Cairo University
Faculty of Engineering
Department of Computer Engineering

X-Reporto

A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by
Ahmed Sabry

Supervised by
Dr. Yahia Zakaria

July 2024

Contacts

Team Members

Name	Email	Phone Number
Ahmed Sabry	ahmed.ahmed017@eng-st.cu.edu.eg	+2 01067371115

Supervisor

Name	Email	Number
Dr. Yahia Zakria	yahiazakaria13@gmail.com	+2 01xxxxxxxxx

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

To address the complexities of automated medical report generation, X-Reporto employs advanced language models, specifically leveraging the power of transformers and GPT-2. Language models like GPT-2 are designed to understand and generate human-like text, making them ideal for creating detailed and coherent medical reports. Transformers, the underlying architecture of GPT-2, use self-attention mechanisms to process and relate different parts of the input data effectively. This capability allows the model to generate contextually accurate and fluent text based on the features extracted from medical images.

1.1. Motivation and Justification

The motivation for incorporating transformers and GPT-2 in X-Reporto stems from their proven effectiveness in handling sequential data and generating high-quality text. Traditional methods, such as CNN-RNN architectures, have limitations in capturing long-range dependencies and often struggle with maintaining coherence in generated text. Transformers overcome these challenges by utilizing self-attention mechanisms, which enable the model to consider the entire input sequence simultaneously. GPT-2, a transformer-based model, has demonstrated exceptional performance in natural language processing tasks, making it a suitable choice for generating detailed and accurate medical reports. By integrating GPT-2, X-Reporto can produce comprehensive reports that accurately reflect the findings from chest x-rays, significantly aiding radiologists in their diagnostic processes.

1.2. Document Organization:

1. **Introduction:**
   This chapter summarizes the report, giving a brief summary of the problem addressed by our study, project objectives, and problem definition.
2. **Literature Survey:**
   This chapter provides a comprehensive review of the existing literature on transformer architecture and language model in general and gpt2 specifically.
3. **System Design and Architecture:**
   This chapter describes the design and architecture of Language model and backend system, including the approach taken to develop the tool, its key features, and the technologies used in its development.
4. **System Testing and Verification:**
   This chapter presents the results of the testing and verification process for language model, including its performance, accuracy, and efficiency.
5. **Conclusions and Future Work:**
   This chapter outlines Language model significant results. It also identifies expected future work and development areas.
6. **References:**
   This chapter lists all the references cited in the report.

Chapter 2: Literature Survey

2.1 Transformer

2.1.1. Overview

The Transformer architecture revolutionized the field of natural language processing by utilizing a mechanism known as self-attention, which allows the model to weigh the importance of different words in a sentence when making predictions. This architecture facilitates parallel processing of data, leading to significantly faster training times compared to traditional recurrent neural networks. Building on the Transformer framework, GPT-2, a large-scale, unsupervised language model pre-trained on diverse internet text. GPT-2 excels at generating coherent and contextually relevant text by predicting the next word in a sequence, given all the previous words within the context. This capability makes GPT-2 highly effective for a wide range of natural language understanding and generation tasks. My work leverages GPT-2's advanced language generation capabilities to produce detailed and accurate medical reports from visual features extracted from X-rays, ensuring that the generated text is both medically accurate and contextually appropriate.

2.1.2. Architecture

GPT-2, or Generative Pre-trained Transformer 2, is built upon the Transformer architecture illustrated in figure 3.12.2 , which relies on a self-attention mechanism to model long-range dependencies in text. The core of GPT-2's architecture consists of multiple layers of transformer decoders, each comprising multi-head self-attention and feed-forward neural networks. The model employs layer normalization and residual connections to improve training stability and performance. Each self-attention layer allows GPT-2 to consider the context from all positions in the input sequence simultaneously, enabling it to generate coherent and contextually relevant text. With 1.5 billion parameters for large or 170 million parameters for small, GPT-2 is capable of understanding and generating a wide range of human-like text, thanks to its extensive pre-training on diverse internet text data. This vast pre-training allows GPT-2 to capture a broad spectrum of linguistic patterns and nuances, making it highly effective for tasks such as text generation, summarization, translation, report generation and more. By fine-tuning GPT-2 on specific domains, such as medical literature, its performance can be further optimized for generating specialized content, like detailed medical reports from visual data.We will discuss the building blocks of the network in the following subsections.

