LLM-EmBEditor: Universal Base Editing Efficiency Prediction via Large Language Model Embeddings

Lucas Schneider
Department of Computer Science, University of Oxford
Oxford, UK
lucas.schneider@cs.ox.ac.uk

Peter Minary
Department of Computer Science, University of Oxford
Oxford, UK
peter.minary@cs.ox.ac.uk

ABSTRACT

Base editing efficiency varies dramatically across experiments, creating critical bottlenecks for therapeutic applications. While current specialized computational models achieve good performance on individual base editor types, they require extensive feature engineering, custom architectures, and editor-specific optimization that limit practical applicability across diverse base editing systems. We introduce LLM-EmBEditor, a holistic base editing efficiency rate prediction model that leverages Large Language Models (LLMs). By encoding base editing features as comma-separated key-value strings, our method extracts rich contextual embeddings and trains lightweight regression heads, eliminating both specialized architectures and manual feature engineering while seamlessly integrating sequence and numerical features through natural language formatting.

To our knowledge, our model, LLM-EmBEditor, is the first to successfully leverage mixed base editor datasets, enabling effective transfer learning across different base editor types (ABE and CBE), while existing models are limited to a single editor type. This cross-editor generalization capability allows LLM-EmBEditor to achieve strong performance, outperforming traditional benchmarks by 13.5% in Pearson's R and 18.8% in Spearman's ρ on the ABE combined dataset (Pearson's $R = 0.717$, Spearman's $\rho = 0.797$ vs. $R = 0.632$, $\rho = 0.671$ for FORECasT-BE) and achieving competitive performance on the CBE combined dataset ($R = 0.662$, $\rho = 0.689$ vs. $R = 0.739$, $\rho = 0.816$ for igRNA-ABE), while uniquely providing predictions across all editor types with $R = 0.705$, $\rho = 0.775$, and $R^2 = 0.496$. Our ablation study confirms the robustness of our approach across multiple model components, such as transformer architecture, pooling strategy, and model size.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Neural networks**; • **Applied computing** → **Bioinformatics**.

KEYWORDS

base editing, large language models, embedding-based regression, LLM embeddings, transformer architectures, pooling strategies, base editing efficiency prediction, machine learning

1 INTRODUCTION

Base editing is a genome engineering approach that allows precise editing of single nucleotides and has the potential to cure rare diseases caused by point mutations. Compared to traditional CRISPR-Cas9 methods, base editing does not introduce double-stranded breaks and therefore reduces the risk of unwanted mutations [6]. There are two types of base editors: Adenine Base Editors (ABEs), which convert an adenine base (A) into a guanine base (G), and Cytosine Base Editors (CBEs), which convert a cytosine base (C) into a thymine base (T).

The outcome of base editing is not easy to predict as it depends not only on the underlying type but also on a wide range of factors, including the guide RNA sequence, the target sequence and cellular conditions [2]. Furthermore, testing base editors in a lab is resource intense and time consuming. Therefore, computational models have been developed to help practitioners design efficient base editors, such as DeepBaseEditor [13], FORECasT-BE [9], BEDICT-V2 [5], DeepBE [4], and igRNA-ABE [7]. These models predict base editing efficiency, defined as the probability that a base editor successfully introduces nucleotide changes at the target site. Efficiency rate prediction serves as a critical screening step, enabling researchers to filter out base editors early in their design pipeline.

Existing base editing efficiency models have two critical weaknesses: the reliance on manual feature engineering, which is difficult in genomic contexts and may lead to omitting crucial information in the process, and the lack of generalization across different editing systems. Current approaches develop models in isolation for ABE and CBE editor classes, often targeting only individual base editors [4, 5, 7, 9, 13]. This prevents efficient transfer learning between different base editor types.

Recent studies demonstrate that large language models (LLMs) can be utilized for numerical regressions to build well-performing regression models [8, 14, 15]. The RAFT method by Lukasik et al. [8] and the OmniPred framework by Song et al. [14] show that fine-tuned LLMs can outperform traditional models across diverse tasks, from sentiment scoring to molecular property prediction. Besides fine-tuning, LLMs can produce compact embeddings of input features that serve as effective inputs to regression heads. Tang et al. [15] show these representations can capture complex relationships among features. Chen and Zou [1] demonstrate that GPT-3.5-based gene embeddings efficiently encode single-cell biology, matching or surpassing models pre-trained on gene-expression profiles from millions of cells. Moreover, Sadeghi et al. [11] find that embeddings derived from SMILES-encoded molecules using LLaMA and GPT rival domain-specific models for drug-drug interaction and property prediction. Finally, Zhang and Yang [18] introduce PolyLLMem, which fuses LLaMA-3 text embeddings with Uni-Mol structural embeddings and matches or exceeds models pre-trained on millions of samples.

To address the shortcomings of the current base editing efficiency models, we propose a novel embedding-based regression framework, called LLM-EmBEditor, that leverages frozen large language model representations for base editing efficiency prediction. Our approach eliminates the need for manual feature engineering

by presenting biological features as plain-text key-value pairs. This also allows us to fuse ABE and CBE datasets together, resulting in a holistic model. Our model outperforms traditional benchmarks by 13.5% in Pearson's R and 18.8% in Spearman's ρ on the ABE combined dataset and achieves competitive performance across all base editing systems. Uniquely, our approach enables the fusion of multiple editor types for successful prediction on the all editors dataset, a capability beyond the scope of traditional methods.

The remainder of the manuscript is organized as follows. Section 2 introduces our methodology, LLM-EmBEditor architecture, base editing dataset, and benchmark models. Section 3 presents the results and discusses their implications. Finally, Section 4 offers concluding remarks and points to future research directions.

