



August 11, 2024

University of Sciences and Arts in Lebanon

Faculty of Sciences and Arts

X-Ray Report Generation Using Transformer-Based Models

Computer Science Final Year Project – Data Science

Aya Haydous- 1121118

Mariam Srour- 1121115

Hiba Rezek-1121193

Supervisor: Dr Ali Ezzedine

t

Abstract

In our final project, we present a novel approach for generating comprehensive reports based on image inputs. Traditional methods for image analysis often rely on either textual descriptions or visual features alone, limiting their ability to capture nuanced relationships between visual and semantic content.

Our proposed system leverages the power of transformer architectures to bridge this gap, enabling the generation of detailed reports that encapsulate both the visual and semantic aspects of an image.

This report presents a detailed exploration and implementation of an image captioning model, focusing on fine-tuning pre-trained models on a custom dataset to generate descriptive captions for medical images, specifically chest and bone X-rays. The methodology includes data preprocessing, model training, and evaluation, with the integration of advanced techniques to enhance model performance.

Throughout the project, we faced the challenge of searching, browsing, reading, and selecting appropriate models. This iterative process involved trying multiple models and approaches until we found one that met our expectations. The evaluation metrics used, such as BLEU, METEOR, ROUGE, and CIDEr, helped assess the quality of the generated captions. The results indicate that while the model performs adequately at capturing individual words and short phrases, it struggles with generating longer, coherent sequences, as reflected in the relatively low BLEU-4 and CIDEr scores. These findings underscore the challenges in generating accurate and contextually rich captions for complex medical images.

To address these challenges, we explored various strategies including hyperparameter tuning and architectural enhancements. The final model, optimized through extensive experimentation, shows promise but highlights the need for further refinement, particularly in improving semantic coherence and relevance in generated captions.

The report concludes with a discussion on the deployment of the model in a .NET Core Web API for real-time inference, enabling its application in clinical settings where rapid and accurate image captioning is critical. This project not only advances the field of automated image captioning but also provides a framework for integrating AI models into real-world healthcare environments.

Table of Contents

Abstract	2
Table of Contents	3
1 Introduction	6
1.1 Background	6
1.2 Objective	6
1.3 Scope	7
1.4 Methodology	7
1.5 Limitations	10
2 Related Research	11
2.1 Similar Existing Products	11
2.2 Project's Added Value	11
3 Technical Discussion	12
3.1 Model Paradigm	12
1. Environment Setup	13
○ 1. PyTorch	14
○ 2. Pillow	14
○ 3. Flask and Flask-Ngrok	14
○ 4. ROUGE	15
○ 5. Torchvision	15
○ 6. Albumentations	15
○ 7. Pandas	15
○ 8. Spacy	15
○ 9. NLTK (Natural Language Toolkit)	16
○ 10. Scikit-learn	16
○ 11. Torch	16
2. Code Structure and Development	16
3. Model Training and Fine-Tuning	17
4. Evaluation and Inference	18
5. Ready-to-Run Code	18
3.2 Data	18
Explanation of the Folder Structure	18

1. Dataset Composition	19
3. Organizing the Images and Reports	20
4. Translation of Reports	20
The first Approach:	22
The Second Approach:	25
The Third Approach:	26
3.3 Evaluation	26
Comparison Based on Tests Made for Both Models	29
Hyperparameter Fine-tuning:	30
Potential Shortcomings and Mitigation:	33
3.4 Baseline Model	33
3.5 Advanced Model	35
4 User Interface	38
Backend Implementation	52
1. User Model.....	52
2. Role Model.....	53
3. Permission Model	54
4. RolePermission Model.....	54
5. Holiday Model	55
6. Appointment Model	55
7. XRayImage Model	56
8. Report Model.....	57
Backend Architecture	58
1. Domain Layer.....	58
2. Infrastructure Layer	58
3. Services Layer	59
4. UnitOfWork Layer	59
5. XRayReportApi Layer	59
5 Challenges	59
6 Future Work	60
7 Conclusion	61
8 Bibliography	61

1 Introduction

1.1 Background

In recent years, there has been a growing interest in multimodal learning, which aims to develop models capable of understanding and generating content across different modalities such as text, images, and audio. One significant challenge within this field is the effective integration of visual and textual information, particularly in tasks like image captioning and report generation.

