Toward Predicting Temporal Changes in a Patient's Chest X-Ray Images Based on Electronic Health Records: Re-Implementation and Practical Enhancements

Written by Mariam Joan, Shang Lu, Jesus Jimenez Garizao

[1]University of Illinois Urbana-Champaign
601 East John Street, Champaign, IL 61820, USA
mjoan2@illinois.edu, lu94@illinois.edu, jesusaj2@illinois.edu

Abstract

The research from *Towards Predicting Temporal Changes in a Patient's Chest X-ray Images based on Electronic Health Records* (Kyung et al. 2025) aimed to predict chest X-rays from a patient's prior medical imaging and medical events using Latent Diffusion Modeling (LDM) and a Convolutional Neural Network (CNN) called U-Net. This project re-implements the authors' research, reproducing the authors' EHRXDiff framework with additional features to support privacy, cost, and improved accuracy. Specifically, we replace external API embeddings with locally hosted, domain-adapted BERT encoders and explore the addition of RAG to minimize retraining and provide greater contextual information for Intensive Care Unit (ICU) signals when patient data changes rapidly. The evaluation metrics will cover the quality of the medical images generated. We hypothesize these practical enhancements will better capture rapid clinical changes and yield higher model accuracy.

Code — https://drive.google.com/drive/u/0/folders/1npVWhiMnXSFOYYR9jKW42lY_qT2gtzQX

Datasets — https://drive.google.com/drive/u/0/folders/1CmwwrV9O9QUnOQANsJKELTvOiCaY-q4M

Presentation Video — https://drive.google.com/drive/u/0/folders/1JqHOJil4TCWwC5v2khRidrKhhy5b8dgJ

PyHealth Pull Request — https://github.com/sunlabuiuc/PyHealth/pull/647

Introduction

For patients with chest obstructions or legions needing medical care, it is critical to have medical history and imaging to help support diagnosis, with the ability to track the patient's health progress. Research from *Towards Predicting Temporal Changes in Patient Chest X-ray Images based on Electronic Health Records* (Kyung et al. 2025) provides a unique way to predict future chest X-rays from a patient's previous medical imaging and medical events, in an attempt to provide the patient with a future prognosis depending on the severity of the patient's current state, so that a treatment plan might be started, putting the patient on the path to better health, with goal of mitigating the severity of future symptoms.

The unique benefits of the authors' EHRXDiff novel framework is the ability to predict future chest X-rays (CXR) not based simply on imaging from a single point in time, but rather the patient's historical medical imaging in addition to the patient's electronic health records (EHR). The generative diffusion based models currently on the market use only current imaging and do not factor in previous medical imaging with EHR.

The EHRXDiff model architecture consists of two encoders, Variational Autoencoder (VAE) and Contrastive Language-Image Pre-training (CLIP), a latent diffusion model (LDM) and a convolutional neural net (CNN) that processes sequential longitudinal CXR images and the patient's EHR to produce high quality CXR imaging.

The evaluation process will assess the quality of medical imaging, while the authors' research covered additional areas such as, consistency across medical events, and demographics.

Scope of Reproducibility

We successfully reproduced all image and text dataset processing steps, embeddings, model construction, and training, as outlined in the authors' research. With approval and due to timing constraints, we limited our evaluation metrics to only confirmation of image quality. We utilized the authors' open source EHRXDiff (Daeun Kyung 2025) Github repository for the code implementation, modifying specific areas such as the embeddings component and the LDM to support re-implementation, given our compute constraints. In all, we replicated the following components:

- Data processing (MIMIC-IV, MIMIC-CXR-JPG)
- Embeddings model
- VAE and CLIP models (Encoder)
- U-Net model (CNN)
- Diffusion model (LDM)
- Evaluation (Fréchet Inception Distance)

Methodology

Environment

The project's development environment consisted of a Google Drive with 200 GB storage, and Google Colab Pro+ notebooks running Pandas, Numpy, PyTorch, and Facebook

AI Similarity Search (FAISS), among other Python dependencies. We utilized the authors' files from their EHRXDiff repository and setup 11 Google Colab Pro+ notebooks, with separate scripts according to each component function. The model implementation, training and evaluation was processed in Python 3.12.12 and PyTorch 2.9 with the following dependency packages:

- **numpy**: 2.0.2
- **pandas**: 2.1.1
- **torch**: 2.9.0+cu126
- **transformers** >=4.40
- **tokenizers** >=0.15
- **faiss-gpu-cu12**: 1.13.0
- **h5py**

Data

The original authors utilized MIMIC-IV and MIMIC-CXR-JPG datasets for their research and in our implementation, we obtained these datasets through PhysioNet, which enabled integration with Amazon S3 and Google Cloud Platform (GCP). However, utilizing PyHealth (SunLab), an open source python library and toolkit for accessing healthcare data applications, to support dataset retrieval, was the most efficient outcome based on resourcing and time to setup.

The MIMIC-IV dataset includes anonymized EHRs from roughly 50k patients admitted to Beth Israel Deaconess Medical Center (BIDMC) between 2008 and 2019. The MIMIC-CXR-JPG dataset is derived from MIMIC-CXR, and includes over 227k images with Digital Imaging and Communication in Medicine (DICOM) (NEMA) identifiers from BIDMC between 2011 and 2016.

Please see Table 1 for cohort summary statistics from this data. We construct a CXR cohort by linking MIMIC-IV `subject_id` with the MIMIC-CXR-JPG subset from PyHealth. The authors filter to CXR imaging to frontal view that occur only during a period from when the patient has entered the hospital to when they are discharged, with a maximum two day time period (Kyung et al. 2025). EHR data is also filtered to patients over 18 years of age, and includes EHR medical events such as chart, lab, prescriptions, and microbiology.

Data Preprocessing A critical component to the authors' research, enabling their ability to implement a multimodal environment for modeling is the triplification of data. The EHRs and imaging data are extracted and transformed into triples that contain: $(I_{\text{prev}}, \mathbf{S}_{\text{event}}, I_{\text{trg}})$ which refers to the previous image, medical EHR events, and the target image, respectively.

The EHR events are structured data such as charts and medications, filtered to cardiovascular and chest related imaging. The EHR events are referenced as a structured data table from Hur et al. (Kyunghoon Hur 2022) methodology. We utilize this structured table data in the embeddings component and it also plays a role in evaluation metrics.

Table 1: Data summaries.

	MIMIC-IV	MIMIC-IV-CXR-JPG
Number of CXR studies		53,177
Number of CXR images		122,735
Number of patients	10,297	10,885
Number of stays	12,516	13,387
LoS (mean, days)	22.33	
LoS (median, days)	16.97	
In-hospital mortality (%)	12.27	
Gender (% male)	54.94	
Age (mean, years)	66.06	
Age (median, years)	68.00	

Model

Please see Figure 1 for the authors' diagram of the model architecture, which includes up to six different models. Among the most important are the Latent Diffusion Model (LDM) and the U-Net Convolutional Neural Network (CNN), which are core to supporting the EHR and CXR predictions. The open source EHRXDiff Github repository is available at https://github.com/dek924/EHRXDiff.

Latent Diffusion Modeling In 2022, state-of-the-art research from Rombach et al. (Rombach et al. 2022) introduced a way to train images using pretrained autoencoders, in the latent space, which greatly reduced computation and allowed for usability in multiple use cases, such as for generating high resolution in convolutional applications. The LDM used in this research is comprised of a Variational Autoencoder (VAE) (Kingma and Welling 2019) including encoder and decoder, and a type of CNN architecture called U-Net (Ronneberger, Fischer, and Brox 2015).

The authors reference Google's DeepMind van den Oord et al. (van den Oord, Vinyals, and Kavukcuoglu 2018) for the VAE component of the LDM, because van den Oord et al. developed a framework, VQ-VAE, that models in the latent space and learns an autoregressive prior, which is the approach the EHRXDiff authors took but replaced the autoregressive prior with a diffusion model where previous CXR imaging and medical EHR events are integrated as conditioning features for the prior.

From the authors, the LDM is trained to predict noise ϵ using the following Eq. (1) further explained in VAE section below.

$$\begin{aligned} z_t &= \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \\ where &: \epsilon \sim \mathcal{N}(0, I), \quad t \in \{1, \ldots, T\}. \end{aligned} \tag{1}$$

Variational Autoencoder A variational autoencoder (VAE) framework enables computation optimization in a latent space, using stochastic gradient descent (SGD) (Kingma and Welling 2019). The VAE generative model, "learns a joint distribution, with a prior distribution" (Kingma and Welling 2019).