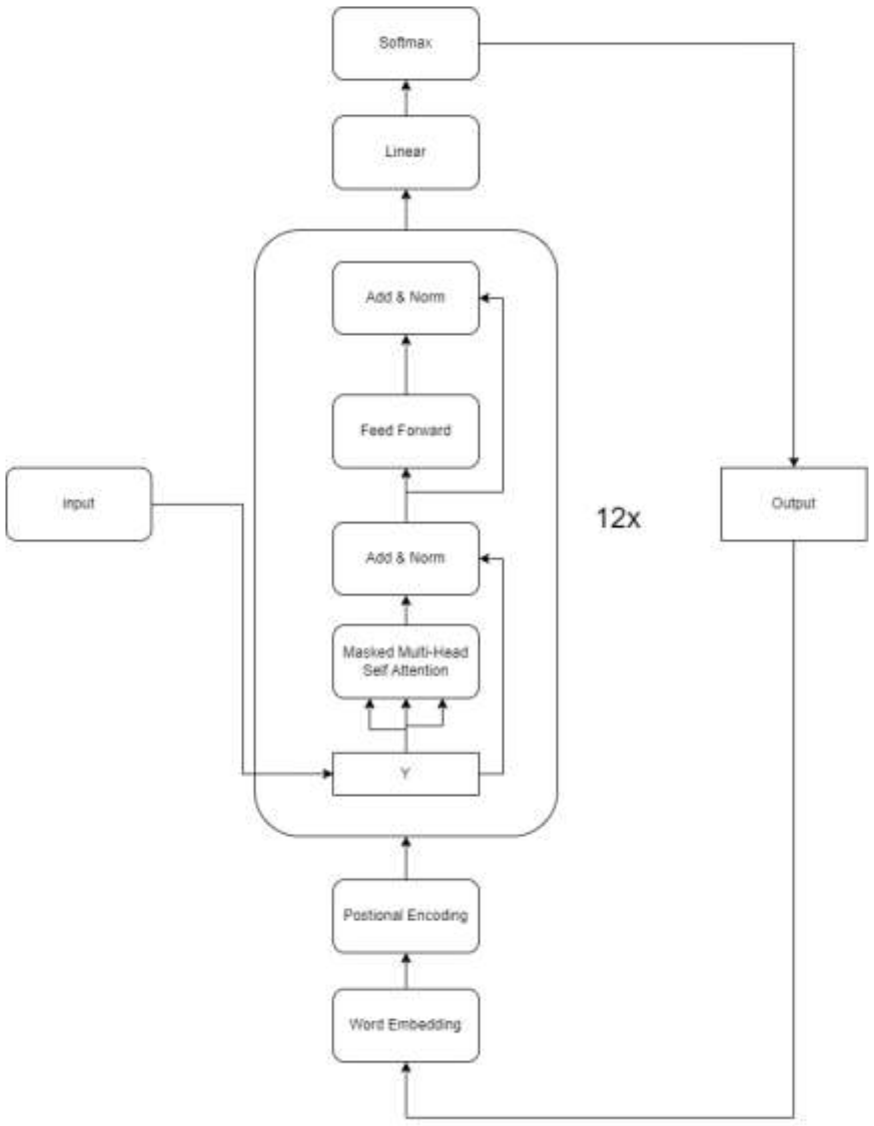

Figure 1 Transformer Decoder Architecture

2.1.2.1 Word Embedding

We used a word embedding of a pre-trained GPT2 model that trained self-supervised on medical articles so that we could understand medical words and generate good word representations of them.

2.1.2.2 Positional Encoding Embedding

We used the position information of the current token to help the model predict the next word correctly and calculate correct attention scores for other words in a sentence.

2.1.2.3 Masked Multi-Head Self-Attention

In this step, the model uses masked multi-head self-attention mechanisms to focus on different parts of the input sequence simultaneously. The masking ensures that the prediction for a given position only depends on the known outputs at prior positions, maintaining the autoregressive property. The equation of the normal self-attention equation is shown below

$$SA(Y) = softmax\left((YW_q)(YW_k)^T\right)(YW_v)$$

where Y represents the token embeddings and Wq, Wk, Wv are the query, key, and value projection parameters

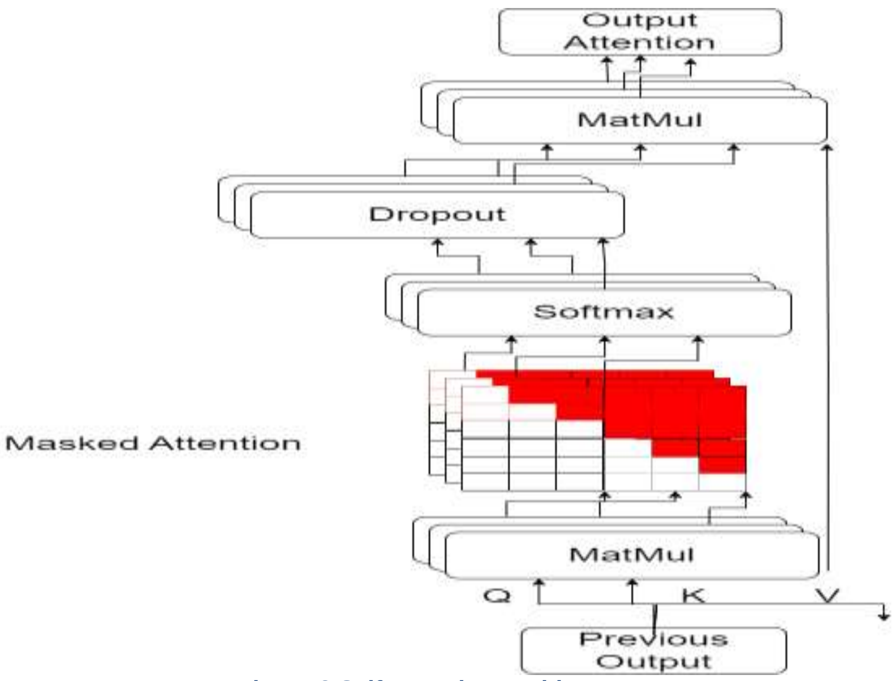

Figure 2 Self Attention Architecture

2.1.2.4 Residual Connections and Normalization

After the self-attention mechanism, the model adds a normalization layer to the input and then applies the self-attention mechanism then the dropout layer, and finally adds the input and output of the dropout layer, These steps stabilize and speed up the training process by maintaining consistent input distributions across layers and prevent overfitting of model and reduce dependency between blocks.

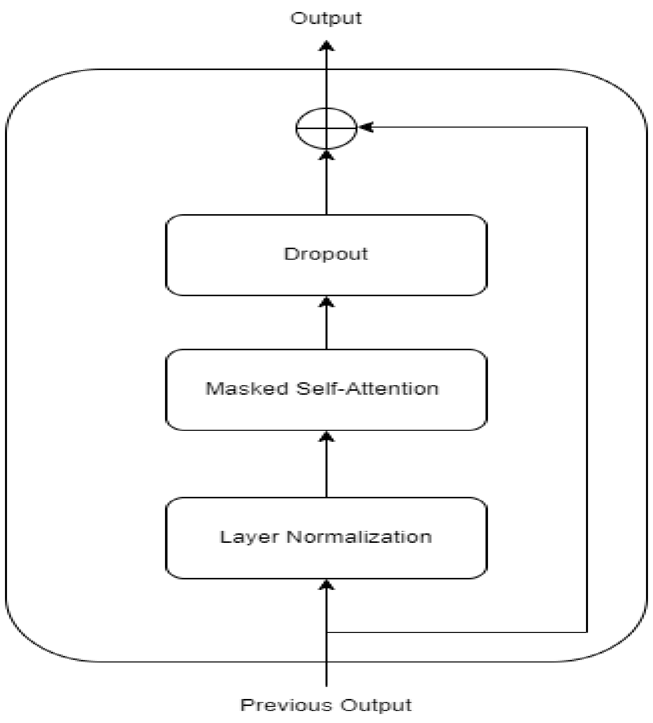

Figure 3 Residual Block

2.1.2.5 Feed Forward

The output from the addition and normalization step is then passed through a feed-forward neural network. This network typically consists of two linear layers with a ReLU activation function in between, allowing the model to learn complex transformations.