2 METHODOLOGY

2.1 Theoretical Framework

Inspired by the framework of Tang et al. [15], designed to understand LLM embeddings in regression tasks, we propose to employ frozen pre-trained language models as feature extractors for base editing efficiency prediction. Instead of manually engineering features, we leverage the rich representational capacity of LLMs to generate efficient embeddings to capture complex relationships between input features (gRNA sequences, target sequences, melting temperatures, energy terms, etc.). We then train lightweight regression heads for efficiency prediction on the obtained embeddings.

Our approach consists of five sequential steps:

(1) **String Representation:** We encode multimodal data, including DNA sequences, as comma-separated key-value pairs following the format used by Tang et al. [15]. The following example shows an excerpt of our comma-separated key-value string representation for efficiency prediction:

> *grna:GGTCCAAGCCTGTTCGATTA,*
> *pam_sequence:AGG, cell:HEK293T,*
> *base_editor:ABE20m,*
> *energy_7:7.2350, energy_8:12.2549,*
> *free_energy:-1.6000,*
> *melt_temperature_grna:51.8152*

(2) **Tokenization:** The string representation x is tokenized into L tokens using the model's vocabulary $x \to \{t_1, t_2, \ldots, t_L\}$. For example, a byte-pair encoding (BPE) tokenizer converts the string representation above to the following tokens:

> *[901, 3376, 25, 22254, 51, 94307, 1890, 34, 1162, 25388, 7749, 38, 828, 15204, 11, 41190, 23735, 25, 1890, 38, 11, 8500, 25, ...]*

(3) **Forward Pass:** We apply the pre-trained transformer to obtain contextualized representations $H \in \mathbb{R}^{T\times d}$, where T is the sequence length and d is the model's hidden dimension.

(4) **Pooling Operation:** From the contextualized token representations $H \in \mathbb{R}^{T\times d}$ with rows $\{h_1, h_2, \ldots, h_T\}$ where $h_t \in \mathbb{R}^d$ denotes the embedding at token position t, we apply attention-weighted pooling to obtain a fixed-size representation $v \in \mathbb{R}^d$:

$$v^{\text{attn}} = \sum_{t=1}^{T} \alpha_t \, h_t, \tag{1}$$

where T is the sequence length, d is the hidden dimension, $m_t \in \{0, 1\}$ is the padding mask (1 = valid token, 0 = padding), $\varepsilon = 10^{-9}$, and the attention weights α_t are computed from the transformer's internal attention patterns via:

(a) Average attention across transformer layers n_ℓ and attention heads n_h:

$$\bar{A}_{q,t} = \frac{1}{n_\ell n_h} \sum_{\ell=1}^{n_\ell} \sum_{h=1}^{n_h} A_{h,q,t}^{(\ell)}, \tag{2}$$

where $A_{h,q,t}^{(\ell)}$ is the attention weight from query token q to key token t at layer ℓ and head h.

(b) Average attention to each token:

$$\tilde{a}_t = \frac{1}{T} \sum_{q=1}^{T} \bar{A}_{q,t}. \tag{3}$$

(c) Apply masking and normalize:

$$\alpha_t = \frac{m_t \, \tilde{a}_t}{\sum_{s=1}^{T} m_s \, \tilde{a}_s + \varepsilon}. \tag{4}$$

(5) **Regression Head:** The pooled embedding $v \in \mathbb{R}^d$ is fed to a trainable regression function to predict the base editing efficiency:

$$\hat{y} = f_\theta(v) \in \mathbb{R}, \tag{5}$$

where f_θ is a parameterized regression function (e.g., MLP, linear layer) with trainable parameters θ.

2.2 Dataset & Preprocessing

We utilize the Be-dataHIVE base editing database [12], which consolidates and standardizes data from multiple high-throughput base editing studies. The dataset includes base editing efficiency measurements across different experimental conditions, gRNA sequences, target sequences, and offers a wide range of features. To mitigate potential experimental batch effects and ensure data consistency, we use data points derived from Pallaseni et al. (database ID 3) for our analysis [9]. We create stratified train/validation/test splits (80%/10%/10%) for efficiency prediction across three dataset configurations:

- **Individual Base Editors:** Single-editor datasets (ABE20m, ABE8e, ABERA, BE4-1, BE4-2, FNLS). ABERA was excluded from individual analysis due to insufficient efficiency variance (mean: 0.002, max: 0.36 vs. 1.00 for other datasets), which precluded reliable model evaluation, but was retained in combined datasets for training diversity.
- **Editor Type Combined:** All single-editor data combined in two datasets split by base editor type: adenine (ABE combined) and cytosine (CBE combined).
- **All Editors:** All data points combined.

To prevent data leakage, we ensure that gRNA, target sequence, and editor triplets are uniquely assigned to a single split, such that no combination appears in multiple splits. Depending on the underlying dataset type, we include additional features, namely, base editor type for the all editors dataset or base editor names for the editor type specific datasets (ABE combined and CBE combined).

2.3 Our Model: LLM-EmBEditor

We propose LLM-EmBEditor[1], a large language model embedding-based regressor for base editing efficiency rates. Our architecture combines frozen LLM representations with a specialized regression head for the final prediction.

We employ the dense decoder-only transformer model Qwen3-8B [17] as our embedding layer to generate contextualized representations from the input variables. To reduce dimensionality to 4096, we apply attention-weighted pooling (see Section 2.1). After obtaining the representations, we normalize the embeddings across the individual data points before training our regression head.

The regression component consists of a simple MLP with four layers: $4096 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 1$. Each hidden layer incorporates batch normalization, ReLU activation for non-linearity, and dropout (p=0.3) for regularization. The final layer applies a sigmoid activation to constrain predictions to the [0,1] efficiency range. The full architecture is illustrated in Figure 1.

To illustrate the robustness of our approach, we explore the effects of different transformer architectures across LLM model families, transformer sizes, and pooling strategies in our ablation studies in Section A.1.