In Eq. (1), z_t is the latent variable at timestep t and z_0 is equal to the $E_{\text{VAE}}(x)$ denoting the input image x. The $\epsilon_t \sim \mathcal{N}(0, I)$ is the *Gaussian* noise iteratively added

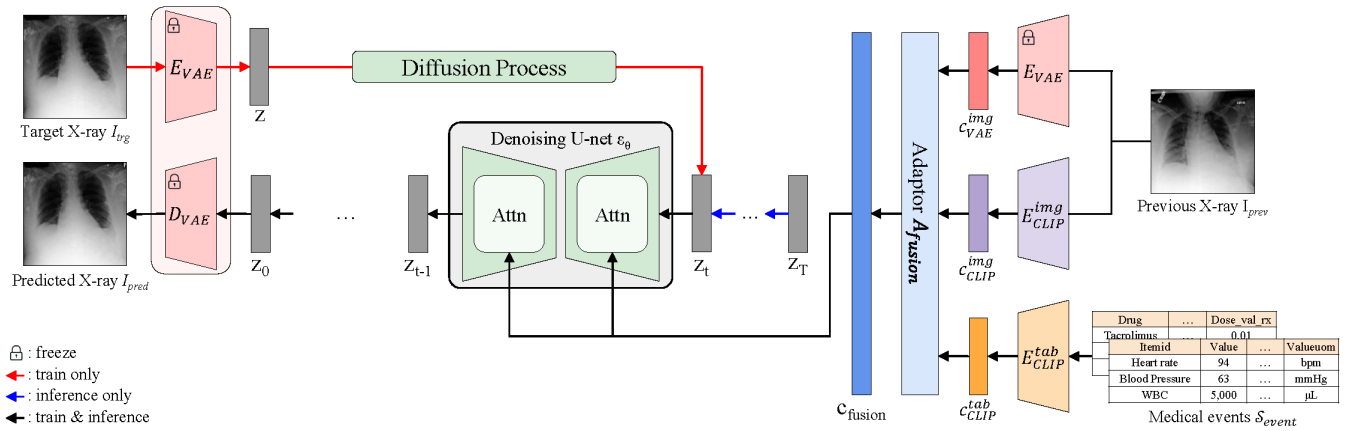

Figure 1: EHRXDiff framework.
(Kyung et al. 2025)

to z_0 during the diffusion process, while scalars $\sqrt{\bar{\alpha}_t}$ and $\sqrt{1-\bar{\alpha}_t}$ support how much signal is present in the noise at timestep t.

The E_{VAE} encoder is one of the two encoders used in the EHRXDiff framework that is responsible for higher resolution details such as the shape of organs, and location or size of lesions (Kyung et al. 2025).

Contrastive Language Image Pretraining The Contrastive Language-Image Pretraining (CLIP) model was developed by OpenAI in 2021 and is available as an open-source Python package at https://github.com/OpenAI/CLIP.

The CLIP encoders used in this research are identified as E_{CLIP}, either for image as $E^{\text{img}}_{\text{CLIP}}$ or tabular data as $E^{\text{tab}}_{\text{CLIP}}$. According to research, it is common for LDMs to use CLIP, being trained on "image-text pairs" (Kyung et al. 2025) however, at the time of the authors' research, there were "no existing pretrained models for image and tabular modalities in the medical domain" (Kyung et al. 2025) so the authors had to pretrain the CLIP model on $(\mathbf{S}_{\text{event}}, I_{\text{trg}})$, that would output as CLIP embeddings to be used in the adapter module.

Adapter for Multimodal Fusion The authors implemented an adapter module denoted as A_{fusion} to support embedding processes, and CXR image pair inconsistencies, such as misalignment between temporal images, which we learned can come from uncontrollable circumstances such as patient posture.

A clear novel solution, includes the approach the authors took for the fusion model, which was to instead of concatenating the previous image and tabular data, the authors use the implemented cross-attention. Now, with the fusion model, the final output embedding is fed to the U-Net cross attention component.