Traditional methods for image analysis often rely on either purely visual features or textual descriptions, which may not fully capture the rich semantic relationships between visual content and natural language. This limitation hampers the ability of systems to generate comprehensive and contextually relevant reports based on image inputs.

To address this challenge, recent advancements in transformer-based architectures, coupled with multimodal pre-training techniques, have shown promising results in bridging the semantic gap between images and text. By leveraging the joint understanding of both modalities, these approaches can generate more informative and contextually relevant descriptions of images.

However, while significant progress has been made in this area, there is still a need for further research to explore the full potential of transformer-based architectures in multimodal understanding and report generation. This project seeks to contribute to this growing body of research by exploring and experimenting with multiple models to develop a system capable of generating comprehensive reports based on image inputs. Through a process of trial and error, we experimented with various models and approaches, refining our methodology until we met our expectations in terms of performance and accuracy.

1.2 Objective

- **Develop a System for Image-Based Report Generation:**
 - Develop a system to generate comprehensive reports based on image inputs
 - Bridge the semantic gap between visual content and natural language, enabling the generation of contextually relevant descriptions.
- **Make Report Generation Faster and More Accurate:**
 - Improve how quickly and accurately reports can be made compared to doing it by hand (traditional methods).
 - Use automation to reduce mistakes and make sure reports are consistent.
 - Lower the chances of missing important details in reports, improving their quality and reliability.
- **Build a Simple Website to Use the Model:**
 - Create a small user-friendly website where users can upload images and get reports generated by the model.
 - Added some features as booking an appointment from a doctor.
- **Fine-Tune and Compare Different Models:**

- Experiment with fine-tuning different models using the same data to compare their performance and choose the one with the most effective report generation

1.3 Scope

Included:

Dataset Collection: The project includes collecting and preprocessing real data from two renowned Lebanese hospitals: Bahman Hospital and Al Zahraa Hospital University Medical Center.

Script writing and Preprocessing: a script is written to change the format of images from dicom to other format (bitmap/jpeg) while cropping the images to exclude the patient's meta data and information. Data preprocessing was performed on both images and reports.

Transformer Architecture Fine-tuning: The project encompasses the fine-tuning of transformer-based architecture for multimodal report generation on a custom dataset. This includes the adaptation of transformer models to process both visual and textual inputs efficiently.

Hyperparameter Finetuning: Multiple hyper parameters were changed and trained on. The validation dataset was used to choose the best hyper parameters.

Model Monitoring: MLOps concepts and frameworks were utilized to monitor the training and validation processes against different metrics.

Evaluation and Comparison: The project includes conducting a thorough evaluation and comparison of the developed system against baseline labels. This involves assessing the quality, accuracy, and contextual relevance of the generated reports across different datasets and scenarios using the corresponding evaluation metrics.

User Interface Development: The project encompasses the development of a user interface for interacting with the system. The focus is on the front-end and back-end implementation of the report generation system and its core functionalities.

Excluded:

Extensive Model Training: While the project includes fine-tuning the chosen models, extensive training of underlying transformer architectures from scratch is excluded. The emphasis is on leveraging pre-trained models and adapting them to the specific task of report generation.

Semantic Image Understanding: While the project aims to bridge the semantic gap between images and text, in-depth semantic image understanding tasks such as object detection, segmentation, or scene understanding are excluded. The focus is on generating reports based on the overall semantic content of the images rather than detailed object-level analysis.

1.4 Methodology

Approach:

Our approach involves leveraging transformer architecture alongside several advanced models, including CLIP-GPT2, BioViL-T, NLPConnect/vit-gpt2-image-captioning, Microsoft/Florence-2-base-

ft, and DenseNet121, to generate comprehensive reports from image inputs. These models collectively enable the system to process and understand the visual content of medical images and translate that understanding into detailed and contextually relevant medical reports. By combining these powerful models, we aim to improve the accuracy and relevance of the generated reports, addressing the specific challenges associated with medical image captioning and report generation.

