From One to Zero: RAG-IM Adapts Language Models for Interpretable Zero-Shot Clinical Predictions

Sazan Mahbub[*1], **Caleb Ellington**[1], **Sina Alinejad**[5], **Kevin Wen**[4], **Yingtao Luo**[1], **Ben Lengerich**[†3], **Eric P. Xing**[†1,2,6]

[1]Carnegie Mellon University
[2]Mohamed bin Zayed University of Artificial Intelligence
[3]University of Wisconsin - Madison
[4]Massachusetts Institute of Technology
[5]Iran University of Science and Technology
[6]GenBio AI

Abstract

Clinical machine learning models must adapt to new settings such as different hospitals, clinicians, or patient populations. These differing environments present related but subtly distinct tasks, where diseases and medical interventions share common foundations but vary in meaningful ways. In contrast to one-size-fits-all invariant feature learning, we believe representing meaningful differences between domains and adapting to these differences will improve accuracy, utility, and interpretability of machine learning in health. Here, we introduce Retrieval-Augmented Generation of Interpretable Models (RAG-IM), a highly performant method for adapting statistical models to new domains based on their descriptions. By leveraging the strengths of Retrieval-Augmented Generation (RAG), our framework retrieves relevant models from related tasks and combines them with contextual insights from pre-trained language models. RAG-IM generates task-specific, interpretable models that perform reliably, even in few-shot and zero-shot scenarios where data are limited or completely unavailable. Through experiments on 7487 related tasks, we find that RAG-IM is a promising general-purpose platform to enable model-based analysis to data-limited and heterogeneous regimes by connecting statistical analysis with natural language.

1 Introduction

Recent advances in clinical machine learning have aimed to balance interpretability and performance, especially in low-data scenarios. Meta-models [1, 2] are designed to dynamically generate models that are tailored to specific contexts or tasks. These meta-models use pre-training on diverse datasets to fine-tune context-specific models, enabling scalability in low-resource settings.

In contrast to such approaches that train the meta-model from scratch, several recent efforts have leveraged pre-trained language models as a form of prior knowledge [3]. For example, language models have been used in reinforcement learning to explore nuanced actions efficiently [4, 5, 6], in feature selection [7], and in causal graph discovery [8, 9]. Health-LLM [10] integrates health reports and medical knowledge into large models using retrieval-augmented generation (RAG), improving feature extraction and prediction accuracy. However, these language model based approaches are not designed for interpretability, limiting their use in healthcare.

[*]smahbub@cs.cmu.edu
[†]Corresponding authors: epxing@cs.cmu.edu, lengerich@wisc.edu

NeurIPS 2024 Workshop on Adaptive Foundation Models.

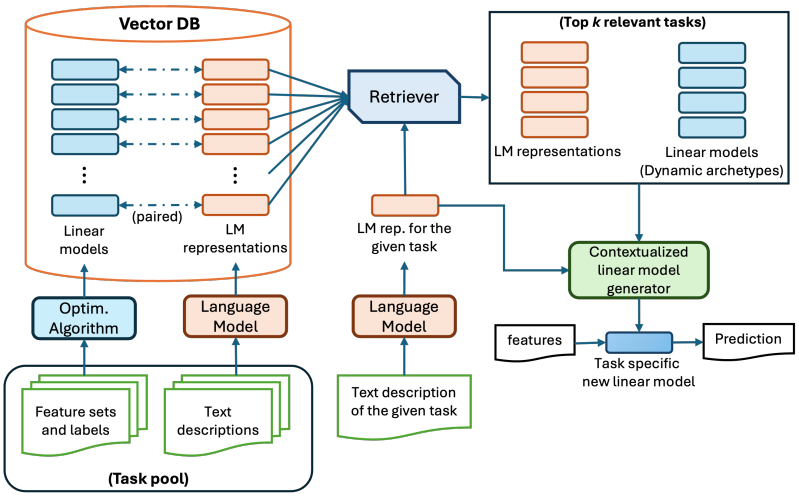

Figure 1: RAG-IM pipeline for adapting a family of regression tasks to few-shot or zero shot (see details in Section 2).