Convolutional Neural Net: U-Net The U-Net, a type of full CNN for image segmentation, was first developed by Ronneberger et al. (Ronneberger, Fischer, and Brox 2015) in 2015. The impetus for developing the U-Net architecture, which is shaped like a U, stemmed from the need to train deep neural networks without requiring thousands of labeled images. However, instead of using a full U-Net the authors modify the architecture to include a timestep layer and attention component given the diffusion process.

A typical U-Net architecture, consists of a left or contracting block which is the encoder, and right, or expansive block, which is the decoder. In the authors modified version, the left and right blocks are replaced with what are called residual blocks, which take in a timestep embedding or position, to keep time, and then proceed with convolutions, using not rectified linear unit (ReLU) but a sigmoid-weighted linear unit (SiLU) which is implemented in PyTorch which provides "non-zero gradients for negative inputs bypassing the 'dying ReLU' problem" (Huang and Schlag 2025). Additionally, the novelty here is that a typical CNN architecture will achieve local connectivity, (Singh et al. 2023), that is the convolution kernel moves over small patches of the image (Singh et al. 2023) to detect image segments however, with the added attention block there is no limitation to just local detection.

The authors describe the denoising of the U-Net mathematically as diffusion in Eq. (2) below, where ϵ_t is the normal Gaussian noise in the forward pass given latent noise z_t, the condition embedding $\tau_\phi(\mathbf{y})$ (previous CXR and EHR events), and timestep t.

$$\mathcal{L}_{\text{LDM}} = \mathrm{E}_{z_t \sim \mathcal{N}(0,1),\, t} \left\| \epsilon_t - \epsilon_\theta\big(z_t,\, \tau_\phi(\mathbf{y}),\, t\big) \right\|_2^2, \qquad (2)$$

(Kyung et al. 2025) where τ_ϕ is an embedding model for the condition $\mathbf{y}$. The U-Net architectural components are described below:

- **Timestep block:** takes the diffusion timestep t, applies a positional embedding, and processes it with a small `nn.Sequential` network.
- **Residual block:**
  - Input path: one normalization layer with 32 channels, a SiLU nonlinearity, and a 3×3 convolution.
  - Output path: one normalization layer with 32 channels, a SiLU nonlinearity, dropout, and a 3×3 convolution.

  - Depending on the stage, the block either downsamples (encoder, reducing spatial size) or upsamples (decoder, increasing spatial size).
- **Attention block:** applies a normalization layer, a 1×1 convolution, multi-head self-attention (Q, K, V projections), and a final 1×1 convolution.

The cross-attention mechanism noted in the multifusion adapter model section, is implemented here in the Attention block. The model fuses the condition embedding $\tau_\phi(\mathbf{y})$ with the intermediate U-Net feature representation ϵ_i through the attention layer as in Eq. 3 and Eq. 4:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right) V. \quad (3)$$

Per the author, queries, keys, and values are as follows:

$$Q = \epsilon_i W_Q^{(i)}, \qquad K = \tau_\phi(\mathbf{y}) W_K^{(i)}, \qquad V = \tau_\phi(\mathbf{y}) W_V^{(i)}, \quad (4)$$

(Kyung et al. 2025)

Embeddings The authors utilize embeddings throughout the model architecture, as with CLIP and VAE image embeddings, and table embeddings from the CLIP table encoder (Kyung et al. 2025). Additionally, it was noted that medical experts reviewed a large volume of data coming from the Intensive Care Unit (ICU) as part of EHR data, to support higher quality labeled data and utilized OpenAI's `text-embedding-ada-002` (OpenAI 2022) model to build free-text embeddings.

For this implementation, we saw an opportunity to take a more novel approach by bringing in Bidirectional Encoder Representations from Transformers (BERT) embeddings, as a replacement to OpenAI's. We found that the medical natural language processing (NLP) community has widely adopted several open-source, clinically focused BERT models, and we selected BioClinical ModernBERT, which is currently state-of-the-art.

The BioClinical ModernBERT model is an encoder-based transformer model that uses bidirectional self-attention, pretrained on the "largest biomedical and clinical corpus, with over 53.5B tokens, that outperforms existing clinical and biomedical encoders on four downstream tasks" (Sounack et al. 2025). We use the base pretrained version of BioClinical ModernBERT from Huggingface at https://huggingface.co/thomas-sounack/BioClinical-ModernBERT-base.