Techniques:

We employ techniques such as fine-tuning the CLIP-GPT2, BioViL-T, DenseNet121-based models, GPT-2, Florence 2, BLIP and Kosmos-2 models on a custom dataset of X-ray images and reports, ensuring the integration of visual and textual information.

Data Collection and Preprocessing:

1. Data Collection:

The data collection phase was the longest phase, lasting approximately for 5 months, March to July. The data was collected from two renowned Lebanese hospitals: Bahman Hospital and Al Zahraa Hospital University Medical Center, where each image is retrieved from its respective database which is PACS¹. Then each image is matched with its corresponding report from a separate reporting system.

2. Data Preprocessing:

For Bahman Hospital dataset, a custom script is developed to convert DICOM images to a more accessible format (e.g., bitmap or JPEG) while cropping out any sensitive patient information. This ensures the privacy and confidentiality of patient data while preparing the images for further analysis.

For Al Zahraa Hospital University Medical Center, the images we exported individually as JPEG images from the ULTIMA PACS without the patient's information or the picture's metadata. The folders' names were the patients' names originally, that was changed later to the accession number which belongs to each patient individually, ensuring both privacy and uniformity among the sample. The reports were extracted from the Hospital Information System individually based on the image's date and time. A notebook was prepared to preprocess both reports and images that includes normalizing and cleaning the images and texts before feeding the data to the model. Most reports were in the French language, so this was also handled by translating the texts into English via Google's deep translator library. The notebook prepares a dataset of X-ray reports for further analysis. The steps include installing necessary libraries, mounting Google Drive, loading the dataset, translating non-English reports to English, preprocessing the text data, and splitting the dataset into

¹ Picture Archiving and Communication System (PACS), is a medical imaging technology which provides economical storage and convenient access to images from multiple modalities.

training, validation, and testing subsets. Finally, the processed datasets are saved to Google Drive. The link to the reports' preprocessing notebook is [here](#).

Model Selection and Pre-training:

Multiple models were combined and finetuned for this project, including:

1. **CLIP-GPT2 Models:** The strengths of CLIP (Contrastive Language–Image Pre-training) in extracting rich image features and GPT-2 (Generative Pre-trained Transformer 2) in generating text descriptions based on these features helped to understand images in the context of natural language.
2. **Kosmos-2 Model:** Employ the Kosmos-2 model, another multimodal model capable of interpreting images and generating text descriptions.
3. **DenseNet121-based Models:** These models were employed for their strong performance in medical image analysis, particularly in feature extraction from X-ray images. DenseNet121's ability to retain features across layers made it a valuable component in understanding complex medical images.
4. **GPT-2 (Generative Pre-trained Transformer 2):** GPT-2 was utilized for its capability in generating coherent and contextually relevant text descriptions based on the features extracted from images. Its integration with other models allowed for the generation of detailed and informative medical reports.
5. **Florence 2 Model:** Florence 2, a sophisticated multimodal model, was selected for its ability to interpret both images and text, contributing to the generation of comprehensive and contextually accurate descriptions.
6. **BioViL-T Model:** BioViL-T is a vision-language model specialized in chest X-ray and radiology report analysis. It uses temporal multi-modal pre-training to better integrate image and text data, enhancing tasks like image/text classification and language decoding.
7. **BLIP Model:** BLIP (Bootstrapping Language-Image Pretraining) was incorporated for its capabilities in image captioning and understanding. BLIP's architecture enables the generation of descriptive text from images, making it valuable for producing detailed and accurate medical reports based on X-ray images.

Fine-tuning Transformer Architecture:

1. **Transformer Architecture:** We employed a transformer-based architecture, which facilitates the efficient processing of both visual and textual inputs.
2. **Fine-tuning:** The fine-tuning process involved adapting several pre-trained models, including DenseNet121-based models, GPT-2, Florence 2, CLIP-GPT2, BioViL-T, BLIP and Kosmos-2, to the specific task of generating comprehensive reports from medical images. This fine-tuning was conducted on the collected datasets from Bahman Hospital and Al Zahraa Hospital University Medical Center. The training process focused on enabling the models to learn the nuanced relationships between images and their corresponding reports, optimizing their performance in this specialized medical context.