2.1.2.6 Softmax

Finally, the model applies a softmax function to the output logits, converting them into probabilities. This step is crucial for generating the most likely next word in the sequence, completing the process of text generation based on the input visual features.

3.15. Comparative Study of Previous Work

Recent advancements in automatic radiology report generation have increasingly focused on Transformer-based architectures. Early approaches primarily utilized CNN-RNN frameworks, but the shift towards Transformers has highlighted their effectiveness in integrating visual and textual features. Studies have adapted the standard Transformer by incorporating relation memory units or memory matrices to enhance cross-modal interactions. Various methods have also been proposed to improve attention to abnormal regions, such as aligning visual features with disease tags and leveraging medical knowledge graphs. In contrast, our approach emphasizes the direct extraction of visual features from abnormal regions through object detection, supplemented by an abnormality classification module to encode relevant information.

Furthermore, literature has explored hierarchical strategies for generating coherent reports by breaking down the task into manageable steps. Similar to prior work, we also decompose report generation into multiple stages, with a particular focus on region-level feature extraction. Unlike other methods that employ multi-head transformers for specific anatomical regions, our approach utilizes object detection to generate region-level sentences through a shared transformer decoder. While requiring more supervision, this method is less technically complex, enhancing interactivity and transparency in report generation.

3.15.1 Comparison With Previous Work

Aspect	Previous Work	Our Approach
Architecture	CNN-RNN, adapted Transformers	Transformer with object detection
Feature Extraction	General image-level features	Specific region-level features
Attention Mechanism	Enhanced with memory units or graphs	Direct extraction from abnormal regions
Report Generation Method	Hierarchical or multi-head transformers	Shared transformer decoder
Complexity	Often more complex due to multiple heads	Less technically complex, more interactive
Supervision Requirement	Varies	Requires more supervision
Transparency	Limited transparency	Higher degree of transparency and explainability

Table 3.14.1 1 Comparison to X-Reporto Competitors

Chapter 3: System Design and Architecture

In this chapter, we provide a comprehensive overview of the design and architecture of the project, detailing the steps taken to develop a robust system for generating medical reports from X-ray images. The project is structured around a series of interconnected modules, each contributing to the overall functionality. By employing a systematic approach, we designed the system to ensure efficient processing, cleaning image, detecting anatomical regions, accurate outputs of medical reports, and efficient diagnosis . This chapter will discuss the methodologies employed, from the initial design concepts to the implementation of each module, allowing readers to understand and replicate our work effectively.

3.1. Overview and Assumptions

In designing the system, we aimed to create an integrated workflow that facilitates the transformation of raw medical images into detailed reports. The architecture is built on five primary modules: image denoising, object detection, selection binary classifiers, language model generation for medical reporting, and a heatmap module for disease classification. Each module operates cohesively, with defined inputs and outputs, ensuring smooth data flow throughout the system. Key assumptions include the availability of high-quality medical images, the capability of the object detector to accurately identify anatomical features, and the reliability of the language model in generating coherent text. These assumptions underpin our design choices and guide the development process.

3.2. System Architecture

The architecture of the system is structured around a modular design that enhances scalability and maintainability. The overall system is represented as a block diagram that shows interactions between the different modules and their respective functions.

As we have 6 separated modules, each module has specific functionality and expects certain modules output and generates correct output for next modules. Follow of our modules is that

1. We first should remove any noise in the x-ray chest.
2. We detect anatomical regions in chest and generate visual features for them
3. We Select some regions that have findings in it.
4. From these visual features we generate corresponding findings in it including abnormality and diagnosis.
5. Finally, we generate predictions for possible diseases and its location in x-ray.

3.2.1. Block Diagram

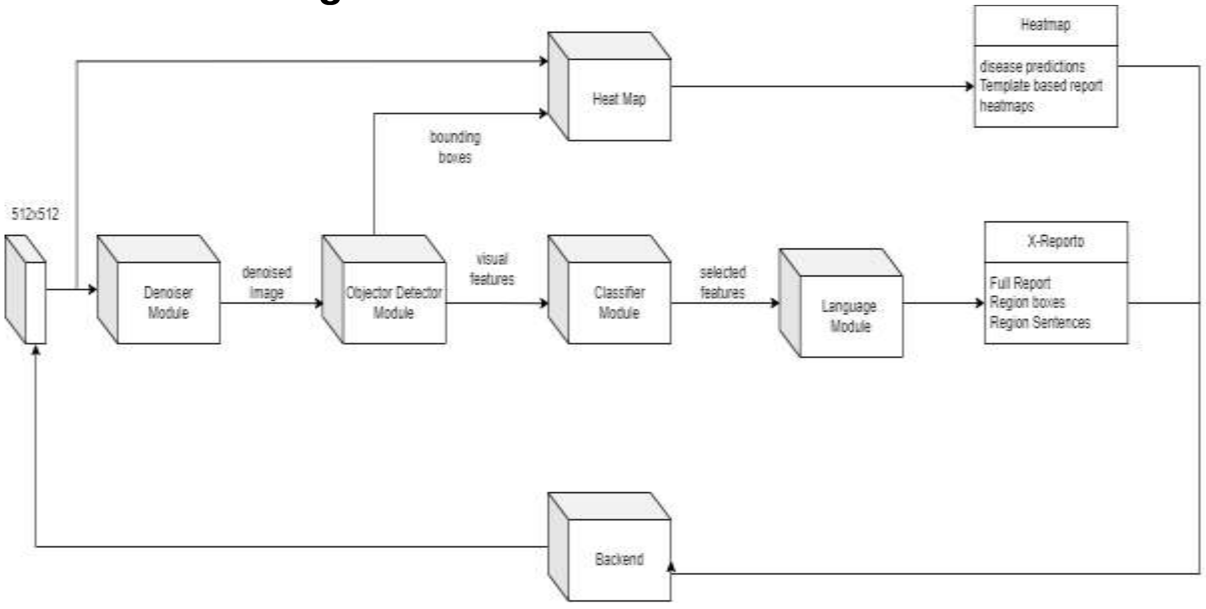

Figure 4 X-Reporto System Block Diagram

3.3. Language Model

3.3.1. Functional Description

To generate an accurate report for an X-ray that describes it correctly and includes all medical information and abnormalities present in the chest, we need a language model capable of producing reports based on visual features obtained from object detection and classifiers. This model should generate human-like text that comprehensively describes all the findings in the X-ray.

