

¹ LabelFusion: Learning to Fuse LLMs and Transformer Classifiers for Robust Text Classification

³ Michael Schlee¹, Christoph Weisser¹, Timo Kivimäki², Melchizedek
⁴ Mashiku⁴, and Benjamin Saefken³

⁵ 1 Centre for Statistics, Georg-August-Universität Göttingen, Germany ² Department of Politics and
⁶ International Studies, University of Bath, Bath, UK ³ Institute of Mathematics, Clausthal University of
⁷ Technology, Clausthal-Zellerfeld, Germany ⁴ Tanaq Management Services LLC, Contracting Agency to
⁸ the Division of Viral Diseases, Centers for Disease Control and Prevention, Chamblee, Georgia, USA

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Open Journals](#) ↗

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#)).²¹

⁹ Summary

¹⁰ LabelFusion is a fusion ensemble for text classification that learns to combine a traditional
¹¹ transformer-based classifier (e.g., RoBERTa) with one or more Large Language Models (LLMs)
¹² such as OpenAI GPT, Google Gemini, or DeepSeek) to deliver accurate and cost-aware predictions
¹³ across multi-class and multi-label tasks. The package provides a simple high-level interface
¹⁴ (AutoFusionClassifier) that trains the full pipeline end-to-end with minimal configuration,
¹⁵ and a flexible API for advanced users. Under the hood, LabelFusion concatenates vector signals
¹⁶ from the ML backbone (logits) and LLM(s) (per-class scores) and trains a compact multi-layer
¹⁷ perceptron (FusionMLP) to produce the final prediction. This learned fusion approach captures
¹⁸ complementary strengths of LLM reasoning and traditional transformer-based classifiers,
¹⁹ yielding robust performance across domains—achieving 92.4% accuracy on AG News topic
²⁰ classification—while enabling practical trade-offs between accuracy, latency, and cost.

²¹ Statement of Need

²² Modern text classification spans diverse scenarios—from sentiment analysis to complex topic
²³ tagging—often under constraints that vary per deployment (throughput, cost ceilings, data
²⁴ privacy). While transformer classifiers such as BERT/RoBERTa achieve strong supervised
²⁵ performance ([Devlin et al., 2018](#); [Liu et al., 2019](#)), frontier LLMs can excel in low-data,
²⁶ ambiguous, or cross-domain settings ([OpenAI, 2023](#)). No single model family is typically
²⁷ uniformly best: LLMs are powerful, but comparatively costly, whereas fine-tuned transformers
²⁸ are efficient but may struggle with out-of-distribution cases.

²⁹ LabelFusion addresses this gap by: (1) exposing a minimal “AutoFusion” interface that trains a
³⁰ learned combination of an ML backbone and one or more LLMs; (2) supporting both multi-class
³¹ and multi-label classification; (3) providing a lightweight fusion learner that directly fits on LLM
³² scores and ML logits; and (4) integrating cleanly with existing ensemble utilities. Researchers
³³ and practitioners can therefore leverage LLMs where they add value while retaining the speed
³⁴ and determinism of transformer models.

³⁵ State of the Field

³⁶ In applied NLP, common tools such as scikit-learn ([Pedregosa et al., 2011](#)) and Hugging Face
³⁷ Transformers ([Wolf et al., 2019](#)) offer strong baselines but do not provide a learned fusion of
³⁸ LLMs with supervised transformers. Orchestration frameworks (e.g., LangChain) focus on tool
³⁹ use rather than classification ensembles. LabelFusion contributes a focused, production-minded
⁴⁰ implementation of a small learned combiner that operates on per-class signals from both model
⁴¹ families.

42 Functionality and Design

43 LabelFusion consists of three layers:

- 44 ■ ML component: a RoBERTa-style classifier produces per-class logits for input texts.
- 45 ■ LLM component(s): provider-specific classifiers (OpenAI, Gemini, DeepSeek) return
- 46 per-class scores via prompting. Scores can be cached to minimize API calls when cache
- 47 locations are provided.
- 48 ■ Fusion component: a compact MLP concatenates ML logits and LLM scores and outputs
- 49 fused logits. The ML backbone is trained/fine-tuned with a small learning rate; the fusion
- 50 MLP uses a higher rate, enabling rapid adaptation without destabilizing the encoder.