See Appendix for performance metrics showing BioClinical ModernBERT with the top scores across classification, and NER tasks.

Retrieval Augmented Generation The retrieval-augmented generation (RAG) model was first developed in 2020, by researchers from Facebook AI, University College of London and New York University, providing a way to "combine pre-trained parametric and non-parametric memory for language generation" (Lewis et al. 2021).

We incorporated a RAG module to support sparse EHR events utilizing Facebook AI Similarity Search (FAISS) (Douze et al. 2025) vector database for high dimensional embeddings, creating efficiencies for RAG to locate most relevant text for a large language model (LLM). Here, FAISS retrieves clinically relevant symptom descriptions and the LLM produces a contextualized prompt. This improved text representation becomes the input for BioClinical ModernBERT. See Appendix for RAG-BioClinical ModernBERT model architecture.

Table 2: Hyperparameters for embedding and retrieval components.

Model	Learning Rate	Batch Size	Hidden Size	Dropout
BioClinical ModernBERT	–	64	768	0.0
RAG Contriever	2×10^{-5}	64	768	0.1
CLIP Image (ViT-B/32)	1×10^{-4}	32	768	0.0
CLIP Text (2-layer)	1×10^{-4}	32	1536→768	0.1

Training

For training, the model pipeline was run in the sequential order below:

1. **RAG with MIMIC-IV**: Using first 1,000k lines in isolated training
2. **RAG with BioClinical ModernBERT**: Combines RAG with Batch BERT embeddings saved to Hierarchical Data Format (HDF5) file
3. **CLIP with RAG-BioClinical ModernBERT**: Combines HDF5 file with images, uses pretrained model from MaCheX (Weber et al. 2023). CLIP encoders are Base Vision Transformer with 32×32 patch size (ViT-B/32) and 2 layer Transformer encoder for tabular data (Kyung et al. 2025). CLIP dimension is 768.
4. **Adapter Fusion Module**: Applies linear projection layer, with autoencoded (I_{prev})
5. **U-Net CNN**: Uses convolutional neural network with up or down sampling and cross-attention
6. **LDM**: Uses **Cheff** state-of-the-art diffusion model settings. (Weber et al. 2023)

Hyperparameters We trained the U-Net and LDM at 100 epochs with batch size of 128 using AdamW optimizer (Kyung et al. 2025). Key hyperparameters for the embedding and retrieval components are summarized in Table 2 with full hyperparameters for CLIP in Table 3 in Appendix.

Computational Requirements

During our development phase, we explored the EHRXDiff repository in Github to understand the model architecture. From there we separated out major components of code into Google Colab Pro+ notebooks running in Google Drive with 100GB of storage. We performed all data processing and text based embeddings training on local machines with the following setup below.

The RAG-BioClinical ModernBERT training took the longest given the compute intensity of 3k text image paired samples with EHR events. The max sequence length for each

patient text image pair was 1024. We believe the I/O bottleneck was due to Google Drive read and write processing, in addition to the HDF5 file operations and compute limitations of Google Colab Pro+ which is a shared GPU resource platform.

The embeddings training was using Google Colab Pro+ with A100 GPU. Once it came time to run the LDM and U-Net model pipeline end to end, we moved the codebase to a free online platform called Weights & Biases available at `wandb.ai`, where we were able to test and evaluate our predicted CXR image output. Total runtime for LDM and U-Net model was 23h 59m 29s.

- **Operating System**: Macbook Pro, Tahoe 26.1
- **RAM**: Google Colab Pro+ environment 25GB
- **CPU**: Apple M4 Pro
- **Python Version**: 3.12

Evaluation

For our evaluation metrics, we focused on overall image quality. The original authors targeted three components: 1) preservation of medical information, 2) preservation of demographic information and 3) image quality, however, due to time constraints and with approval from our teaching assistant, we focused only on the third.

The results from this experiment can be found in the `Evaluation.ipynb` notebook in our shared Google Drive `dl4h/PredictingChestXRays` directory. In this workflow, we follow the authors' procedure, utilizing the eval.py script in the scripts and FID folder from EHRXDiff Github repository.