2.4 Benchmark Comparison Against Existing Methods

To evaluate the effectiveness of our LLM embedding-based approach, we compare it against five state-of-the-art computational methods for base editing efficiency prediction from the literature: DeepBaseEditor [13], FORECasT-BE [9], igRNA-ABE [7], BEDICT-V2 [5], and DeepBE [4].

DeepBaseEditor uses a convolutional neural network (CNN) with one-hot encoded nucleotide sequences as input. The architecture employs convolutional layers for pattern recognition followed by fully connected layers for the final prediction. The authors devised CBE and ABE specific architectures. The model takes a 25 nucleotide sequence as input and contains approximately 500K parameters [13].

FORECasT-BE uses gradient boosting regressors with engineered features, including nucleotide composition, positional encoding and melting temperatures. It takes a 20 nucleotide sequence as input and has two model variations: one for ABE editors and one for CBE editors. Despite being a traditional machine learning approach with approximately 1,000 parameters, it demonstrates surprisingly competitive performance across multiple datasets [9].

igRNA-ABE employs a deep convolutional architecture with quaternion-based sequence alignment features (four-dimensional representations of nucleotide sequences) for adenine base editing. The model consists of 5 Conv2D layers with 256 filters each, followed by 3 dense layers ($256 \rightarrow 128 \rightarrow 32 \rightarrow 1$), totaling 3.86M parameters [7].

BEDICT-V2 uses a compact CNN architecture for efficiency prediction. The model employs a single Conv1D layer with kernel size 2, followed by a dense layer. In total, the model has around 193K parameters. It takes a 24 nucleotide sequence (20bp target + 4bp PAM) as input [5].

DeepBE predicts the efficiency rate using a PAM-specific model. The architecture consists of a single Conv1D layer with 2,000 filters (kernel size varying from 7-10 depending on PAM variant), followed by dense layers ($1500 \rightarrow 100 \rightarrow 1$), totaling 90.2M parameters. It takes a 30 nucleotide context sequence as input. DeepBE represents the largest non-LLM baseline in our comparison [4].

All of these models were originally developed using similar underlying data sources. To ensure a fair comparison, we retrain all benchmark models on our exact dataset configurations using their respective architectures. For the all editors dataset, we use a single architecture variant for each benchmark model.

3 RESULTS

We evaluate LLM-EmBEditor against the five state-of-the-art baseline methods across multiple base editor datasets. Table 1 presents the comprehensive comparison of our approach with FORECasT-BE, DeepBaseEditor, igRNA-ABE, BEDICT-V2, and DeepBE across individual base editors and combined datasets. Figure 2 shows the actual versus predicted base editing efficiency scatter plots for the key all editors dataset. The training configurations are reported in A.3.

LLM-EmBEditor achieves strong performance on the all editors dataset, where cross-editor prediction remains beyond the capabilities of traditional editor-specific methods. However, when retraining benchmark models on the all editors dataset, the performance gap becomes apparent: LLM-EmBEditor outperforms the second-best model, FORECasT-BE, by 82.2% in Pearson's R ($R = 0.705$ vs. $R = 0.387$) and 92.8% in Spearman's ρ ($\rho = 0.775$ vs. $\rho = 0.402$). On the ABE combined dataset, LLM-EmBEditor outperforms all baselines with Pearson's $R = 0.717$ (Spearman's $\rho = 0.797$) by 14.2% over DeepBaseEditor ($R = 0.628$) and 13.5% over FORECasT-BE ($R = 0.632$). On the CBE combined dataset, LLM-EmBEditor underperforms slightly compared to igRNA-ABE ($R = 0.739$, $\rho = 0.816$) by 11.6% in Pearson's R and 18.4% in Spearman's ρ, with values of $R = 0.662$ and $\rho = 0.689$. When looking at smaller and less diverse datasets, such as BE4-2 with approximately 7,000 training samples, LLM-EmBEditor performs moderately with a Pearson's $R = 0.442$ and Spearman's $\rho = 0.500$. Overall, specialized benchmark methods achieve higher performance metrics on most individual base editor datasets, with DeepBE-PAM, and FORECasT-BE showing strong performance on single-editor tasks. There are two drivers for this pattern: insufficient training data for complex models to learn robust generalizable patterns and potential overfitting due to the less diverse datasets.

To illustrate the robustness of our approach, we conduct comprehensive ablation studies examining transformer architectures, model sizes, and pooling strategies. The complete ablation study results are provided in Section A.1.

4 CONCLUSION AND FUTURE WORK

We introduce LLM-EmBEditor, a unified framework that achieves strong performance in base-editing efficiency prediction while eliminating manual feature engineering and specialized architectures. Our approach demonstrates that frozen language model representations can effectively capture sequence-function relationships across