Our RAG-IM framework builds on these approaches by not only leveraging language model prior information but also ensuring that the models it generates are interpretable. By retrieving and adapting models from related tasks and environments, RAG-IM can generate simple models focused to highly-specific tasks and environments, simultaneously providing interpretability and accuracy. These characteristics make RAG-IM particularly well-suited for healthcare applications that demand both adaptability and transparency in low-data environments.

2 RAG-IM: Retrieval-Augmented Generation of Interpretable Models

In this section, we discuss our proposed framework RAG-IM. RAG-IM has three major components: (1) vector database creation, (2) archetype retrieval, and (3) linear model generation. The overall framework is shown in Figure 1.

Vector Database Creation. We assume that at training time, we have access to tasks drawn from a pool $\mathbb{T} = \{(X_i, Y_i, c_i) \ : \ i \in [1, \mathbf{n}]\}$ such that $\forall\ \mathbb{T}_i \in \mathbb{T}$ we have m_i samples with known feature sets $X_i \in \mathbb{R}^{m_i \times f}$ (f := # of total possible features) and labels $Y_i \in \mathbb{R}^{m_i \times 1}$. We also have access to the text description of each task c_i, which we refer to *context*, consistent with prior work on contextualized modeling [2, 11].

For each task $\mathbb{T}_i$, we then learn a linear model $\hat{\theta}_i \in \mathbb{R}^f$ by optimizing the loss function $\mathcal{J}(.)$,

$$\hat{\theta}_i = \arg\min_{\theta} \ \mathcal{J}(X_i, Y_i, \theta). \tag{1}$$

Next, using an existing language model LM(.), we compute the sentence embedding $\mathcal{E}_i = \text{LM}(c_i) \in \mathbb{R}^e$ (e = embedding dimension of the language model). We store $(\hat{\theta}_i, \mathcal{E}_i)\ \forall i \in [1, \mathbf{n}]$ in a database to be used in the subsequent modules of our framework. This database, usually referred to as *vector database (DB)* [12], can be defined as a set $\mathbb{V} = \{(\hat{\theta}_i, \mathcal{E}_i) : i \in [1, \mathbf{n}]\}$. Thus, as n, the number of training tasks, grows, this vector database also grows and increases coverage of diverse settings.

Archetype Retrieval. The aim of our retrieval pipeline is to retrieve the top-k most relevant vector from $\mathbb{V}$, given a query q. In our case, the query is the sentence embedding of the new task's description through the same language model we used during the vector-DB creation. Specifically, for a given task description c_t, the query is a vector $q_t = \text{LM}(c_t) \in \mathbb{R}^e$.

In order to retrieve the most relevant tasks, we first define a metric $\mathcal{R}(q_t, \mathbb{V}_i)$ that compute how much the i-th task is relevant to the given task. The function $\mathcal{R}(q_t, \mathbb{V}_i)$ can be designed in various ways [13, 14]. Here, we follow the practice of [15] to use the cosine similarity between task embeddings:

$$\mathcal{R}(q_t, \mathbb{V}_i) = \text{cos\_sim}(q_t, \mathcal{E}_i) = \frac{q_t.\mathcal{E}_i}{\|q_t\|\|\mathcal{E}_i\|}, \tag{2}$$

and note that this could be extended to learnable distance metrics. We note that here only the language embedding $\mathcal{E}_i$ component in $\mathbb{V}_i = (\hat{\theta}_i, \mathcal{E}_i)$ is being used, however it is also possible to incorporate models $\hat{\theta}_i$ which we leave as a future work.