We built our model upon gpt2 architecture, we started with gpt2 small basic structure and modified it according to our problem requirements and constraints. We modified the gpt2 attention mechanism and added a new attention mechanism that incorporates visual feature information along with the attention of generated tokens to generate output scores of attention.

Then we added an encoder model to apply transformation from visual features generated by the object detector to text features that can be understood by language models and act as intermediate information carrying medical information and abnormality exists in each region in the x-ray chest.

Finally, we added gradient checkpointing for each block so that we could fix the memory requirement of the model while training so that it can train the model with higher batch sizes and fit our requirement so that we can train locally or on a server with high resources and gain the best performance of the model.

3.3.2. Modular Decomposition

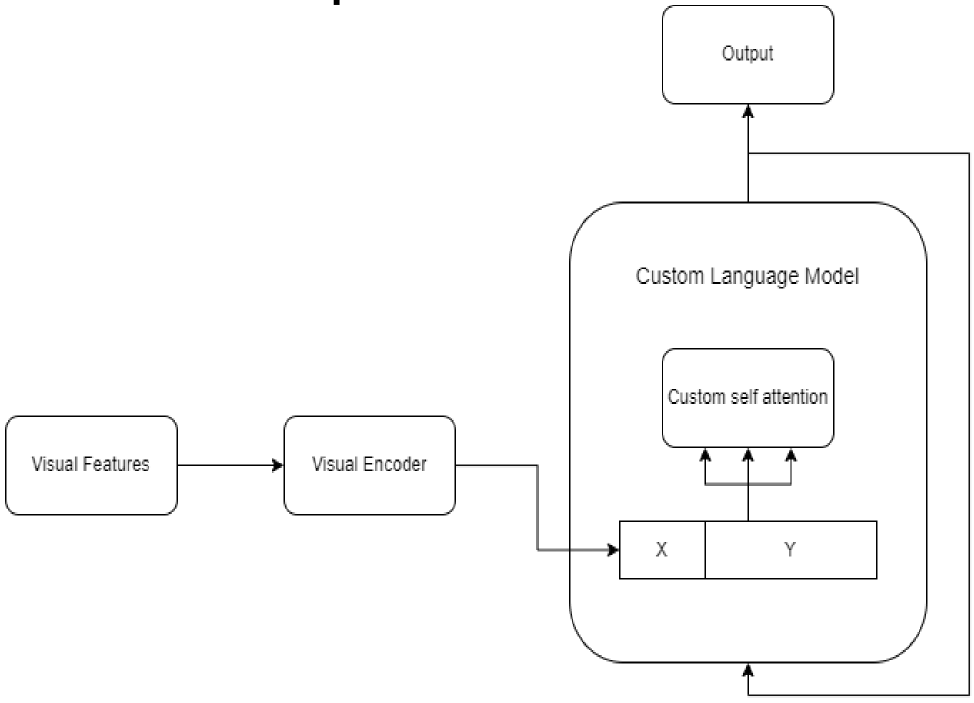

Figure 5 3 Language Model Block Diagram

In Our model we have added new blocks to fix our problems on how to make transformer decoder works on visual features

1. Visual Encoder model
2. Custom Self attention block

3.3.2.1. Visual Encoder

Image Transformation Encoder used for transforming visual features from image space to text space features can be understood by language model to generate text from.We used two linear layers where the input is a visual feature vector of length 1024 and the output is also a vector of the same length.

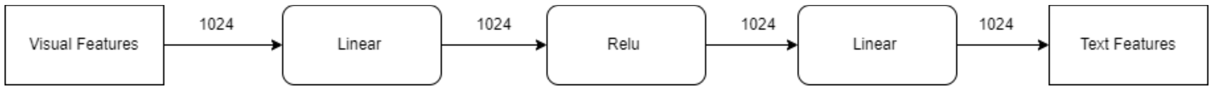

Figure 5.8.2.1 Visual Encoder Block Diagram

3.3.2.2. Custom Multi-Head Self-attention

In this step, the model uses masked multi-head self-attention mechanisms to focus on different parts of the input sequence simultaneously. The masking ensures that the prediction for a given position only depends on the known outputs at prior positions, maintaining the autoregressive property.
Normal self-attention equation is $SA(Y) = softmax\left((YW_q)(YW_k)^T\right)(YW_v)$
where Y represents the token embeddings and Wq, Wk,
Wv are the query, key, and value projection parameters.
To condition the language model on the region's visual features, we use pseudo-self-attention to directly inject the region's visual features into the self-attention of the model,
Pseudo self-attention equation is $PSA(Y) = softmax\left((YW_q)\left[\square \genfrac{}{}{0pt}{}{XU_k}{YWk_{\square}}\right]^T\right)(YW_v)$
where X represents the region's visual features, and Uk and Uv are the corresponding (newly initialized) key and value projection parameters. This allows text generation conditioned on both previous tokens and region visual features.