[1]Code and models are available at: https://github.com/Lucas749/LLM-EmBEditor

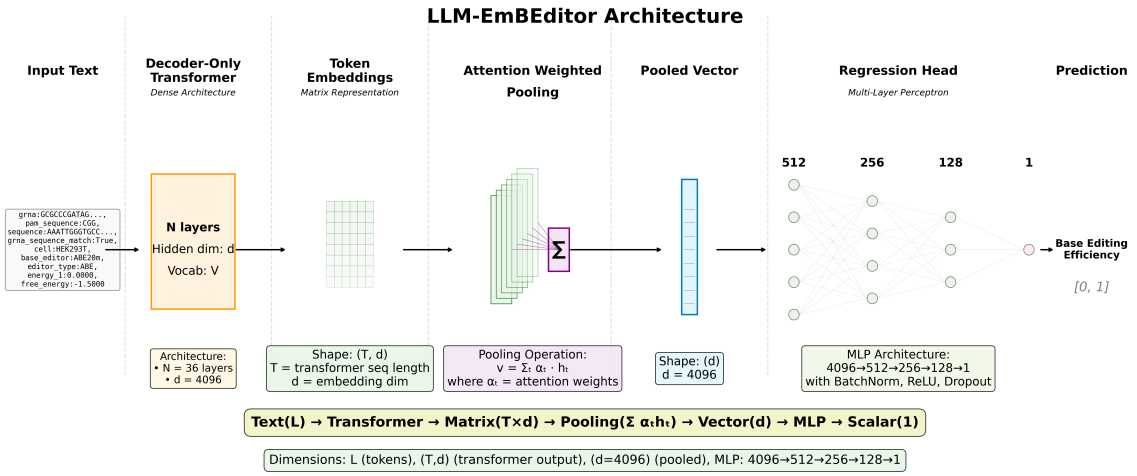
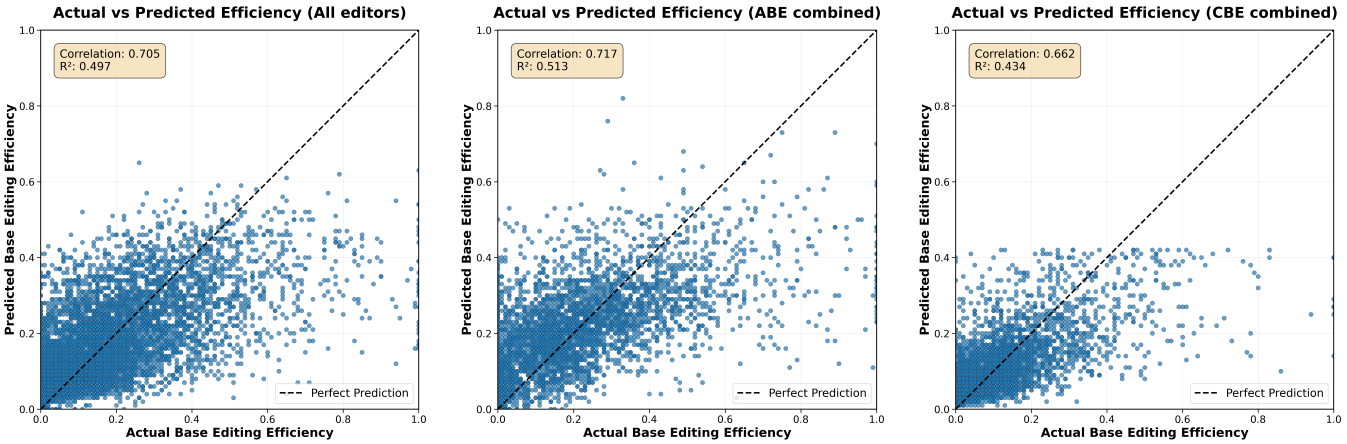


Figure 1: LLM-EmBEditor architecture overview showing the five-step process: (1) string representation of multimodal base editing features, (2) tokenization, (3) forward pass through frozen LLM, (4) attention-weighted pooling to fixed dimension vector, and (5) regression head for efficiency prediction.



(a) All Editors Test Dataset (b) ABE Combined Test Dataset (c) CBE Combined Test Dataset

Figure 2: Actual vs predicted base editing efficiency scatter plots showing model performance across different datasets.

base editing systems and enable efficient domain transfer by combining multiple editor types within a single computational model.

We present several key methodological contributions. First, we systematically apply LLM embeddings to base editing prediction, showing that general-purpose language models encode relevant genomic patterns without domain-specific training, thus eliminating the need for feature engineering. Second, our multimodal text integration uses key-value formatting to combine DNA sequences, experimental conditions, and numerical features into unified representations compatible with standard tokenizers. Third, we evaluate diverse LLM architectures (decoder-only, encoder-decoder, encoder-only), model sizes, and pooling strategies to showcase the robustness of our approach.

Our framework offers significant computational advantages through frozen embeddings representations, enabling rapid experimentation without extensive model retraining. We demonstrate effective embedding-based regression for base editing prediction and show that a single model works across editing systems (ABE, CBE), unlike existing system-specific approaches.

Base editing efficiency represents only one aspect of the computational base editing pipeline. Future research should investigate applying our framework to bystander prediction tasks, which involve forecasting editing outcomes across multiple sequence positions simultaneously.

Table 1: Performance comparison of LLM-EmBEditor against baseline methods across all base editor datasets