The evaluation metric used was Fréchet Inception Distance. We were limited in terms of test and train data due to RAG+BioClinical ModernBERT accruing long training times, and for 170 rows from our test set we achieved a FID score of 18.484 which is higher than the authors achieved.

Results

Please see Figure (2) for model output from our predicted CXR images and Figure (3) for our model dashboard monitor after running the end to end pipeline.

(a) Predicted CXR 1 (b) Predicted CXR 2 (c) Predicted CXR 3

Figure 2: Example predicted chest X-ray images from our model.

Compare and Contrast to Original Research

Our reproduction of the EHRXDiff framework followed the authors' code and data pipeline and successfully generated future CXR images. However, on a reduced test cohort we obtained a higher (worse) FID than reported in the original paper, and we evaluated only image quality, not preservation of medical or demographic information. Several factors likely contributed to these discrepancies:

- **Smaller scale and shorter training.** Due to Google Colab Pro+ limits, we used subsets of MIMIC-IV/MIMIC-CXR-JPG, trained RAG on only the first 100k text entries, and reached only $\sim$4500 global steps for the LDM (vs. the authors' default of at least 5000).
- **Modified EHR embedding pipeline.** We replaced the original OpenAI text embeddings with a RAG + BioClinical ModernBERT setup. This improves privacy and practicality but changes the conditioning space, so our results are not a one-to-one replication of the original configuration.
- **Infrastructure and preprocessing differences.** Reliance on Google Drive + HDF5 and shared GPUs/TPUs introduced I/O bottlenecks and small differences in cohort construction and EHR sequence truncation, which likely contributed to the observed gap in performance.

Discussion

Implications of Experimental Results

The implications of the experimental results are far reaching, given we were able to produce high quality CXR images per patient, using the EHRXDiff code base with our modifications for novelty. The primary goal hereafter would be to ensure that the CXR predicted image output had relevance in a patient's real life which would require partnership with hospitals and or organizations that would allow experimentation and testing on real patient data using the modified EHRXDiff framework.

The original paper and author research was reproducible, in large part due to the open source Github repository code base. In fact, without the Github repository, the time to produce predicted CXR output would have been greatly increased because of the complexity of the EHRXDiff pipeline and minimal documentation. To note, the research paper was thorough in detail on implementation, which we relied on heavily in addition to reading through the code and making assumptions.

Challenges

The most challenging areas of reproducing the EHRXDiff framework included:

- **Limited compute and memory.** Google Colab Pro+ and our local machines did not provide enough stable GPU memory or runtime for long training runs, especially for the full LDM pipeline and the RAG + BioClinical ModernBERT components.
- **Inability to use full cohorts.** Because of these resource and storage constraints, we were unable to work with the full MIMIC-IV and MIMIC-CXR-JPG datasets and instead had to downsample and work with smaller subsets to move forward with experimentation.

Figure 3: AI platform model training dashboard.

Recommendations

To improve reproducibility in future work, we recommend heavily documenting the codebase so that readers can understand the model sequencing and the architecture of each component.

Author Contributions

- Jesus Jimenez supported the data processing (image and text) and integration efforts with MIMIC-IV and MIMIC-CXR-JPG into our Google Drive, setting up our EHRXDiff database, in addition to running a full BERT implementation for exploration and, finally, the PyHealth pull request.
- Shang Lu supported the model architecture setup that included the U-Net, VAE/CLIP and LDM, and FAISS+RAG, in addition to setting up the Weights & Biases online AI training platform to enable full model training end to end.
- Mariam Joan supported writing of the Proposal and Final Project research paper, setup and training of the BioClinical-ModernBERT-base, implementation of the BERT+RAG embeddings training, in addition to the model evaluation.

Acknowledgments

We would like to acknowledge the support of teaching assistants, Chufan Gao and Trisha Das from the CS598 Deep Learning for Healthcare course during the Fall 2025 semester.