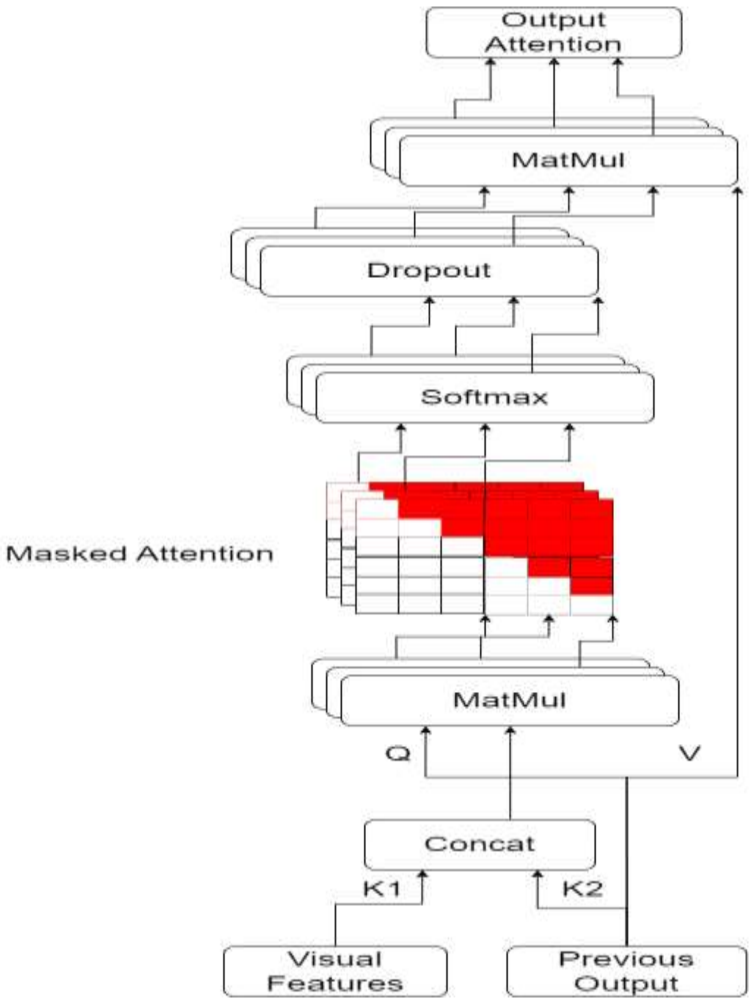

Figure 6 Custom Self Attention

3.3.3. Design Constraints

Visual Features Encoding: Our first problem was how to integrate visual features generated by the object detector and classifier into our language model gpt2 and maintain consistent features that both models can learn correctly so that object detection can correctly use these features in generating bounding boxes and at the same time incorporate medical information and abnormality in it, and language model can understand these features and get benefits from it then generate auto regressively output sentence that truly describes regions.

Our solution with Image transformation uses two linear layers with relu between them, these layers should be responsible for transforming visual features and medical information into text representation as an encoding layer that language model understands and can use to generate sentences using a decoder block.

Visual Features Integration: The second constraint was how to integrate information of visual features in the attention mechanism so as to generate the next tokens from previous tokens and visual features and calculate the attention score.
Our solution was using a pseudo-self-attention mechanism that incorporates visual information as key input concatenated with the key of token information and then applies a normal self-attention mechanism.
Pseudo self-attention equation is $PSA(Y) = softmax\left((YW_q)\begin{bmatrix} XU_k \\ YW_k \end{bmatrix}^T\right)(YW_v)$.
This reduces model computation instead of applying two attention blocks, one for previous tokens and one for visual features, and at the same time applies correct conditioning on these inputs.

Model Size: The third constraint was the model size, what number of layers to be used that can achieve the best performance, not overfitted, and can be trainable using not many resources.
We used 12 layers of attention block in the above diagrams. these helped us to train our full pipeline with minimal resources we have. The size of the model is around 170M parameters and 700 MB in memory.

3.3.4. Methodologies

3.3.4.1. Training

To train a full pipeline of object detectors, classifiers, and language model with at least one example “batch size = 1”, and one example contains 29 sentences which correspond to that input to language model is 29 examples which are very huge and can not fit in our memory.
We fixed this problem by applying gradient checkpoint which is a technique used in deep learning to reduce memory usage during training of neural networks, especially those with large memory requirements. Normally, deep neural networks store intermediate activations (tensors) during forward propagation to use during backpropagation for computing gradients. This storage can become quite memory-intensive, particularly in models with many layers or parameters which occurs in our case in gpt2.
Gradient checkpointing addresses this issue by selectively storing only a subset of intermediate activations, known as checkpoints, rather than all of them. During backpropagation, the network can then recompute the omitted activations dynamically as needed, trading off computation for reduced memory consumption.
We applied Gradient checkpointing on each layer block of 12 layers of gpt2. so that storing only the inputs and outputs of each block. With this enhancement, we reduced the memory requirements of the gpt2 model by 1/12 factor of memory requirements of gpt2 without gradient checkpointing but we doubled the training time of the model as we nearly applied forward pass twice.
We were able to apply batch size = 4 for the object detector and batch size = 128 for

language model sentences without any accumulating gradients as in object detector. trained for **2 epochs** with a very small **learning rate** equals 10^{-5} We used cross-entropy loss as our loss function and Adam as our optimizer.
Not only was the language model trained, but the full pipeline was trained. We applied a weight mechanism for each model while training where the object detector weight equals 1, classifier weight equals 5, and language model equals 2. These parameters needed to focus more on the selection of regions and abnormality in features then focus on the language model and finally object detector where its weights were trained 3 times.

3.3.4.2. Post Processing

After Obtaining the report results we found redundant repeats sentences in the description for each region.After report generation, we have k selected regions have k sentences that describe the finding in each anatomical regions of chest that detected by object detector, since regions overlaps with each other, we found out that model outputs nearly same finding of overlapped regions therefore we cannot just concatenate sentences to form report but we should select unique sentences to form final report.We tried different solutions to fix this problem either software or ai.

Software Solution: We tried to filter duplicate sentences by using string matching to remove copies of these sentences. but it was not very good as sentences can differ in a small number of characters or words but have the same meaning or can be totally different words but have exactly the same meaning.

AI Solution: We tried to use word embeddings to filter sentences as and remove duplicates. as similar words in meaning will have high score “small distance between word embeddings vectors”. We used the Roberta Language model to decide if two sentences are equivalent or not by giving it two sentences and output -1 to 1 score for equivalence. we then check if score > 0.9 that means two sentences has same meaning then finally we remove shorter sentences to keep as much as possible information in findings in report.