User Interface Development:

- **Front-end Development:** We design and implement a user-friendly interface for interacting with the system. This involves developing intuitive features for uploading images, initiating report generation, and displaying the generated reports.
- **Back-end Development:** The back-end of the interface is responsible for processing user inputs, invoking the report generation system, and presenting the generated reports to the user.

Tools and Technologies:

- **Programming Languages:** Python is primarily used for development due to its extensive libraries for machine learning and image processing.
- **Frameworks:** We leverage popular deep learning frameworks such as PyTorch for implementing the transformer architecture and fine-tuning various models, including DenseNet121-based models, CLIP, GPT-2, BioViL-T, Kosmos-2, BLIP and Florence 2.
- **Data Processing:** Executable running script and notebooks using python.
- **User Interface:** Web development frameworks like .NET core and angular are utilized for building the user interface.

Experimental Setup:

Initially, we utilized local computing resources and Google Colab's T4 GPU for model development and training. Our local setup involved training models on a personal computer, which, although slower and prone to overheating during prolonged sessions, provided a necessary fallback when Colab's resources were insufficient. Since the training process didn't demand higher GPU capabilities, we did not employ additional cloud services for enhanced computational power. This combination of local and Colab resources allowed us to balance accessibility with the computational demands of the project.

1.5 Limitations

The team faced several limitations throughout the project. First, the novel nature of our approach presented significant challenges, as the idea is still in the research phase globally. This made it particularly challenging to find established methodologies, models, and best practices. As a result, building the project workflow pipeline was an exhaustive process that required extensive research and experimentation.

Second, data availability was a major limitation. The highly confidential nature of the medical data we needed made it difficult to obtain, especially in Lebanon, where the practice of requesting large datasets from hospitals for AI tasks is not yet mature. Despite our efforts to collect data from Al Zahraa and Bahman hospitals, we faced significant delays and constraints, ultimately limiting the amount of data we could use for training.

Third, time constraints further compounded the difficulties we faced. The delays in data acquisition and the challenges in accessing computational resources limited our ability to conduct multiple rounds of

hyperparameter tuning or engage in a more intensive development phase. These limitations highlighted the need for greater flexibility and resource availability in future projects.

2 Related Research

2.1 Similar Existing Products

In the landscape of image-to-text projects, existing endeavors predominantly focus on image captioning, aiming to generate descriptive captions for various types of images. However, there is a noticeable dearth of projects specifically addressing the detection of bone problems from X-rays and reporting through this rough this methodology. Despite this gap, several relevant studies and solutions provide valuable insights into the domain of medical image analysis and report generation.

Image Captioning Projects: Numerous image captioning projects leverage deep learning techniques to generate succinct textual descriptions of image content. Notable examples include "Show and Tell" by Vinyals et al. (2015) and "Show, Attend and Tell" by Xu et al. (2015). These projects typically focus on generating natural language descriptions that capture the salient features and objects depicted in images. While effective for general image understanding, they lack the specialized domain knowledge required for medical image analysis and report generation.

Medical Image Analysis Solutions: In the domain of medical imaging, there are solutions aimed at tasks such as image segmentation, classification, and disease diagnosis. For instance, convolutional neural networks (CNNs) have been widely employed for tasks like tumor detection in radiology images. However, these solutions often focus on binary classification tasks rather than comprehensive report generation.