This distance metric induces a permutation function $\pi(i)$ that sorts the entries in $\mathbb{V}$ such that $\mathcal{R}(q_t, \mathbb{V}_{\pi(1)}) \geq \mathcal{R}(q_t, \mathbb{V}_{\pi(2)}) \geq \cdots \geq \mathcal{R}(q_t, \mathbb{V}_{\pi(\mathbf{n})})$. The retrieval pipeline outputs $\mathbb{V}^r \subseteq \mathbb{V}$ with the top k most relevant entries from $\mathbb{V}$, where $\mathbb{V}^r = \{\mathbb{V}_{\pi(j)} : j \in [1, k]\}$. From the retrieved set $\mathbb{V}^r$, we want to generate a task-specific model $\Phi(c_t)$ that is a function of the task description c_t. To do this, we consider the retrieved models $\hat{\theta}_j \in \mathbb{V}^r_j$ as *archetypes* [2, 16] that will be combined to form a single linear model.

Interpretable Linear Model Generation. According to previous studies by [2] and [16], archetypes can be combined according to a simple linear combination:

$$\Phi(c_t) = \sum_{j=1}^{k} \alpha(c_t)_j \hat{\theta}_j, \tag{3}$$

where $\alpha(c_t)_j$ is a predicted scalar weight of $\hat{\theta}_j$ for the linear combination. However, here the assumption behind $\hat{\theta}_j$ is that they will always be the same set of archetypes and will have the same order. In our approach, we relax this assumption and redesign $\Phi(\cdot)$ and $\alpha(\cdot)_j$ as functions of both c_t and $\mathbb{V}^r$, To address this, we redesign $\Phi(\cdot)$ and $\alpha(\cdot)_j$ as functions of both c_t and $\mathbb{V}^r$,

$$\Phi(c_t, \mathbb{V}^r) = \sum_{j=1}^{k} \alpha(c_t, \mathbb{V}^r)_j \hat{\theta}_j. \tag{4}$$

This way $\alpha(c_t, \mathbb{V}^r)_j$ can assign a scalar weight for each model $\hat{\theta}_j$ based on all the retrieved models and their corresponding task descriptions (encoded as $\mathcal{E}_j$).

We note that Equation 4 is a form of cross-attention mechanism on the retrieved models. This inspires us extend this further to multihead cross-attention (MHCA), which is a relaxation of Equation 4 by allowing the exploration of multiple potential attention-pooling trajectories simultaneously [17, 18]. The MHCA follows the same formulation as in [17]. Here we start by generating three matrices $V^0 \in \mathbb{R}^{k \times d}$, $K^0 \in \mathbb{R}^{k \times d}$, and $Q^0 \in \mathbb{R}^{1 \times d}$ as follows,

$$V^0 = [\hat{\theta}_{\pi(1)}, \ldots, \hat{\theta}_{\pi(k)}]\, W_V \tag{5}$$

$$K^0 = [[\hat{\theta}_{\pi(1)} \| \mathcal{E}_{\pi(1)}], \ldots, [\hat{\theta}_{\pi(k)} \| \mathcal{E}_{\pi(k)}]]\, W_K \tag{6}$$

$$Q^0 = q_t\, W_Q, \tag{7}$$

where "||" represents concatenation operation, and $W_V \in \mathbb{R}^{f \times d}$, $W_K \in \mathbb{R}^{(f+e) \times d}$, and $W_Q \in \mathbb{R}^{e \times d}$ are learnable parameters for linear projection. Note that, while K^0 is a function of the model parameters $\hat{\theta}_j$ and the sentence embeddings $\mathcal{E}_j$, V^0 is a function of just the linear models. This design will allow the model to leverage latent information encoded in both $\hat{\theta}_j$ and $\mathcal{E}_j$ when computing the importance of the model $\hat{\theta}_j$ in generating the final model. Using these components we can get a new formulation for $\Phi(c_t, \mathbb{V}^r)$,

$$\Phi(c_t, \mathbb{V}^r) = \text{MHCA}^l(V^0, K^0, Q^0)\, W_o \tag{8}$$

where $\text{MHCA}^l(.)$ is the multihead cross-attention module, and $W_o \in \mathbb{R}^{d \times f}$ is a learnable matrix that projects back to the feature space.