Method	All editors	ABE combined	CBE combined	ABE20m	ABE8e	BE4-1	BE4-2	FNLS
Pearson's R								
LLM-EmBEditor	**0.705**	**0.717**	0.662	0.576	0.596	0.474	0.442	0.571
FORECasT-BE	0.387	0.632	0.697	0.607	0.665	0.716	**0.756**	**0.751**
DeepBaseEditor	0.326	0.628	0.644	0.591	0.603	0.645	0.648	0.683
igRNA-ABE	0.332	0.533	**0.739**	0.734	0.736	0.684	0.663	0.708
BEDICT-V2	0.283	0.451	0.511	0.646	0.675	0.553	0.246	0.644
DeepBE-PAM	0.296	0.537	0.713	**0.753**	**0.776**	**0.761**	0.704	0.750
Spearman's ρ								
LLM-EmBEditor	**0.775**	**0.797**	0.689	0.653	0.655	0.524	0.500	0.585
FORECasT-BE	0.402	0.671	0.786	0.650	0.708	0.809	**0.805**	**0.827**
DeepBaseEditor	0.336	0.672	0.717	0.637	0.654	0.677	0.701	0.722
igRNA-ABE	0.354	0.597	0.816	0.801	0.803	0.756	0.728	0.780
BEDICT-V2	0.334	0.609	0.658	0.792	0.798	0.643	0.258	0.773
DeepBE-PAM	0.377	0.602	**0.819**	**0.823**	**0.841**	**0.841**	0.769	0.823
Coefficient of Determination (R^2)								
LLM-EmBEditor	**0.496**	**0.514**	0.434	0.331	0.350	0.206	0.161	0.324
FORECasT-BE	0.149	0.398	0.484	0.368	0.439	0.512	**0.572**	**0.560**
DeepBaseEditor	0.105	0.390	0.142	0.343	0.359	0.276	0.280	0.248
igRNA-ABE	0.102	0.278	**0.538**	0.522	0.532	0.446	0.416	0.470
BEDICT-V2	-0.334	-0.114	-0.587	-0.214	0.111	-0.531	-1.138	-0.141
DeepBE-PAM	-0.020	0.280	0.491	**0.563**	**0.601**	**0.575**	0.469	0.529
Root Mean Squared Error (RMSE)								
LLM-EmBEditor	**0.119**	**0.132**	0.100	0.140	0.170	0.104	0.070	0.135
FORECasT-BE	0.157	0.152	**0.096**	0.136	0.158	**0.081**	**0.050**	**0.109**
DeepBaseEditor	0.158	0.153	0.124	0.139	0.169	0.099	0.065	0.143
igRNA-ABE	0.167	0.176	0.100	0.131	0.151	0.092	0.068	0.140
BEDICT-V2	0.204	0.218	0.185	0.208	0.208	0.154	0.130	0.205
DeepBE-PAM	0.178	0.176	0.105	**0.125**	**0.140**	**0.081**	0.065	0.132

REFERENCES

[1] Yiqun Chen and James Zou. 2025. Simple and effective embedding model for single-cell biology built from ChatGPT. *Nature Biomedical Engineering* 9, 4 (apr 2025), 483–493. https://doi.org/10.1038/s41551-024-01284-6

[2] G. Giner, S. Ikram, M. J. Herold, and A. T. Papenfuss. 2023. A systematic review of computational methods for designing efficient guides for CRISPR DNA base editor systems. *Briefings in Bioinformatics* 24, 4 (2023), bbad205. https://doi.org/10.1093/bib/bbad205

[3] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. 2021. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics* 37, 15 (02 2021), 2112–2120. https://doi.org/10.1093/bioinformatics/btab083 arXiv:https://academic.oup.com/bioinformatics/article-pdf/37/15/2112/57195892/btab083.pdf

[4] Nahye Kim, Sungchul Choi, Sungjae Kim, Myungjae Song, Jung Hwa Seo, Seonwoo Min, Jinman Park, Sung-Rae Cho, and Hyongbum Henry Kim. 2024. Deep learning models to predict the editing efficiencies and outcomes of diverse base editors. *Nature Biotechnology* 42, 3 (March 2024), 484–497. https://doi.org/10.1038/s41587-023-01792-x

[5] Lucas Kissling, Amina Mollaysa, Sharan Janjuha, Nicolas Mathis, Kim F. Marquart, Yanik Weber, Woohyun J. Moon, Paulo J. C. Lin, Steven H. Y. Fan, Hiromi Muramatsu, Máté Vadovics, Ahmed Allam, Norbert Pardi, Ying K. Tam, Michael Krauthammer, and Gerald Schwank. 2025. Predicting adenine base editing efficiencies in different cellular contexts by deep learning. *Genome Biology* 26, 1 (May 2025), 115. https://doi.org/10.1186/s13059-025-03586-7

[6] Alexis C. Komor, Yongjoo B. Kim, Michael S. Packer, John A. Zuris, and David R. Liu. 2016. Programmable editing of a target base in genomic DNA without double-stranded DNA cleavage. *Nature* 533, 7603 (2016), 420–424. https://doi.org/10.1038/nature17946

[7] Bo Li, Xiagu Zhu, Dongdong Zhao, Yaqiu Li, Yuanzhao Yang, Ju Li, Changhao Bi, and Xueli Zhang. 2024. igRNA Prediction and Selection AI Models (igRNA-PS) for Bystander-less ABE Base Editing. *Journal of Molecular Biology* 436, 18 (2024), 168714. https://doi.org/10.1016/j.jmb.2024.168714

[8] Maciej Lukasik, Boris Dadachev, and Andreas Krause. 2025. Better Autoregressive Regression with LLMs via Regression-Aware Fine-Tuning. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=xGs7Ch3Vyo

[9] Ananth Pallaseni, Elin Madli Peets, Jonas Koeppel, Juliane Weller, Thomas Vanderstichele, Uyen Linh Ho, Luca Crepaldi, Jolanda van Leeuwen, Felicity Allen, and Leopold Parts. 2022. Predicting base editing outcomes using position-specific sequence determinants. *Nucleic Acids Research* 50, 6 (03 2022), 3551–3564. https://doi.org/10.1093/nar/gkac161 arXiv:https://academic.oup.com/nar/article-pdf/50/6/3551/43246266/gkac161.pdf

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *CoRR* abs/1910.10683 (2019). arXiv:1910.10683 http://arxiv.org/abs/1910.10683

[11] Shaghayegh Sadeghi, Alan Bui, Ali Forooghi, Jianguo Lu, and Alioune Ngom. 2024. Can large language models understand molecules? *BMC Bioinformatics* 25, 1 (jun 2024), 225. https://doi.org/10.1186/s12859-024-05847-x

[12] Lucas Schneider and Peter Minary. 2024. Be-dataHIVE: A harmonized, multi-assay base editing dataset for benchmarking machine learning approaches. *BMC Bioinformatics* 25, 1 (2024), 1–15. https://doi.org/10.1186/s12859-024-05898-0