2.2 Project's Added Value

Our project offers several distinct advantages and contributions compared to existing solutions:

1. **Tailored Medical Reporting:** Unlike traditional image captioning approaches, which provide generic descriptions of visual content, our system focuses specifically on extracting detailed medical information from X-ray images. This ensures reports are both descriptive and useful for doctors, meeting their specialized needs.
2. **Clinical Relevance:** Using a large set of annotated medical reports, our system generates reports that are both accurate and relevant, capturing detailed diagnostic information to aid in healthcare decisions.
3. **Automated Report Generation:** By automating the extraction of information from X-rays, our project makes the report generation process quicker and less burdensome for healthcare professionals, speeding up diagnosis and treatment.
4. **Scalability and Adaptability:** Through the use of deep learning techniques and data-driven approaches, our system is scalable and adaptable to various medical scenarios and

imaging modalities. This scalability ensures that the system can handle diverse datasets and accommodate evolving healthcare needs over time.

5. **Advancement in Medical Imaging:** By bridging the gap between image analysis and medical diagnosis, our project contributes to the advancement of medical imaging technologies. The ability to accurately detect and report issues from X-ray images has implications for improving patient outcomes and enhancing the quality of healthcare delivery.

3 Technical Discussion

3.1 Model Paradigm

We're addressing the task of generating comprehensive medical reports from X-ray images, a problem that falls under the domain of multimodal learning. This approach integrates both computer vision (CV) and natural language processing (NLP) to effectively bridge the semantic gap between visual content and textual descriptions in medical imaging.

Our model paradigm leverages deep learning techniques, particularly transformer-based architectures, which have shown success in multimodal learning tasks. Transformers excel in processing sequential data, making them well-suited for tasks involving both images and text. By fine-tuning pre-trained transformer models on our custom dataset of X-ray images and medical reports, we enable the model to understand the complex relationships between visual content and natural language descriptions.

This model paradigm allows us to accurately analyze and describe diagnostic findings in X-ray images, providing valuable insights for healthcare professionals in clinical decision-making. By combining insights from computer vision and natural language processing, our model effectively addresses the challenges inherent in generating comprehensive medical reports from X-ray images. The task at hand was to develop an image captioning model, with a specific focus on generating captions for chest X-ray images. We opted for using three approaches to cover this project.

The first approach:

The project aimed to develop an image captioning model for generating descriptions of chest X-ray images by combining CLIP (Contrastive Language–Image Pre-training) for image feature extraction and GPT-2 (Generative Pre-trained Transformer 2) for text generation. This integration leverages CLIP's capability to interpret medical visual data and GPT-2's strength in generating coherent, contextually relevant text. The resulting model bridges the gap between image and text, producing medically accurate and contextually aligned reports, crucial for effective diagnosis.

CLIP was used to extract rich visual features from chest X-ray images. CLIP, a model trained on a vast dataset of image-text pairs. CLIP is known for its ability to connect visual content with textual descriptions effectively. The extracted image features encapsulate the medical details from the X-ray images, which are crucial for generating accurate reports.

After obtaining the image features from CLIP, these features were integrated into the text generation process using GPT-2. GPT-2, known for generating coherent and contextually relevant text, was fine-tuned to generate medical reports that align with the specific context provided by the X-ray image features. The image features extracted from CLIP were summarized and incorporated into the prompt provided to GPT-2. This prompt served as the basis for GPT-2 to generate the text, ensuring that the generated reports were contextually aligned with the visual information from the X-ray images.

During inference, the approach implemented mechanisms to handle potential out-of-vocabulary (OOV) tokens, which could arise due to the unique and domain-specific vocabulary used in medical reports.

Here's the link to the [notebook](#) that handles the training, testing and inference of the CLIP-GPT2 models.