3.3.4.2. Model Hallucination

Model hallucination happens when the language model generates output that is nonsensical, irrelevant, or factually incorrect. because of producing text that deviates from the input context, resulting in meaningless sentences or random sequences of characters. For instance, while generating a medical report, the model might output medically irrelevant information, random jargon, or strings of unrelated words and symbols. such sentences must be removed from the final report.

We fixed this problem by making Regex that searches for non meaning words or patterns of repeated characters or longer words than longest english words. then we remove this sentence from the model.

ex: **`r"(.)\1{2,}" # match words have repeated chars more than 3`**

3.3.5. Report Generation

During the prediction phase, the language model generates the report by sequentially predicting the next word based on the visual features and previously generated words. The process begins with the encoded visual features, which provide the context for the report. The model utilizes the word embeddings and positional encodings to maintain an understanding of the medical terminology and the structure of the report. As each word is generated, it is fed back into the model to predict the next word, ensuring coherence and relevance throughout the report. The masked multi-head self-attention mechanism allows the model to focus on different parts of the input sequence, capturing intricate relationships between the visual features and the generated text. This iterative process continues until a complete, detailed, and accurate medical report is produced, describing the findings in the X-ray, including any abnormalities and relevant medical information. The use of softmax ensures that the most probable words are selected at each step, resulting in a human-like, comprehensive report.

To enhance the quality of the generated medical report, we used different decoding strategies, such as greedy search and beam search. Greedy search selects the most probable word at each step, aiming for immediate local optimization, which can lead to less coherent overall reports due to potentially missing better alternatives. Beam search, on the other hand, explores multiple possible word sequences simultaneously by keeping track of the top-k most likely sequences at each step. This method allows the model to consider a broader context and make more informed decisions, resulting in more coherent and accurate reports. Beam search is particularly beneficial for medical report generation as it helps in capturing complex relationships and dependencies within the visual features and the medical terminology, ensuring that the final report is both comprehensive and precise.so we used beam search with k = 8 which was quite enough to generate complex sentence correctly and does not hurt much running time of prediction.

3.4. Backend

3.4.1. Functional Description

In this section we present the backend System OvervieThe backend system is designed to manage the complete workflow of medical image reporting, incorporating a REST API, an AI server, and a database. This integrated system ensures seamless communication between various components, facilitating efficient data processing, report generation, and data storage.

REST API:

- Endpoint Management: The REST API provides various endpoints for uploading medical images, retrieving generated reports, and managing user authentication.
- Request Handling: It handles incoming requests, validates the data, and directs them to the appropriate services, such as the AI server or the database.
- Response Generation: After processing, the REST API sends responses back to the client, including the status of the request and any requested data.

AI Server:

- Report Generation: The AI server is responsible for processing the uploaded medical images using object detection and classification models to extract visual features. These features are forwarded to the language model to generate the actual report using a beam search algorithm.
- Heatmap Generation: The AI server is responsible for processing the uploaded medical images using a heatmap classifier, generating predictions of possible diseases, and heatmap for each positive disease to determine the location of the disease.

DataBase:

- Data Storage: We built a database system for medical application that serves doctors , employees, studies, and results, where we store results of ai models and written reports , tracking activity of doctor and new studies added.
- Query Management: It supports complex queries to retrieve specific reports, study data, or ai results. This facilitates efficient data management and quick access to necessary information.

4.4.2. Modular Decomposition

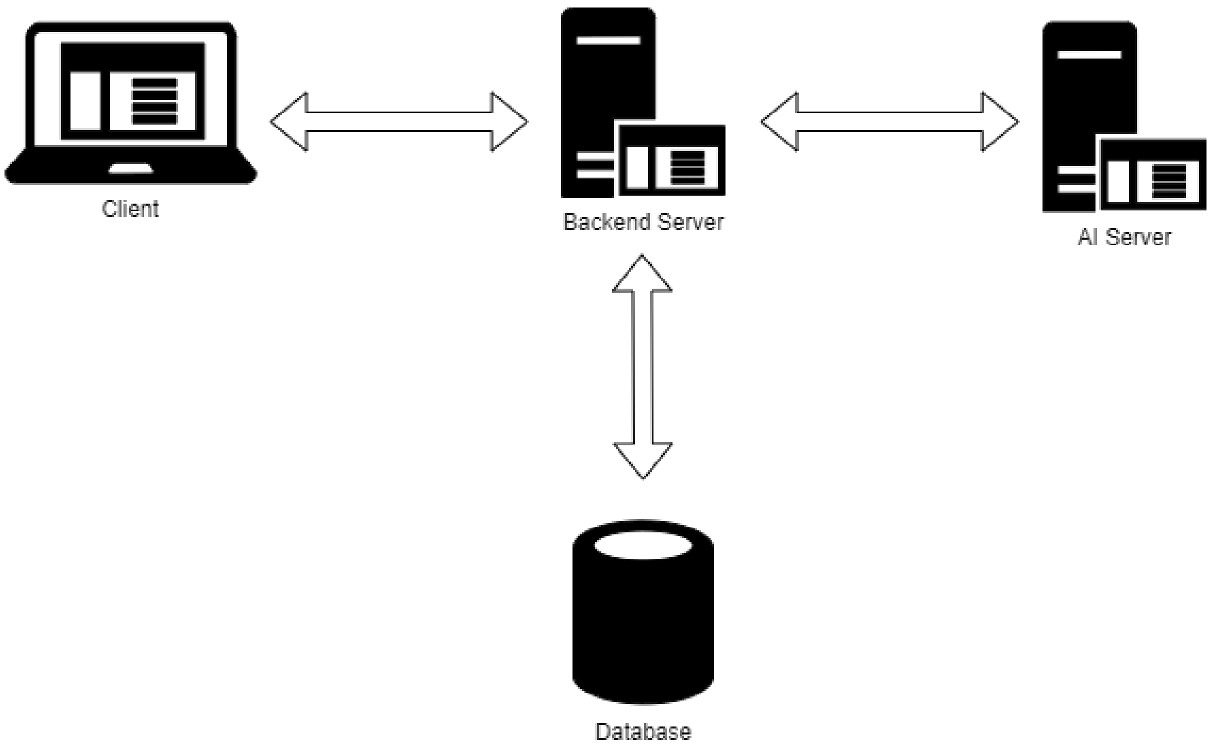

Figure 7 System Design

Our system is designed with three main components: the backend server, the AI server, and the database server. Here's an overview of how the system works:

1. Backend Server: The backend server acts as an intermediary, handling communication between the client and the other servers. It uses REST APIs to process client requests, perform CRUD operations, execute complex queries, and manage AI model execution.
2. Client Communication: Clients interact with the backend server through REST API requests. These requests can involve creating, reading, updating, or deleting data (CRUD operations), as well as executing more complex queries and invoking AI models.
3. Database Server: The backend server communicates with the database server to perform various operations such as inserting, deleting, updating, and retrieving data. The database server handles all relations in the database, ensuring data integrity and efficient query execution.
4. AI Server: Medical images are sent from the backend server to the AI server via REST APIs. The AI server processes these images to generate diagnostic reports, labels, and corresponding heatmaps. The results from the AI server are then stored back in the database.
5. Response to Client: Finally, the backend server compiles the results generated by the AI server and the database queries, and sends a comprehensive response back to the client. This response includes AI-generated results such

as diagnostic reports and heatmaps.

This system design ensures efficient and accurate processing of client requests, seamless interaction between the different servers, and the generation of valuable medical insights through AI.

3.4.3. Design Constraints

3.4.3.1. AI Server

We built two inference classes for our two full models x-reporto and heatmap, then we built our ai server with fastapi.

we mainly have 3 endpoints:

1. x-reporto/report/: returns bounding boxes - boxes sentences - full report
2. x-reporto/denoise/: returns denoised image of GANs model
3. heatmap/generate/: returns prediction of disease - heatmaps - template base report

3.4.3.2. Database Design

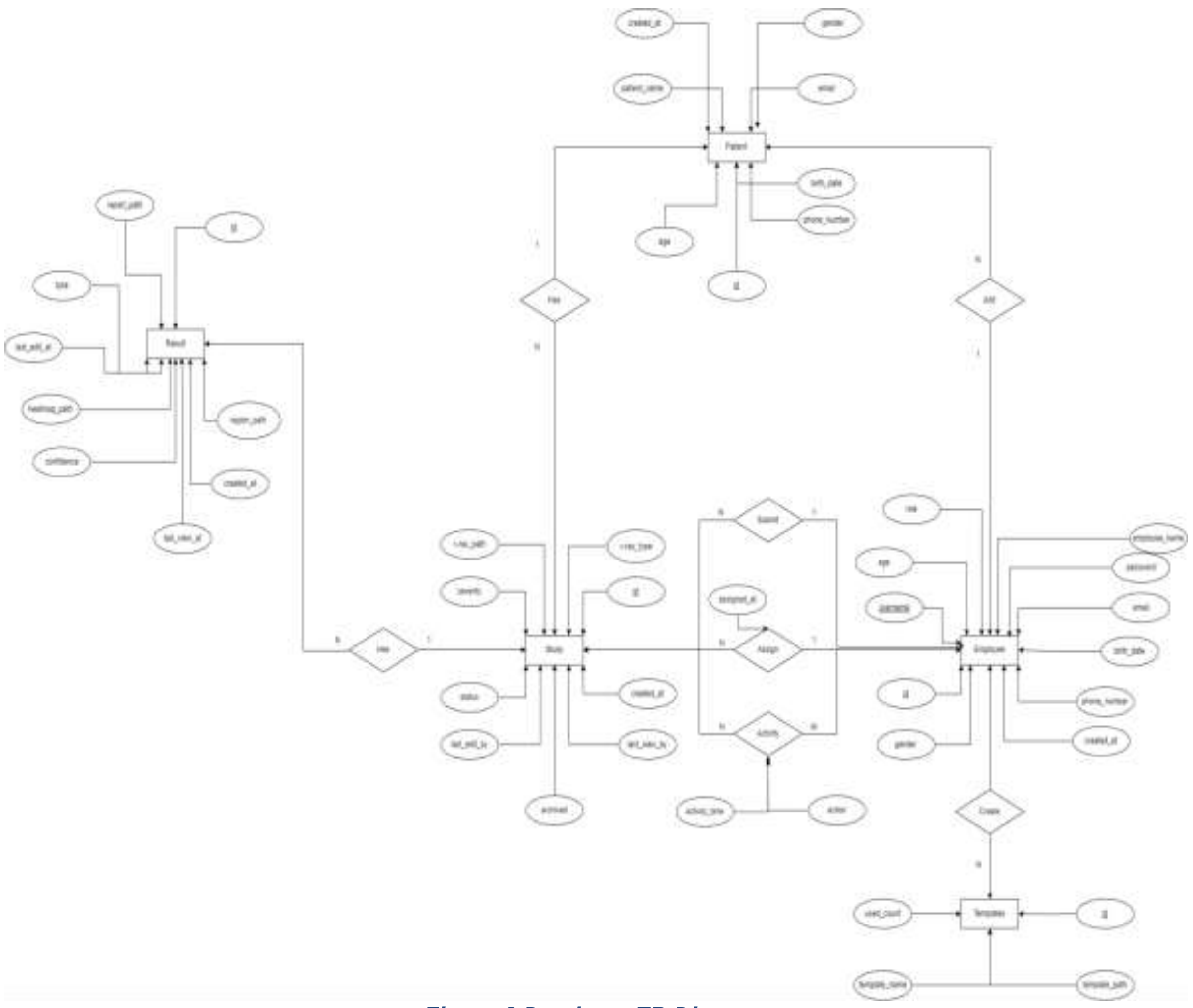

Figure 8 Database ER Diagram

- ## Entities

Patient Entity: The Patient entity represents individuals who undergo medical examinations and studies. This table includes attributes such as patient_id, name, birth_date, gender, contact_information, and other relevant demographic information. The patient_id serves as the primary key, ensuring each patient is uniquely identifiable. This entity is essential for maintaining patient records and linking them to their respective medical studies and results.