We can further extend the framework to apply multiple layers of MCHA. We can achieve this by iteratively applying Equations 9 and 10.

$$V^l = \text{MHCA}^l(V^0, K^0, Q^{l-1}),\ \forall l \in [1, L] \tag{9}$$

$$Q^l = Q^l + \text{MLP}_b^l(V^l),\ \forall l \in [1, L-1] \tag{10}$$

Equations 9 and 10 show the operations performs by the l-th layer, where V^l and Q^l are the updated values, and $\text{MLP}_b(.)$ is the bottleneck multilayer perceptron [19] module of that layer. Note that in layers $l \in [1, L-1]$ we iteratively update the query vector to enrich it with more useful information about other relevant tasks encoded and stored in $\mathbb{V}^r$. Finally, our generated model $\Phi(c_t, \mathbb{V}^r)$ is,

$$\Phi(c_t, \mathbb{V}^r) = V^L\, W_o. \tag{11}$$

We depict the architecture of this contextualized linear model generator in Figure 3 (Appendix A).

On top of this, we also provide a conditional skip connection from the task-specific linear model (of the given task) to $\Phi(c_t, \mathbb{V}^r)$. This connection is conditioned on the availability of a pre-trained linear model for the task, as well as the abundance of its training data, which tells us how well the model was trained. The rationale behind this is, if we have a well-trained linear model to start with, we can directly leverage that or augment that with our generated model., i.e., the skip connection. Otherwise, we should directly use $\Phi(c_t, \mathbb{V}^r)$ only. In our experiments, we set the condition of data abundance as tasks with at least ≥ 50 samples, which is met by only about 3.4% of all tasks. For the rest of the tasks, this condition is not met and the skip connection switches off. Detailed experimental results are discussed in Section 3.

3 Results and Discussion

3.1 Dataset

For our experiments, we used MIMIC IV dataset [20]. We provide a discussion on this dataset and and our preprocessing approach in Appendix B.

3.2 Experiments

We have in total 7487 tasks in the dataset with at least one positive sample. Each task represents a procedure recommended by doctors (a binary classification task), and each positive sample corresponds to a patient during one hospital admission. Examples are shown in Appendix B. For training and testing, we randomly select an equal number of negative samples from other tasks, resulting in twice the number of samples per task. The dataset is then randomly split so that both the training and test sets have equal numbers of positive and negative samples. Tasks with only one positive sample are used for testing only, as they cannot be split into train-test sets. In our experiment, we choose task-specific Logistic Regression models as our baselines that are trained on each of the 4412 tasks with at least two samples in the training split.

The distribution of samples in each task is shown in Figure 2 (the exponentially decaying graph shown in orange). The tasks are sorted by the decreasing order of the number of samples in them. For example, the first task (with ID = 0) has 8504 samples, but then we see a very sharp decay to only 1 sample per-task. For our experiments, we split the tasks in five different groups based on the abundance of samples as well as the performance of the baseline Logistic Regression models (the baseline scores in the table within Figure 2).

The first group has tasks with at least 50 samples each, and as expected, baseline performance here is strong. This group represents the *high-abundance data regime*. The second group consists of tasks with 10 to 50 samples, where performance remains good, though much lower than the high-abundance regime. This group represents the *moderate-abundance data regime*. Next are tasks with 5 to 10 samples, representing the *low-abundance data regime*. Although it is difficult to learn effective task-specific models with such limited data, the models still perform well above random prediction (50%). The group with 2 to 4 samples per-task is the *few-shot regime*, which covers almost 25% of all tasks in the dataset. The optimization algorithms are exposed to very few training samples, and thus baseline performance hovers around random chance, close to 50%. The final group includes 3075 tasks, with only one sample each and it used only for testing. With no training data, zero-shot prediction relies on meta-learning from more data-abundant regimes for knowledge transfer. We call this the *zero-shot regime*. Task-specific models fail here, providing no predictions (marked as "N/A" in Figure 2). It is important to note that about 41% of the tasks fall under the zero-shot regime, which is the largest group of this dataset. This shows that doing well on few-shot and zero-shot tasks is crucial for a model to perform well in the real-world.