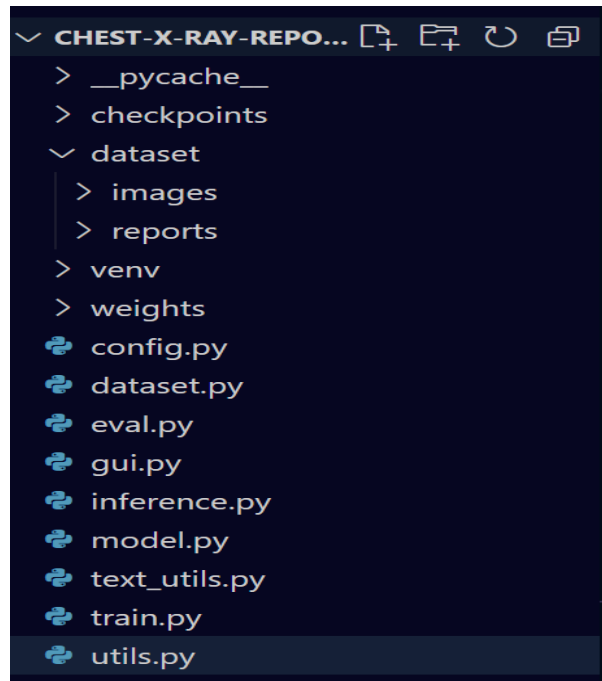
The second approach:

This second approach focused on developing a robust and well-structured environment for training a deep learning model that generates medical reports from chest X-ray images. By combining the strengths of a CNN-based encoder and an RNN-based decoder with attention, the model effectively bridges the gap between visual content and textual descriptions, providing valuable insights for healthcare professionals. The environment setup, code development, and fine-tuning steps were meticulously carried out to ensure that the code is ready to run and capable of generating accurate and relevant medical reports.

1. Environment Setup

To begin the development process, the environment was carefully set up to ensure compatibility with the required libraries and frameworks. This involved:

- **Python Environment:** A Python virtual environment (venv) was created to manage dependencies and prevent conflicts with other projects. This environment ensures that all necessary libraries are isolated and can be easily managed.



- **Library Installation: Library Installation**
- In the development of the chest X-ray report generation model, several essential libraries were installed to provide the necessary tools for building, training, evaluating, and deploying the model. Below is a breakdown of these libraries and their roles in the project:

- **1. PyTorch**

- **Installation Command:** `pip install torch`
- **Purpose:** PyTorch is the core deep learning library used in this project. It provides dynamic computational graphs, making it easier to build and train neural networks. PyTorch is used for defining the model architecture, performing forward and backward passes, and optimizing the model parameters during training.

- **2. Pillow**

- **Installation Command:** `pip install Pillow`
- **Purpose:** Pillow is a powerful image processing library in Python. It is used to load, manipulate, and save images. In this project, Pillow is utilized to handle X-ray images, converting them into a format that can be processed by the deep learning model.

- **3. Flask and Flask-Ngrok**

- **Installation Command:**
 - `pip install Flask`

- `pip install Flask-Ngrok`
 - **Purpose:** Flask is a lightweight web framework used to create a simple web application for interacting with the model. Flask-Ngrok allows the Flask application to be easily exposed over the internet using Ngrok, which is useful for testing and demonstrating the model's capabilities in a live environment.
- **4. ROUGE**
 - **Installation Command:** `pip install rouge`
 - **Purpose:** ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a set of metrics used to evaluate the quality of text generated by the model. It compares the generated text with reference text and measures the overlap of n-grams, providing insight into how well the generated report matches the actual medical report.
- **5. Torchvision**
 - **Installation Command:** `pip install torchvision`
 - **Purpose:** Torchvision is a PyTorch extension that provides datasets, model architectures, and image transformations that are widely used in computer vision tasks. In this project, it is used for pre-trained models like DenseNet121, which is fine-tuned for generating reports from X-ray images.
- **6. Albumentations**
 - **Installation Command:** `pip install albumentations`
 - **Purpose:** Albumentations is a fast image augmentation library. It is used to apply transformations to the X-ray images during training, such as resizing, normalization, and other augmentations that help the model generalize better to unseen data.
- **7. Pandas**
 - **Installation Command:** `pip install pandas`
 - **Purpose:** Pandas is a powerful data manipulation and analysis library. Although not heavily used in this project, Pandas can be useful for handling tabular data, such as loading and processing metadata associated with the X-ray images and reports.
- **8. Spacy**
 - **Installation Command:** `pip install spacy`

- **Purpose:** Spacy is an advanced natural language processing (NLP) library. In this project, Spacy is used for tokenizing the medical reports into words, which are then processed and fed into the model for text generation.