51 Key features:

- 52 ■ **Multi-class and multi-label support** with consistent data structures and unified training
- 53 pipeline.
- 54 ■ **Optional LLM response caching** reuses on-disk predictions when cache paths are supplied,
- 55 with dataset-hash validation to guard against stale files.
- 56 ■ **Batched scoring** processes multiple texts efficiently with configurable batch sizes for both
- 57 ML tokenization and LLM API calls.
- 58 ■ **Results management** via ResultsManager tracks experiments, stores predictions, com-
- 59 putes metrics, and enables reproducible research workflows.
- 60 ■ **Flexible interfaces**: Command-line training via `train_fusion.py` with YAML configs for
- 61 research; or minimal AutoFusion API for quick deployment.
- 62 ■ **Composable design**: LabelFusion can serve as a strong base learner in higher-level
- 63 ensembles (e.g., voting/weighted combinations of multiple fusion models).

64 Formally, multi-class classification assigns each input $x \in \mathcal{X}$ to exactly one label among K
 65 mutually exclusive classes:

$$f_{\text{mc}} : \mathcal{X} \rightarrow \{1, \dots, K\}.$$

66 In contrast, multi-label classification predicts a subset of relevant classes, represented as a
 67 binary indicator vector $\mathbf{y} \in \{0, 1\}^K$, where $y_k = 1$ denotes membership in class k :

$$f_{\text{ml}} : \mathcal{X} \rightarrow \{0, 1\}^K.$$

68 Minimal Example (AutoFusion)

```
from textclassify import AutoFusionClassifier

config = {
    'llm_provider': 'deepeek',
    'label_columns': ['positive', 'negative', 'neutral']
}

clf = AutoFusionClassifier(config)
clf.fit(train_dataframe)           # trains ML backbone, gathers LLM scores, fits fusi
pred = clf.predict(["This is amazing!"]) # fused prediction
```

69 CLI and Configuration

70 Users can generate a starter config and train via the command line:

- 71 ■ Create config: `python train_fusion.py --create-config fusion_config.yaml`
- 72 ■ Train: `python train_fusion.py --config fusion_config.yaml`
- 73 ■ Optional test data and output artifacts are also supported.

74 Quality Control

75 The repository ships legacy unit tests under `tests/evaluation/old/` that cover configuration
 76 handling, core types, and package integration. Fusion-specific logic is currently exercised
 77 through CLI-driven workflows and notebooks that run end-to-end training with deterministic
 78 seeds where applicable.

79 Evaluation scripts (`tests/evaluation/`) provide comprehensive benchmarking on standard
 80 datasets: - **AG News** (Zhang et al., 2015): 4-class topic classification with experiments
 81 across varying training data sizes (20%–100%) - **GoEmotions** (Demszky et al., 2020): 28-class
 82 multi-label emotion classification for validating multi-label fusion performance

83 LLM scoring paths implement retries and disk caching; transformer training supports standard
 84 sanity checks (overfit a small batch, reduced batch sizes for constrained hardware). Metrics
 85 (accuracy/F1, per-label scores) are computed automatically and stored with run artifacts to
 86 facilitate regression tracking and reproducibility.

87 Availability and Installation

88 LabelFusion is distributed as part of the `textclassify` package under the MIT license and
 89 is available at <https://github.com/DataandAIResearch/LabelFusion>. The fusion components
 90 require Python 3.8+ and common scientific Python dependencies (PyTorch, transformers,
 91 scikit-learn, numpy, pandas, PyYAML). Optional plotting depends on matplotlib/seaborn.
 92 Installation and quick-start snippets are provided in the `README` and `FUSION_README.md`.

93 Production-Ready Features

94 Beyond the core fusion methodology, LabelFusion includes features for practical deployment:

- 95 **▪ LLM Response Caching:** Optional disk-backed caches reuse prior predictions when cache
 96 paths are supplied, with dataset hashes to flag inconsistent inputs.
- 97 **▪ Results Management:** Built-in `ResultsManager` tracks experiments, stores predictions,
 98 and computes metrics automatically. Supports comparison across runs and configuration
 99 tracking.
- 100 **▪ Batch Processing:** Efficient batched scoring of texts with configurable batch sizes for
 101 both ML and LLM components.