In Figure 2, we also show the summarized results of our proposed framework RAG-IM in these five regimes. For the We can see, except for the weighed average accuracy in the high-abudance regime,

Range of sample counts:		[50, 8504]	[10, 50)	[5, 10)	[2, 5)	{1}
Weighed avg. accuracy	Baseline	**0.796**	0.681	0.602	0.554	N/A
	RAG-IM	0.777	**0.698**	**0.705**	**0.699**	**0.695**
Avg. accuracy	Baseline	0.762	0.672	0.602	0.550	N/A
	RAG-IM	**0.765**	**0.700**	**0.705**	**0.700**	**0.695**

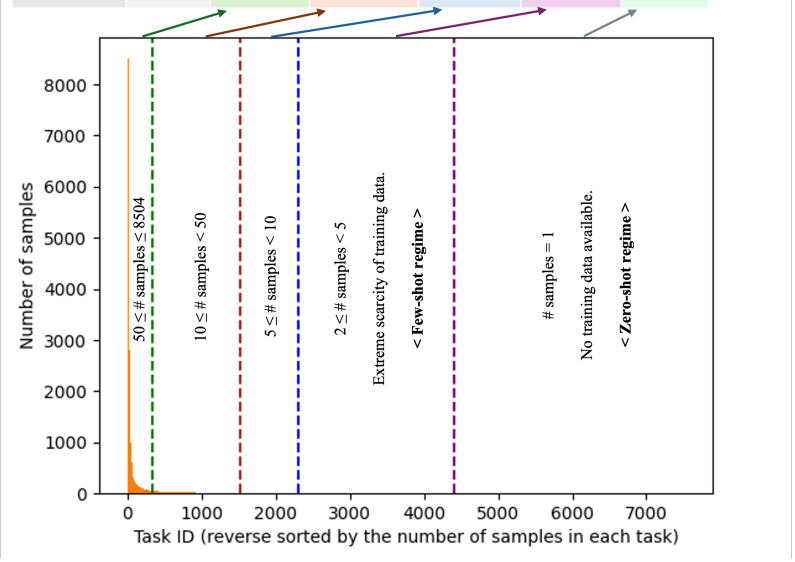

Figure 2: Overview of dataset statistics and experimental results. Tasks are split into five groups based on sample abundance. **Below:** The sample count follows a long tailed distribution (shown in orange barplot). **Top:** A table with experimental results of RAG-IM and baseline Logistic Regression models on each group (see Section 3).

our model significantly out-performs the baseline in all the over case. While the performance of the baseline model quickly decays with fewer training data, RAG-IM keeps performing consistently well (around 70%), and the performance gap keeps getting more significant for more challenging groups. Even in the few-shot and zero-shot regimes, RAG-IM's performance is not affected due to it's capability to effectively transfer knowledge from the most relevant Logistic Regression models in highly-performant regimes, using the learnt prior in the language model. We show more results in Table 1 in Appendix D. In this table we also show the precision, recall, and F1-scores. Here we can see that the performance the baseline and the RAG-IM follows the similar trend as the accuracies for other metrics as well. For these experiments, we use DistilBERT [21] which an efficient distilled version of BERT [22]. In the future we aim to explore other language models [23, 24] that have been shown to achieve state-of-the-art in different retrieval tasks previously [25, 26].

4 Conclusion

In this work, we introduced Retrieval-Augmented Generation of Interpretable Models (RAG-IM), a framework designed to enhance adaptability and interpretability in clinical machine learning, addressing the need for models to adapt across diverse clinical environments. By retrieving relevant models from related tasks and integrating contextual insights from pre-trained language models, RAG-IM offers task-specific, interpretable predictions, even in few-shot and zero-shot scenarios. Our experiments on 7,487 clinical prediction tasks demonstrated that RAG-IM consistently outperforms baseline models, particularly in data-limited and heterogeneous settings. This approach bridges statistical analysis with natural language, enabling robust model adaptation across varied clinical domains. Future work will explore the integration of larger language models and the optimization of the retrieval process with learnable distance metrics.