Appendix

	Model	*Context length*	Classification		Named Entity Recognition		
			ChemProt	Phenotype	COS	Social History	DEID
Base	BioBERT (Lee et al., 2019)	*512*	89.5	26.6	94.9	55.8	74.3
	Clinical BERT (Alsentzer et al., 2019)	*512*	88.3	25.8	95.0	55.2	74.2
	BioMed-RoBERTa (Gururangan et al., 2020)	*512*	89.0	36.8	94.9	55.2	81.1
	Clinical-BigBird (Li et al., 2022)	*4096*	87.4	26.5	94.0	53.3	71.2
	Clinical-Longformer (Li et al., 2022)	*4096*	74.2	46.4	**95.2**	56.8	82.3
	Clinical ModernBERT (Lee et al., 2025)	*8192*	86.9	54.9	93.7	53.8	44.4
	ModernBERT - base (Warner et al., 2024)	*8192*	89.5	48.4	94.0	53.1	78.3
	BioClinical ModernBERT - base (ours)	*8192*	89.9	58.1	95.1	**58.5**	82.7
Large	ModernBERT - large (Warner et al., 2024)	*8192*	90.2	58.3	94.4	54.8	82.1
	BioClinical ModernBERT - large (ours)	*8192*	**90.8**	**60.8**	95.1	57.1	**83.8**

Figure 4: BioClinical ModernBERT median performance in classification and NER tasks.

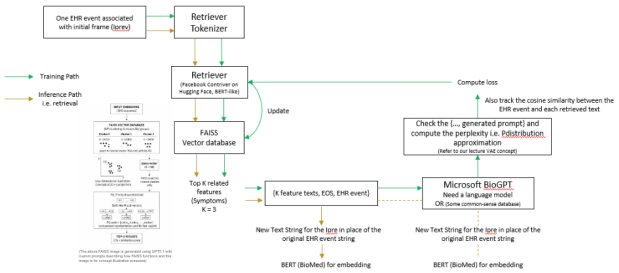

Figure 5: RAG-BioClinical ModernBERT diagram.

Table 3: Hyperparameters for CLIP encoders.

Stage	Shape
CLIP image encoder (ViT-B/32)	
Input image	$(B, 3, 256, 256)$
Patch embed (Conv2d)	$(B, 768, 8, 8)$
Flatten	$(B, 64, 768)$
Prepend CLS token	$(B, 65, 768)$
+ Positional embeddings	$(B, 65, 768)$
12 Transformer blocks	$(B, 65, 768)$
LayerNorm	$(B, 65, 768)$
Extract CLS	$(B, 768)$
CLIP text encoder	
Input EHR sequence	$(B, \texttt{seq\_len}, 1536)$
Transformer layer 1	$(B, \texttt{seq\_len}, 1536)$
Transformer layer 2	$(B, \texttt{seq\_len}, 1536)$
FC projection	$(B, \texttt{seq\_len}, 768)$
Average pooling	$(B, 768)$

Table 4: Hyperparameters for LDM U–Net and autoencoder.

Stage	Shape
LDM U–Net	
Noisy latent input	$(B, 3, 64, 64)$
First conv	$(B, 224, 64, 64)$
Enc. Level 0 (2×ResBlock)	$(B, 224, 64, 64)$
Enc. Level 0 Downsample	$(B, 224, 32, 32)$
Enc. Level 1 (Res+XA)×2	$(B, 448, 32, 32)$
Enc. Level 1 Downsample	$(B, 448, 16, 16)$
Enc. Level 2 (Res+XA)×2	$(B, 896, 16, 16)$
Enc. Level 2 Downsample	$(B, 896, 8, 8)$
Enc. Level 3 (Res+XA)×2	$(B, 896, 8, 8)$
Bottleneck (Res+XA+Res)	$(B, 896, 8, 8)$
Dec. Level 3 concat skips	$(B, 1792, 8, 8)$
Dec. Level 3 (Res+XA)	$(B, 896, 8, 8)$
Dec. Level 3 Upsample	$(B, 896, 16, 16)$
Dec. Level 2 concat skips	$(B, 1792, 16, 16)$
Dec. Level 2 (Res+XA)	$(B, 896, 16, 16)$
Dec. Level 2 Upsample	$(B, 896, 32, 32)$
Dec. Level 1 concat skips	$(B, 1344, 32, 32)$
Dec. Level 1 (Res+XA)	$(B, 448, 32, 32)$
Dec. Level 1 Upsample	$(B, 448, 64, 64)$
Dec. Level 0 concat skips	$(B, 672, 64, 64)$
Dec. Level 0 (2×ResBlock)	$(B, 224, 64, 64)$
Output Norm + SiLU	$(B, 224, 64, 64)$
Output conv (noise)	$(B, 3, 64, 64)$
Autoencoder (encoder)	
Input image	$(B, 3, 256, 256)$
First conv	$(B, 128, 256, 256)$
Down 1 (Res + Downsample)	$(B, 128, 128, 128)$
Down 2 (Res + Downsample)	$(B, 256, 64, 64)$
Down 3 (Res)	$(B, 512, 64, 64)$
Self-attn (Res + Attn + Res)	$(B, 512, 64, 64)$
Latent conv	$(B, 6, 64, 64)$
Mean / logvar	$(B, 3, 64, 64)$
Sampled z	$(B, 3, 64, 64)$
Autoencoder (decoder)	
Input latent z	$(B, 3, 64, 64)$
First conv	$(B, 512, 64, 64)$
Self-attn (Res + Attn + Res)	$(B, 512, 64, 64)$
Up 1 (Res + Upsample)	$(B, 512, 128, 128)$
Up 2 (Res + Upsample)	$(B, 256, 256, 256)$
Level 3 ResBlock	$(B, 128, 256, 256)$
Output norm	$(B, 128, 256, 256)$
Output conv (image)	$(B, 3, 256, 256)$