- **9. NLTK (Natural Language Toolkit)**

- **Installation Command:** `pip install nltk`
- **Purpose:** NLTK is a comprehensive library for working with human language data. It is used in this project for various text processing tasks, such as tokenization, stemming, and evaluating the model's generated text using metrics like BLEU and METEOR.

- **10. Scikit-learn**

- **Installation Command:** `pip install scikit-learn`
- **Purpose:** Scikit-learn is a machine learning library that provides simple and efficient tools for data mining and data analysis. In this project, it is used for splitting the dataset into training and testing sets, as well as for other utility functions like calculating evaluation metrics.

- **11. Toarch**

- **Installation Command:** (Assumed to be a typo of "torch," hence covered under PyTorch)
- **Purpose:** PyTorch is already covered as the primary deep learning framework. Any references to "Toarch" are assumed to be part of PyTorch, which is used for building and training the model.
- **Data Handling:** The dataset of X-ray images and their corresponding reports was organized into a structured format within the dataset directory. Images were stored in a subdirectory (`images`), and the corresponding reports were placed in a separate subdirectory (`reports`). This organization facilitates easy loading and preprocessing of data.

2. Code Structure and Development

With the environment set up, the project proceeded with the development of the model and its components. The codebase is structured to include several key modules, each responsible for different aspects of the model's functionality:

- **Dataset Management (`dataset.py`):**
 - The `XRayDataset` class was implemented to handle the loading, preprocessing, and augmentation of the X-ray images and reports. This class is responsible for

reading the image files, normalizing the textual data, and converting the text into numerical tokens that can be fed into the model.

- A vocabulary was built from the textual reports, enabling the model to convert words into indices and back. This vocabulary handles the mapping of words to unique identifiers and vice versa.
- **Model Architecture (model.py):**
 - The core of the model consists of a CNN-based encoder (DenseNet121) and an RNN-based decoder (LSTM with attention mechanism). The encoder processes the X-ray images, extracting meaningful features, while the decoder generates the textual report based on these features.
 - The model was fine-tuned using a pre-trained DenseNet121 network, which was adapted to focus on the specifics of chest X-ray images. The final layers of the network were modified to output features suitable for text generation.
- **Utility Functions (utils.py):**
 - Utility functions were developed to handle common tasks such as loading datasets, saving and loading model checkpoints, and splitting the dataset into training and testing sets. These functions streamline the training and evaluation processes, ensuring consistency and reproducibility.
- **Training Script (train.py):**
 - The train.py script was created to manage the training loop. This script loads the dataset, initializes the model, and iteratively trains the model by feeding batches of images and corresponding captions. The model's performance is monitored using a loss function, and the weights are updated accordingly.
 - Checkpoints were saved at the end of each epoch, allowing the training to be resumed from the last saved state if needed.

3. Model Training and Fine-Tuning

- **Training Process:**
 - The model was trained using a combination of the X-ray images and their corresponding reports. The CNN encoder extracts features from the images, which are then passed to the LSTM decoder to generate text.
 - The training process involved iterating over the dataset multiple times (epochs), adjusting the model's weights to minimize the loss between the generated text and the actual reports.
- **Fine-Tuning:**
 - Fine-tuning was a critical step in adapting the DenseNet121 model to the specific task of chest X-ray report generation. By freezing the lower layers of the pre-trained network and training the higher layers, the model was able to retain the general visual features learned from ImageNet while adapting to the nuances of medical images.

4. Evaluation and Inference

- **Evaluation (eval.py):**
 - After training, the model's performance was evaluated using various metrics like BLEU, ROUGE, and METEOR scores. These metrics assess how well the generated reports match the actual reports in terms of content and fluency.
- **Inference:**
 - The final model was tested by generating reports for new X-ray images. The inference process involves passing an image through the encoder to extract features, which are then used by the decoder to generate a textual report.

5. Ready-to-Run Code

Once all the components were developed and integrated, the code was ready to be executed. The environment was fully set up with all dependencies, and the model was trained and evaluated. Users can now run the code to generate medical reports from chest X-ray images, utilizing the trained model to assist in clinical decision-making.

