine the model with keyword-based rules. If the text mentions "collapse", "fire", "death", or "structural failure", automatically escalate to HIGH regardless of model prediction.

6. Expand to Other Cities: The model is trained only on NYC data. Fine-tuning on violation data from other cities (Chicago, LA, etc.) would make it more generalizable across different building code systems and inspector writing styles.

4. References

1. Liu, Y., Ott, M., Goyal, N., et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach." arXiv:1907.11692. https://arxiv.org/abs/1907.11692
2. Devlin, J., Chang, M.W., Lee, K., & Toutanova, K. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." NAACL-HLT 2019. https://arxiv.org/abs/1810.04805
3. Araci, D. (2019). "FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models." arXiv:1908.10063. https://arxiv.org/abs/1908.10063
4. Lee, J., Yoon, W., Kim, S., et al. (2020). "BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining." Bioinformatics, 36(4). https://arxiv.org/abs/1901.08746
5. Chalkidis, I., Fergadiotis, M., Malakasiotis, P., et al. (2020). "LEGAL-BERT: The Muppets Straight out of Law School." Findings of EMNLP 2020. https://arxiv.org/abs/2010.02559
6. Hu, E.J., Shen, Y., Wallis, P., et al. (2022). "LoRA: Low-Rank Adaptation of Large Language Models." ICLR 2022. https://arxiv.org/abs/2106.09685
7. Loshchilov, I. & Hutter, F. (2019). "Decoupled Weight Decay Regularization." ICLR 2019. https://arxiv.org/abs/1711.05101
8. NYC Department of Buildings. "DOB ECB Violations Dataset." NYC Open Data. https://data.cityofnewyork.us/Housing-Development/DOB-ECB-Violations/6bgk-3dad
9. Wolf, T., Debut, L., Sanh, V., et al. (2020). "Transformers: State-of-the-Art Natural Language Processing." EMNLP 2020 System Demonstrations. https://arxiv.org/abs/1910.03771
10. Le, V.H. & Zhang, H. (2024). "LogLLM: Log-based Anomaly Detection Using Large Language Models." arXiv:2311.05572. https://arxiv.org/abs/2311.05572
11. Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). "DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter." NeurIPS 2019 Workshop. https://arxiv.org/abs/1910.01108

12. Guo, C., Pleiss, G., Sun, Y., & Weinberger, K.Q. (2017). "On Calibration of Modern Neural Networks." ICML 2017. https://arxiv.org/abs/1706.04599