Acknowledgements

This research has been graciously funded by the National Science Foundation (NSF) awards BCS2040381 and IIS2123952 (to S.M. and E.X.), the Semiconductor Research Corporation (SRC) AIHW award 2024AH3210 (to S.M. and E.X.), National Institutes of Health (NIH) award R01GM140467 (to C.E. and E.X.), Carnegie Mellon University CMLH Translational Fellowship in Digital Health (to Y.L.). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of National Science Foundation, Semiconductor Research Corporation, National Institutes of Health, and Carnegie Mellon University CMLH.

References

[1] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[2] Benjamin Lengerich, Caleb N Ellington, Andrea Rubbi, Manolis Kellis, and Eric P Xing. Contextualized machine learning. *arXiv preprint arXiv:2310.11340*, 2023.

[3] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature medicine*, 29(8):1930–1940, 2023.

[4] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models, 2023.

[5] Thommen George Karimpanal, Laknath Buddhika Semage, Santu Rana, Hung Le, Truyen Tran, Sunil Gupta, and Svetha Venkatesh. Lagr-seq: Language-guided reinforcement learning with sample-efficient querying, 2023.

[6] Wanpeng Zhang and Zongqing Lu. Adarefiner: Refining decisions of language models with adaptive feedback, 2024.

[7] Dyah Adila, Changho Shin, Linrong Cai, and Frederic Sala. Zero-shot robustification of zero-shot models, 2024.

[8] Stephanie Long, Alexandre Piché, Valentina Zantedeschi, Tibor Schuster, and Alexandre Drouin. Causal discovery with language models as imperfect experts, 2023.

[9] Chenxi Liu, Yongqiang Chen, Tongliang Liu, Mingming Gong, James Cheng, Bo Han, and Kun Zhang. Discovery of the hidden world with large language models, 2024.

[10] Mingyu Jin, Qinkai Yu, Dong Shu, Chong Zhang, Lizhou Fan, Wenyue Hua, Suiyuan Zhu, Yanda Meng, Zhenting Wang, Mengnan Du, et al. Health-llm: Personalized retrieval-augmented disease prediction system. *arXiv preprint arXiv:2402.00746*, 2024.

[11] Jannik Deuschel, Caleb Ellington, Yingtao Luo, Ben Lengerich, Pascal Friederich, and Eric P Xing. Contextualized policy recovery: Modeling and interpreting medical decisions with adaptive imitation learning. In *Forty-first International Conference on Machine Learning*, 2024.

[12] Yikun Han, Chunjiang Liu, and Pengfei Wang. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*, 2023.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[14] Boyu Zhang, Hongyang Yang, Tianyu Zhou, Muhammad Ali Babar, and Xiao-Yang Liu. Enhancing financial sentiment analysis via retrieval augmented large language models. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 349–356, 2023.

[15] Liang Zhang, Katherine Jijo, Spurthi Setty, Eden Chung, Fatima Javid, Natan Vidra, and Tommy Clifford. Enhancing large language model performance to answer questions and extract information more accurately. *arXiv preprint arXiv:2402.01722*, 2024.

[16] Caleb N Ellington, Benjamin J Lengerich, Thomas BK Watkins, Jiekun Yang, Hanxi Xiao, Manolis Kellis, and Eric P Xing. Contextualized networks reveal heterogeneous transcriptomic regulation in tumors at sample-specific resolution. *bioRxiv*, pages 2023–12, 2023.

[17] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.

[20] Alistair EW Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, et al. Mimic-iv, a freely accessible electronic health record dataset. *Scientific data*, 10(1):1, 2023.

[21] V Sanh. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[22] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[23] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[24] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[25] Justin Levandoski, David Lomet, and Sudipta Sengupta. Llama: A cache/storage subsystem for modern hardware. In *Proceedings of the International Conference on Very Large Databases, VLDB 2013*, 2013.