References

Daeun Kyung, T. K. E. C., Junu Kim. 2025. Github: Towards Predicting Temporal Changes in a Patient's Chest X-ray Images based on Electronic Health Records (CHIL 2025). https://github.com/dek924/EHRXDiff.

Douze, M.; Guzhva, A.; Deng, C.; Johnson, J.; Szilvasy, G.; Mazaré, P.-E.; Lomeli, M.; Hosseini, L.; and Jégou, H. 2025. The Faiss library. arXiv:2401.08281.

Huang, A. H.; and Schlag, I. 2025. Deriving Activation Functions Using Integration. arXiv:2411.13010.

Kingma, D. P.; and Welling, M. 2019. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4): 307–392.

Kyung, D.; Kim, J.; Kim, T.; and Choi, E. 2025. Towards Predicting Temporal Changes in a Patient's Chest X-ray Images based on Electronic Health Records. arXiv:2409.07012.

Kyunghoon Hur, J. K. J. K. M. J. L. E. C. S.-E. M. Y.-H. K. L. A. E. C., Jungwoo Oh. 2022. GenHPF: General Healthcare Predictive Framework with Multi-task Multi-source Learning. In *GenHPF: General Healthcare Predictive Framework with Multi-task Multi-source Learning*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; tau Yih, W.; Rocktäschel, T.; Riedel, S.; and Kiela, D. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401.

(NEMA), N. E. M. A. 2025. Digital Imaging and Communications in Medicine. https://www.dicomstandard.org/concepts.

OpenAI. 2022. text-embedding-ada-002. https://platform.openai.com/docs/models/text-embedding-ada-002.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv:1505.04597.

Singh, Y.; Farrelly, C.; Hathaway, Q. A.; Choudhary, A.; Carlsson, G.; Erickson, B.; and Leiner, T. 2023. The Role of Geometry in Convolutional Neural Networks for Medical Imaging. *Mayo Clinic Proceedings: Digital Health*, 1(4): 519–526.

Sounack, T.; Davis, J.; Durieux, B.; Chaffin, A.; Pollard, T. J.; Lehman, E.; Johnson, A. E. W.; McDermott, M.; Naumann, T.; and Lindvall, C. 2025. BioClinical ModernBERT: A State-of-the-Art Long-Context Encoder for Biomedical and Clinical NLP. arXiv:2506.10896.

SunLab. ???? Pyhealth Python Library and Toolkit for Accessing Healthcare Applications. https://github.com/sunlabuiuc/PyHealth.

van den Oord, A.; Vinyals, O.; and Kavukcuoglu, K. 2018. Neural Discrete Representation Learning. arXiv:1711.00937.

Weber, T.; Ingrisch, M.; Bischl, B.; and Rügamer, D. 2023. Cascaded Latent Diffusion Models for High-Resolution Chest X-ray Synthesis. arXiv:2303.11224.