[13] Myungjae Song, Hui Kwon Kim, Sungtae Lee, Younggwang Kim, Sang-Yeon Seo, Jinman Park, Jae Woo Choi, Hyewon Jang, Jeong Hong Shin, Seonwoo Min, Zhejiu Quan, Ji Hun Kim, Hoon Chul Kang, Sungroh Yoon, and Hyongbum Henry Kim. 2020. Sequence-specific prediction of the efficiencies of adenine and cytosine base editors. *Nature Biotechnology* 38, 9 (2020), 1037–1043. https://doi.org/10.1038/s41587-020-0573-5

[14] Xingyou Song, Arushi Agarwal, Kevin Swersky, George E Dahl, Golnaz Ghiasi, James Bradbury, Aleksandra Faust, Hugo Larochelle, and Ahmad Beirami. 2024. OmniPred: Language Models as Universal Regressors. *arXiv preprint arXiv:2402.14547* (2024). arXiv:2402.14547 https://arxiv.org/abs/2402.14547

[15] Eric Tang, Bangding Yang, and Xingyou Song. 2024. Understanding LLM Embeddings for Regression. *arXiv preprint arXiv:2411.14708* (2024). arXiv:2411.14708 https://arxiv.org/abs/2411.14708

[16] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. 2024. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. arXiv:2412.13663 [cs.CL] https://arxiv.org/abs/2412.13663

[17] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su,

Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL] https://arxiv.org/abs/2505.09388

[18] Tianren Zhang and Dai-Bei Yang. 2025. Multimodal Machine Learning with Large Language Embedding Model for Polymer Property Prediction. *Chemistry of Materials* 37, 18 (sep 2025), 7002–7013. https://doi.org/10.1021/acs.chemmater.5c00940

A APPENDIX

A.1 Comprehensive Ablation Studies

To assess the robustness of our embedding-based approach and understand the key architectural components driving performance, we conduct comprehensive ablation studies across multiple dimensions. We systematically evaluate how transformer architectures, pooling strategies, and model scale affect base editing efficiency prediction performance, providing insights into the generalizability.

A.1.1 Transformer Design. To understand how different transformer designs affect embedding quality for base editing efficiency predictions, we evaluate four different transformer architectures: Qwen3 [17] representing dense decoder-only models, T5/Flan-T5 [10] as encoder-decoder architecture, ModernBERT [16] for encoder-only designs, and DNABERT-6 [3] as a domain-specific genomic encoder model. For our comparison, we try to use models with similar parameter counts and sizes or in the case of ModernBERT and DNABERT, we use the largest available models. Table 2 presents the parameter counts and embedding dimensions for each model architecture.

Transformer architectures vary fundamentally in their attention mechanisms and training objectives.

- **Decoder-only (Qwen3-8B):** Unidirectional attention optimized for autoregressive generation, trained on diverse text and code with next-token prediction objectives
- **Encoder-decoder (T5-11B):** Bidirectional encoder with autoregressive decoder, trained via span-corruption on text-to-text tasks
- **Encoder-only (ModernBERT Large):** Bidirectional attention throughout, trained on masked language modeling with modern architectural improvements
- **Domain-specific encoder (DNABERT-6):** Bidirectional encoder pre-trained specifically on genomic sequences using masked language modeling with 6-mer tokenization

Table 2: Language model architectures evaluated for base editing efficiency prediction

Model	Architecture	Parameters	Embedding Dimensions
Qwen3-8B	Decoder-only	8.0B	4096
T5-11B	Encoder-decoder	11.0B	1024
ModernBERT Large	Encoder-only	395M	1024
DNABERT-6	Domain-specific encoder	86M	768

Table 3 reports the performance of the different transformer architectures across our eight datasets. All generalized LLM architectures follow the generalization pattern that we discussed in Section 3. For diverse and larger datasets all models perform well while achieving a lower performance for smaller ones. The dense decoder-only transformer shows the most consistent performance and achieves the best scores on all eight datasets. Overall, ModernBERT-Large has the worst performance among general-purpose LLMs and underperforms T5-11B despite having the same dimension size in the last layer, indicating that encoder-only transformer capabilities to generate efficient embeddings are limited.

Table 3: Transformer architecture comparison across all datasets

Architecture	All editors	ABE combined	CBE combined	ABE20m	ABE8e	BE4-1	BE4-2	FNLS
Pearson's R								
Qwen3-8B	**0.705**	**0.717**	**0.662**	**0.576**	**0.596**	**0.474**	**0.442**	**0.571**
ModernBERT Large	0.572	0.600	0.661	0.366	0.377	0.289	0.380	0.323
T5-11B	0.632	0.641	0.661	0.461	0.452	0.340	0.357	0.393
DNABERT-6	0.227	0.175	0.200	0.157	0.111	0.036	-0.052	0.074
Spearman's ρ								
Qwen3-8B	**0.775**	**0.797**	**0.689**	**0.653**	**0.655**	**0.524**	**0.500**	**0.585**
ModernBERT Large	0.573	0.617	0.591	0.439	0.432	0.319	0.378	0.322
T5-11B	0.631	0.673	0.617	0.542	0.519	0.394	0.373	0.439
DNABERT-6	0.213	0.181	0.186	0.188	0.125	0.014	0.007	0.072
Coefficient of Determination (R^2)								
Qwen3-8B	**0.496**	**0.514**	**0.434**	**0.331**	**0.350**	**0.206**	**0.161**	**0.324**
ModernBERT Large	0.327	0.356	0.431	0.091	0.095	-0.003	0.101	0.070
T5-11B	0.396	0.407	0.426	0.178	0.176	0.081	0.088	0.118
DNABERT-6	0.051	0.030	0.036	0.021	0.004	-0.001	-0.055	0.005
Root Mean Squared Error (RMSE)								
Qwen3-8B	**0.119**	**0.132**	**0.100**	**0.140**	**0.170**	**0.104**	**0.070**	**0.135**
ModernBERT Large	0.140	0.157	0.101	0.163	0.200	0.117	0.073	0.159
T5-11B	0.132	0.150	0.101	0.155	0.191	0.112	0.073	0.155
DNABERT-6	0.407	0.439	0.362	0.411	0.458	0.342	0.281	0.405