[26] Michael Wornow, Yizhe Xu, Rahul Thapa, Birju Patel, Ethan Steinberg, Scott Fleming, Michael A Pfeffer, Jason Fries, and Nigam H Shah. The shaky foundations of clinical foundation models: A survey of large language models and foundation models for emrs. *arXiv preprint arXiv:2303.12961*, 2023.

[27] Sana Tonekaboni, Shalmali Joshi, Melissa D McCradden, and Anna Goldenberg. What clinicians want: contextualizing explainable machine learning for clinical end use. In *Machine learning for healthcare conference*, pages 359–380. PMLR, 2019.

[28] Ben Lengerich, Bryon Aragam, and Eric P Xing. Learning sample-specific models with low-rank personalized regression. *Advances in Neural Information Processing Systems*, 32, 2019.

[29] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

[30] Sebastian Bordt, Ben Lengerich, Harsha Nori, and Rich Caruana. Data science with llms and interpretable models, 2024.

[31] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[32] Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality, 2024.

[33] Kristy Choi, Chris Cundy, Sanjari Srivastava, and Stefano Ermon. Lmpriors: Pre-trained language models as task-specific priors, 2022.

A RAG-IM Framework

In Figures 3, we show the architecture of our contextualized model generator.

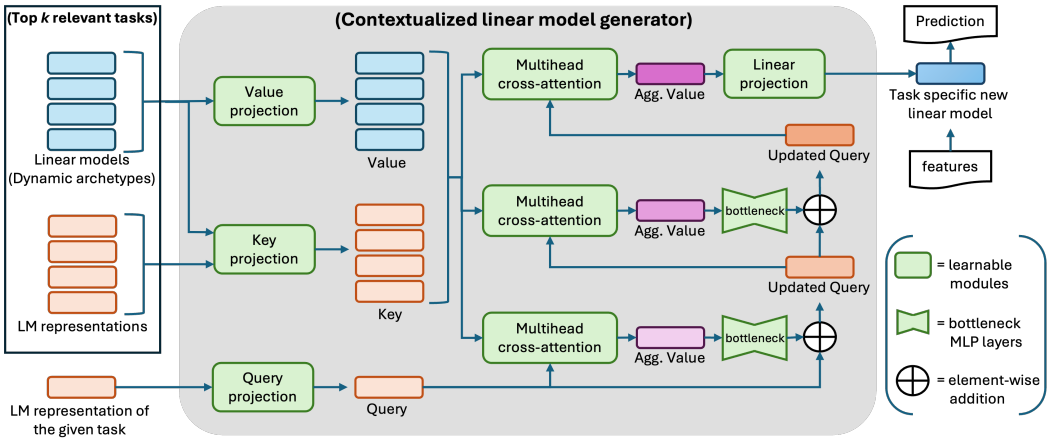

Figure 3: Architecture of our contextualized linear model generator. In this figure, we are showing three layers of multihead cross-attention for example.

B Dataset

The pre-processed dataset mainly consists of two tables. The first table is laboratory events which are grouped by (hadm_id, subject_id) pair. The second table consists of procedures which is again grouped by (hadm_id, subject_id) pair. So with that, we have laboratory measurements which are our input features and procedures applied to that patient in that particular hospital stay, which provides our outputs or labels for the task. There comes a definition for each procedure or task that plays the role of context in the proposed model. Note that we should drop the rows of the second table that don't have corresponding rows in the first table. For the measurements, we simply normalize the values based on the upper range and lower range of that laboratory item using the Equation 12. For example, blood pressure's normal value is typically between 80 and 120 (mmHg), so the lower range and upper range would be 80 and 120 respectively. Patients' blood pressure may be higher than the upper range or lower than the lower range. There were a total of 7487 procedures in the dataset, 4412 of them having at least two samples. We used a split ratio of 50% for the train and test set for each of these 4412 procedures. For each procedure or task, there is an equal number of positive and negative samples too. We use the rest of the 3075 tasks for few-shot performance evaluation.