102 Impact and Use Cases

103 Empirical Performance

104 LabelFusion has been evaluated on standard benchmark datasets to validate its effectiveness.
 105 Key findings demonstrate consistent improvements over individual model components:

106 AG News Topic Classification

107 Evaluation on the AG News dataset (Zhang et al., 2015) (4-class topic classification) with
 108 5,000 test samples shows:

Training Data	Model	Accuracy	F1-Score	Precision	Recall
20% (800)	Fusion	92.2%	0.922	0.923	0.922
20% (800)	RoBERTa	89.8%	0.899	0.902	0.898
20% (800)	OpenAI	84.4%	0.844	0.857	0.844
40% (1,600)	Fusion	92.2%	0.922	0.924	0.922
40% (1,600)	RoBERTa	91.0%	0.911	0.913	0.910
40% (1,600)	OpenAI	84.4%	0.844	0.857	0.844
100% (4,000)	Fusion	92.4%	0.924	0.926	0.924

Training Data	Model	Accuracy	F1-Score	Precision	Recall
100% (4,000)	RoBERTa	92.2%	0.922	0.923	0.922
100% (4,000)	OpenAI	84.4%	0.844	0.857	0.844

109 **Key Observations:** - Fusion consistently outperforms individual models across all training
 110 data sizes - With only 20% training data, Fusion achieves 92.2% accuracy—matching its
 111 performance with full data - Demonstrates superior **data efficiency**: fusion learning extracts
 112 maximum value from limited examples - RoBERTa alone requires 100% of data to approach
 113 Fusion's 20% performance - LLM (OpenAI) shows stable but lower performance, highlighting
 114 the value of combining approaches

115 These results validate that learned fusion captures complementary strengths: the LLM provides
 116 robust reasoning even with limited training data, while the ML backbone adds efficiency and
 117 domain-specific patterns.

118 Application Domains

119 Learned fusion excels in scenarios where model strengths complement each other:

- 120 ▪ **Customer feedback analysis** with nuanced multi-label taxonomies where LLMs handle
 ambiguously sentiment while ML models efficiently process clear cases
- 121 ▪ **Content moderation** where uncertain cases benefit from LLM reasoning while rou-
 tine items rely on the fast ML backbone, enabling real-time processing with accuracy
 guarantees
- 122 ▪ **Scientific literature classification** across heterogeneous topics where domain shift is
 common and LLMs provide robustness to new terminology
- 123 ▪ **Low-resource settings** where limited training data is available but task complexity requires
 sophisticated reasoning

124 The approach enables pragmatic cost control (e.g., the fusion layer learns when to rely more
 125 heavily on the efficient ML backbone versus the more expensive LLM signal) while retaining a
 126 single trainable decision surface that optimizes for the specific deployment constraints.

127 Acknowledgements

128 We thank contributors and users who reported issues and shared datasets. LabelFusion builds on
 129 the open-source ecosystem, notably Hugging Face Transformers ([Wolf et al., 2019](#)), scikit-learn
 130 ([Pedregosa et al., 2011](#)), PyTorch ([Paszke et al., 2019](#)), and LLM provider SDKs. The work
 131 presented in this paper was conducted independently by the author Melchizedek Mashiku and
 132 is not affiliated with Tanaq Management Services LLC, Contracting Agency to the Division
 133 of Viral Diseases, Centers for Disease Control and Prevention, Chamblee, Georgia, USA. We
 134 acknowledge the use of the AG News and GoEmotions benchmark datasets for evaluation.

135 References

- 136 Demszky, D., Movshovitz-Attias, D., Ko, J., Cowen, A., Nemade, G., & Ravi, S. (2020).
 137 GoEmotions: A dataset of fine-grained emotions. *Proceedings of the 58th Annual Meeting*
 138 *of the Association for Computational Linguistics*, 4040–4054.
- 139 Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep
 140 bidirectional transformers for language understanding. *arXiv Preprint arXiv:1810.04805*.
- 141 Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer,
 142 L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach.
 143 *arXiv Preprint arXiv:1907.11692*.
- 144 OpenAI. (2023). GPT-4 technical report. *arXiv Preprint arXiv:2303.08774*.

- 150 Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin,
151 Z., Gimelshein, N., Antiga, L., & others. (2019). PyTorch: An imperative style, high-
152 performance deep learning library. *Advances in Neural Information Processing Systems*,
153 32.
- 154 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
155 Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning
156 in python. *Journal of Machine Learning Research*, 12(Oct), 2825–2830.
- 157 Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T.,
158 Louf, R., Funtowicz, M., & others. (2019). HuggingFace's transformers: State-of-the-art
159 natural language processing. *arXiv Preprint arXiv:1910.03771*.
- 160 Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text
161 classification. *Advances in Neural Information Processing Systems*, 28, 649–657.