Notably, DNABERT-6, the domain-specific model pre-trained on genomic sequences, exhibits the poorest performance across all datasets (Pearson's $R = 0.227$ on all editors, $R = 0.175$ on ABE combined), with some individual editor datasets even showing negative correlations. While DNABERT-6 possesses domain-specific knowledge about genomic sequence patterns that could theoretically benefit base editing prediction, its narrow pre-training scope on DNA k-mers limits its ability to effectively represent our multimodal input format. The key-value pair structure introduces tokens beyond genomic sequences which fall outside DNABERT-6's pre-training distribution. Therefore, the model is not able to efficiently represent the feature set in its embeddings, ultimately offsetting any advantage from domain-specific genomic knowledge. A potential approach to improve DNABERT's performance would be to generate separate embeddings for DNA sequences using DNABERT and for the remaining multimodal features using a general-purpose LLM, then concatenate these representations. However, systematically evaluating such DNABERT-LLM hybrid architectures is beyond the scope of this study.

It becomes apparent that a dense decoder-only architecture provides the best approach to generate embeddings for base editing efficiency prediction. Several factors contribute to this performance. First, decoder-only models are trained with autoregressive next-token prediction on diverse text and code corpora, enabling them to capture complex sequential dependencies and structured patterns that are crucial for genomic sequence analysis. Second, the unidirectional attention mechanism in decoder-only models may be particularly well-suited for capturing positional dependencies in DNA sequences, where the order and context of nucleotides significantly impact base editing outcomes. Third, the higher-dimensional embeddings (4096-dim for Qwen3 vs 1024-dim for T5/ModernBERT) certainly allow to capture richer representations to encode features. However, the impact of the embedding size does not seem to be substantial, as we will see in the next section when we examine the effect of the transformer size on the performance.

A.1.2 Transformer Size. We evaluate the impact of model scale on the results by comparing three Qwen3 variants with different parameter counts while maintaining an otherwise identical architecture and training procedure. This analysis isolates the effect of model capacity on embedding quality and downstream regression task performance.

Model scale affects both computational requirements and representational capacity. Larger models typically capture more nuanced patterns through an increased parameter count and higher-dimensional embeddings, but may also introduce overfitting risks and computational overhead. We systematically compare three model sizes:

- **32B:** 32 billion parameters with 5120-dimensional embeddings, representing the largest model scale we evaluate
- **14B:** 14 billion parameters with 5120-dimensional embeddings
- **8B:** 8 billion parameters with 4096-dimensional embeddings

Table 4 reports the performance metrics for the three different transformer sizes. A clear pattern emerges: the 8B model achieves the best overall performance on the most diverse datasets, reaching $R = 0.705$ on the all editors dataset and $R = 0.717$ on ABE combined. The 32B model performs best on the CBE combined dataset with $R = 0.724$. On individual editor datasets, the 8B model consistently outperforms larger variants on several datasets, achieving $R = 0.576$ on ABE20m versus $R = 0.541$ for 32B, and $R = 0.596$ on ABE8e versus $R = 0.570$ for 32B.

Overall, the performance differences across all three model sizes remain remarkably close throughout the experiments, with variations typically within 2-4% on most datasets. This experiment illustrates that higher-dimensional embeddings (5120-dim for 14B/32B vs 4096-dim for 8B) do not necessarily provide substantial benefits for base editing prediction. The 8B model provides a good balance between generalization capability and overfitting risk, maintaining consistent strong performance across diverse datasets while requiring significantly lower computational resources.

A.1.3 Pooling Strategies. In the last section of the ablation study, we evaluate different pooling strategies for dimensionality reduction. The pooling strategy determines how token-level embeddings

Table 4: Transformer size comparison using Qwen3 architecture

Model Size	All editors	ABE combined	CBE combined	ABE20m	ABE8e	BE4-1	BE4-2	FNLS
Pearson's R								
8B	**0.705**	**0.717**	0.662	**0.576**	0.596	0.474	0.442	**0.571**
14B	0.661	0.670	0.672	0.556	**0.619**	**0.523**	**0.518**	0.564
32B	0.677	0.674	**0.724**	0.541	0.570	0.507	0.484	0.552
Spearman's ρ								
8B	**0.775**	**0.797**	0.689	**0.653**	0.655	0.524	0.500	0.585
14B	0.702	0.714	0.700	0.645	**0.685**	**0.579**	**0.568**	**0.599**
32B	0.707	0.719	**0.717**	0.632	0.641	0.559	0.527	0.593
Coefficient of Determination (R^2)								
8B	**0.496**	**0.514**	0.434	**0.331**	0.350	0.206	0.161	**0.324**
14B	0.435	0.449	0.451	0.292	**0.380**	**0.261**	**0.245**	0.316
32B	0.457	0.453	**0.524**	0.271	0.313	0.228	0.176	0.298
Root Mean Squared Error (RMSE)								
8B	**0.119**	**0.132**	0.100	**0.140**	0.170	0.104	0.070	**0.135**
14B	0.128	0.145	0.099	0.144	**0.166**	**0.100**	**0.067**	0.136
32B	0.125	0.145	**0.092**	0.146	0.175	0.102	0.070	0.138