$$normalized\_feature = \frac{raw\_value - lower\_range}{upper\_range - lower\_range} \tag{12}$$

In our experiments, we leverage the definitions of the procedures as the task descriptions. Some examples include – "Respiratory Ventilation, 24-96 Consecutive Hours"; "Inspection of Upper Intestinal Tract, Via Natural or Artificial Opening Endoscopic"; "Fluoroscopy of Multiple Coronary Arteries using Other Contrast"; "Introduction of Other Antineoplastic into Central Vein, Percutaneous Approach", and etc.

C Background and More Prior Work

Contextualized Machine Learning (ML) is a paradigm for learning heterogeneous and context-dependent effects. Some works that have contributed to this include but are not limited to [2, 11, 27, 28].

RAG architecture, which stands for "Retrieval-Augmented Generation," is an AI framework that enhances the capabilities of a pretrained language model by incorporating an information retrieval system, allowing it to access and leverage external data sources to generate more accurate and relevant responses based on the specific context of a query or prompt [13, 29].

Meta-learning is a machine learning technique that enables AI models to learn how to learn, allowing them to adapt to new tasks on their own. It's also known as "learning to learn". There are some works on that such as [1].

Combining pre-trained language models with interpretable models enables the use of powerful, context-rich knowledge from language models (LM) while maintaining transparency in decision-making through simpler, interpretable models. This fusion ensures high performance without sacrificing the ability to explain the model's reasoning, critical in fields like healthcare. Examples of this are [30]. Some works have focused on using LM priors for downstream tasks, like [31, 4, 5] for Reinforcement Learning, [7] for Feature Selection, and [8, 32] for Causal Graph Discovery. [33] utilize these priors for all the three above tasks.

D Results

Our detailed experimental results are reported in Table 1. Here we report four different metrics (precision, recall, F1-score, and accuracy) for comparison.

Table 1: Performance Comparison of RAG-IM and Baseline. The tasks are grouped into five different groups based on the number of samples in them. The first columns shows the range of the number of samples in the tasks within each group.

Range of sample counts	Method	Metric	Weighted Average over tasks	Average over tasks
[50, 8504]	RAG-IM	Precision	0.7876	0.7732
		Recall	0.7585	0.7539
		F1 Score	0.7701	0.7595
		Accuracy	0.7775	0.7652
	Baseline	Precision	0.7999	0.7639
		Recall	0.7917	0.7641
		F1 Score	0.7947	0.7610
		Accuracy	0.7963	0.7615
[10,50)	RAG-IM	Precision	0.7079	0.7133
		Recall	0.6747	0.6761
		F1 Score	0.6766	0.6779
		Accuracy	0.6978	0.7000
	Baseline	Precision	0.6985	0.6953
		Recall	0.6790	0.6689
		F1 Score	0.6713	0.6597
		Accuracy	0.6807	0.6715
[5, 10)	RAG-IM	Precision	0.7193	0.7183
		Recall	0.6881	0.6882
		F1 Score	0.6766	0.6760
		Accuracy	0.7054	0.7047
	Baseline	Precision	0.6001	0.5975
		Recall	0.5755	0.5745
		F1 Score	0.5456	0.5437
		Accuracy	0.6023	0.6016
[2, 5)	RAG-IM	Precision	0.6546	0.6393
		Recall	0.6887	0.6898
		F1 Score	0.6470	0.6421
		Accuracy	0.6986	0.7002
	Baseline	Precision	0.4050	0.3860
		Recall	0.5303	0.5285
		F1 Score	0.4339	0.4240
		Accuracy	0.5544	0.5500
{1}	RAG-IM	Precision	0.5831	0.5831
		Recall	0.6758	0.6758
		F1 Score	0.6140	0.6140
		Accuracy	0.6954	0.6954
	Baseline	Precision	N/A	N/A
		Recall	N/A	N/A
		F1 Score	N/A	N/A
		Accuracy	N/A	N/A