Employee Entity: The Employee entity represents the staff members within the medical facility, including doctors, nurses, and administrative personnel. Key attributes in this table include employee_id, name, role, department, contact_information, and other professional details. The employee_id is the primary key, ensuring unique identification of each employee. This entity is crucial for tracking which employees create patient records, conduct studies, and manage report templates. Additionally, employees with the role of "doctor" are responsible for creating and managing templates, as well as being assigned to specific studies.

Study Entity: The Study entity represents the medical studies or examinations conducted on patients. Attributes include study_id, patient_id (foreign key referencing the Patient entity), employee_id (foreign key referencing the Employee entity), doctor_id (foreign key referencing the Employee entity), severity, and notes. The study_id is the primary key, uniquely identifying each study. This entity is vital for recording and managing the various medical examinations each patient undergoes, and it establishes a direct relationship between patients and the studies conducted on them. Additionally, an employee (doctor) is assigned to each study, indicating the responsible medical professional.

Result Entity: The Result entity contains the outcomes or findings from each medical study. Attributes in this table include result_id, study_id (foreign key referencing the Study entity), type, date, and paths (such as images or documents). The result_id is the primary key, ensuring each result is uniquely identifiable. This entity is essential for documenting the detailed findings of each study, allowing for thorough analysis and review by medical professionals. The type is identifying what is the result is generative ai, template based, or custom based by doctor. The direct linkage to the Study entity ensures that each result is associated with a specific study.

Template Entity: The Template entity is used for creating standardized report formats that doctors can use when generating study results. Attributes include template_id, employee_id (foreign key referencing the Employee entity), template_name, content, and date_created. The template_id serves as the primary key, ensuring unique identification of each template. This entity allows doctors to create and manage templates, facilitating consistency and efficiency in report generation. Each template is linked to the doctor who created it, promoting accountability and ease of access.

● Relationships

Employee Creates Patients: This relationship indicates that employees, such as administrative staff or medical personnel, are responsible for creating and managing patient records. This ensures that patient data is accurately recorded and maintained.

Employee Creates Studies: This relationship highlights that employees initiate and document medical studies for patients. It underscores the role of employees in managing the entire process of patient examination.

Patient Has Studies: This relationship establishes that each patient can have multiple studies associated with them. It is fundamental for tracking the medical history and examinations of each patient.

Study Has Results: This relationship links each study to its corresponding results, ensuring that the findings from each examination are properly recorded and associated with the correct study.

Employee (Doctor) Creates and Has Templates: This relationship signifies that doctors create and manage report templates, which are used to standardize the reporting process for study results.

Employee (Doctor) Assigned to Study: This relationship indicates that each study is assigned to a specific doctor, who is responsible for conducting the study and generating the report. This ensures accountability and clarity in the assignment of medical examinations.

Chapter 6: System Testing and Verification

In this chapter, we outline the comprehensive approach taken to ensure that the outcomes of the implemented modules are realized accurately and reliably. Testing and verification are critical components of the development process, and we implemented a structured methodology to validate each module and the overall system. By utilizing a combination of unit testing and backend system testing,

6.1. Testing Setup

The testing setup for the project involves a dedicated environment that mirrors the production configuration to ensure reliable results. The system is deployed on a local server equipped with the necessary hardware and software requirements, including high-performance GPUs for model inference and a robust database management system for data storage. The testing environment utilizes Python and relevant libraries such as TensorFlow and PyTorch for model implementation, along with a REST API framework to facilitate interaction between modules.

6.2. Testing Plan and Strategy

The testing plan comprises multiple phases, starting with unit testing of individual ai modules followed by integration of a full pipeline of ai modules and finally integrated system testing to assess overall functionality. Each module is evaluated against predefined metrics to ensure it meets the specified performance criteria.

6.2.1. Module Testing

6.2.1.1 Custom Language Model

To test Language model prediction we have several metrics that help identify performance of our language model to capture medical information and report we used 5 metrics

- BLEU-1
- BLEU-2
- BLEU-3
- BLEU-4
- ROUGE-L

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L
Language model	0.213	0.133	0.10	0.08	0.33

Table 1 Language Model Validation results

Chapter 7: Conclusions and Future Work

7.1. Faced Challenges

7.1.1 Custom Language model

How to deal with visual features generated by object detectors:
We designed our visual Encoder model that applies transformation from image space features to text space features.
How to integrate visual features information in our language model:
We designed our custom masked self-attention block that adds visual features keys along with input to generate output scores.
How to train LLM with limited resources:

- Apply gradient checkpointing mechanism
- Apply gradient accumulating while training.

How to deal with redundant sentences:
Apply sentence similarity using pre pre-trained language model to remove very high similar sentences.
How to deal with hallucination:
We designed a regex that searches for non english words or repeated characters or repeated continuous words, then removes these sentences.

7.2. Gained Experience

7.2.1 Language Model

- Learned Transformer Architecture
- How to train Language Model
- how to customize language model upon your problem
- How to optimize model resources and chose best parameters
- How to generate sentences using trained model

7.3. Conclusions

In conclusion, X-Reporto represents a significant advancement in the field of medical report generation by integrating cutting-edge AI technologies such as transformers and GPT-2. The system not only addresses the challenges of managing and interpreting a high volume of chest x-rays but also enhances the quality of diagnostic reports through automation and improved image clarity. The motivation behind X-Reporto is driven by the need to support radiologists in delivering timely and accurate diagnoses, ultimately leading to better patient outcomes. By leveraging advanced language models and sophisticated backend infrastructure, X-Reporto stands as a testament to the transformative potential of AI in healthcare, paving the way for future innovations in medical imaging and diagnostics.

7.4. Future Work

7.4.1 Language Model

- Try to use bigger model sizes in order to generate more complex sentences and reports.
- Improve Visual Encoder Model to enhance transformation from image space to text space.

References

[1] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019):

[2] Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. InProceedings of the IEEE conference on computer vision and pattern recognition 2017 (pp. 2097-2106).