Table 5: Pooling strategy comparison across all base editor datasets

Pooling Strategy	All editors	ABE combined	CBE combined	ABE20m	ABE8e	BE4-1	BE4-2	FNLS
Pearson's R								
Attention	**0.705**	**0.717**	0.662	**0.576**	**0.596**	**0.474**	0.442	**0.571**
Max	0.597	0.632	0.566	0.440	0.462	0.336	0.316	0.411
Mean	0.659	0.665	**0.689**	0.516	0.550	0.397	**0.488**	0.494
Median	0.611	0.629	0.660	0.414	0.403	0.355	0.362	0.368
Spearman's ρ								
Attention	**0.775**	**0.797**	**0.689**	**0.653**	**0.655**	**0.524**	0.500	**0.585**
Max	0.628	0.675	0.574	0.519	0.518	0.383	0.373	0.422
Mean	0.677	0.701	0.680	0.598	0.606	0.441	**0.519**	0.512
Median	0.626	0.651	0.628	0.487	0.483	0.408	0.407	0.423
Coefficient of Determination (R^2)								
Attention	**0.496**	**0.514**	0.434	**0.331**	**0.350**	**0.206**	0.161	**0.324**
Max	0.344	0.385	0.303	0.127	0.156	0.008	-0.037	0.125
Mean	0.434	0.439	**0.463**	0.247	0.298	0.133	**0.208**	0.241
Median	0.365	0.394	0.433	0.128	0.102	0.064	0.064	0.057
Root Mean Squared Error (RMSE)								
Attention	**0.119**	**0.132**	0.100	**0.140**	**0.170**	**0.104**	0.070	**0.135**
Max	0.138	0.153	0.111	0.160	0.194	0.116	0.078	0.154
Mean	0.128	0.146	**0.098**	0.148	0.177	0.109	**0.068**	0.144
Median	0.136	0.152	0.101	0.160	0.200	0.113	0.074	0.160

from the transformer are combined into a single sequence representation for the regression head.

Different pooling strategies capture distinct aspects of sequence information: statistical pooling methods (mean, median, max) provide simple aggregation approaches, while attention-based pooling enables learned, context-aware weighting of token representations. We systematically compare four representative strategies:

- **Attention Pooling:** Self-attention–derived pooling over all tokens, allowing the model to focus on sequence regions most relevant for base editing efficiency (see Section 2).
- **Mean Pooling:** Arithmetic average of all token embeddings, providing equal weight to all sequence positions, defined as:

$$v^{\text{mean}} = \frac{\sum_{t=1}^{T} m_t\, h_t}{\max(\sum_{t=1}^{T} m_t,\ \varepsilon)}. \tag{6}$$

- **Median Pooling:** Median value across token embeddings, robust to outlier tokens but may lose important signal. Mathematically denoted as:

$$v^{\text{median}} = \underset{t=1,\ldots,T}{\text{median}}(m_t\, h_t + (1 - m_t)\, 10^{-9}\mathbf{1}). \tag{7}$$

- **Max Pooling:** Element-wise maximum across token embeddings, capturing the most activated features per dimension and is defined as:

$$v^{\max} = \max_{t=1,\ldots,T}(m_t\, h_t + (1 - m_t)(-10^{9})\, \mathbf{1}), \tag{8}$$

where $h_t \in \mathbb{R}^d$ are contextualized token representations, $m_t \in \{0, 1\}$ is the padding mask, $\mathbf{1} \in \mathbb{R}^d$ is the vector of ones, and $\varepsilon = 10^{-9}$.

The pooling strategy analysis shows distinct performance patterns across dataset types and sizes. Attention pooling, which derives weights from the model's internal attention, performs best overall across most datasets. On the most diverse all editors dataset, attention pooling achieves the highest performance with $R = 0.705$

and Spearman's $\rho = 0.775$. The attention strategy also dominates on ABE combined ($R = 0.717$, $\rho = 0.797$) and shows strong performance on individual editor datasets like ABE20m with $R = 0.576$ (+11.6%) versus mean pooling's $R = 0.516$.

For some heterogeneous datasets, mean pooling provides competitive results. On the CBE combined dataset, mean pooling achieves $R = 0.689$ (+4.1%) compared to attention pooling's $R = 0.662$, and on BE4-2, mean pooling reaches $R = 0.488$ (+10.4%) versus attention's $R = 0.442$. However, attention pooling maintains superior Spearman correlation across most datasets, suggesting better rank prediction even when Pearson correlation is slightly lower.

The worst-performing strategies are max pooling and median pooling with max pooling particularly underperforming on smaller datasets. On BE4-2, max pooling achieves only $R = 0.316$ compared to attention's $R = 0.442$, representing a 28.5% performance drop. This pattern makes intuitive sense since max pooling typically captures high outlier values, which may not represent meaningful patterns in genomic sequences but rather noise or artifact features that hurt generalization on limited training data.

A.2 Data and Code Availability

The datasets used in this study are available through the Be-dataHIVE base editing database [12]. The LLM-EmBEditor implementation is available at https://github.com/Lucas749/LLM-EmBEditor.

A.3 Training Details

Our model was trained on GPU clusters provided by the Advanced Research Computing (ARC) Service at the University of Oxford. The embeddings were generated using the Qwen3-8B model, a dense decoder-only transformer architecture. The complete model and training configurations are summarized in Table 6.

Table 6: LLM-EmBEditor model and training configuration

Large Language Model	
Parameter	**Value**
Model Name	Qwen/Qwen3-8B
Architecture	Dense Decoder-Only Transformer
Model Layers	36
Pooling Approach	Attention weighted
Max Sequence Length	600
Batch Size	1
Device	cuda
Embedding Dimension	4096
Embedding Source	Last hidden state (normalized)
Regression Head	
Parameter	**Value**
Architecture	$4096 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow 1$
Hidden Activation	ReLU
Output Activation	Sigmoid
Regularization	Batch Normalization + Dropout (0.3)
Optimizer	AdamW
Learning Rate	1e-3
Weight Decay	0.01
Batch Size	32
Epochs	20
Loss Function	Mean Squared Error (MSE)