3.2 Data

The data that the group will work with is real data, collected from Bahman Hospital and Al Zahraa Hospital University Medical Center. The dataset used for this project consisted of chest X-ray images, each accompanied by a corresponding medical report. These reports served as the ground truth captions for the image captioning task. The dataset is stored as a folder containing the images and a csv file containing the image names and their corresponding reports. Initially, all kinds of images (x rays, CTs, MRIs, Panoramic) are stored in a file system hierarchy of medical record number that contains another folder (case study) having a randomly generated number series, that contains another folder for the instances (frontal image...) which contains the actual x ray images.

```
/[Patient Name]_[Patient ID]/  
  /[Study Description]_[Study Date]/  
    /[Series Number]_[Series Description]/  
      [Instance Number]_[Image Type].dcm
```

Explanation of the Folder Structure

[Patient Name]_[Patient ID] Folder:

- This is the top-level folder named after the patient's name and ID. It contains all the studies associated with that patient.

[Study Description]_[Study Date] Folder:

- Each study (e.g., CT Chest, MRI Brain) for the patient is stored in a separate folder. The folder name typically includes the study description and the date it was performed.

[Series Number]_[Series Description] Folder:

- Within each study folder, the data is further organized into series, with each series representing a sequence of images or a specific type of data (e.g., axial slices, 3D reconstructions). The folder name includes the series number and a description of the series.

DICOM Files ([Instance Number]_[Image Type].dcm):

- Inside each series folder, the individual DICOM files are named according to the instance number and sometimes the image type or other relevant identifiers. These are the actual image files that contain the medical data. In our case, we exported the dicom images as jpg.

The needed images are of type x ray, so all other types will be filtered out. Since the data was collected from 2 sources, two different processes were followed.

For Bahman Hospital:

The dataset for this project was provided by the hospital's IT department and consisted of various types of bone X-rays from different body parts. This data was essential for training our model to generate medical reports based on X-ray images. The dataset included a total of 3,220 images, organized into groups, with each group corresponding to a single medical report. In total, there were 1,002 reports, with each report covering a specific group of related images.

1. Dataset Composition

- The dataset was provided in a CSV file with the following columns:
 - **Exam:** The type of examination or X-ray conducted.
 - **Date:** The date on which the X-ray was taken.
 - **CODE:** A unique identifier for each examination, used to match images with their corresponding reports.
 - **Result:** The textual report or findings associated with the X-ray images.
 - **Conclusion:** A summary or conclusion of the report.
 - **Age:** The age of the patient at the time of the examination.
- The "Result" column contained the detailed reports, while the "CODE" column was used to match the images to the reports.
 - **Original Format:** The X-ray images were initially in DICOM format, which is a standard for medical imaging that embeds patient information within the files. DICOM files require specialized software, known as a "Viewer," to access the images.
 - **Conversion to JPEG:** To enhance accessibility and streamline processing, we developed a script that converts DICOM images into JPEG format. JPEG is a widely

recognized image format that can be easily viewed and processed using standard image libraries.

- **Patient Privacy:** Since DICOM files contain sensitive patient information, it was crucial to protect patient privacy during the conversion process. To achieve this, our script included the following steps:
 - **Cropping by Ratio:** The script cropped the images using a predefined ratio to ensure the removal of any identifiable patient information. Initially, we explored cropping based on a black column present in the images, but this method proved ineffective due to the presence of text near the edges in many images. Instead, we implemented a fixed ratio for cropping, which successfully eliminated patient information while preserving the essential parts of the X-ray.
 - **Executable Script:** To make the process even more user-friendly, we compiled the script into an executable file. This allows the script to run on other computers without needing to download additional libraries, simplifying its use across various systems.

For a comprehensive overview of the entire process, including detailed explanations and visual aids, you can refer to our [Medium article](#) that covers everything from start to finish.

3. Organizing the Images and Reports

- **Folder Structure:** When we received the images, they were organized into folders, with each folder containing multiple images of the same patient taken on the same day. These images were associated with a single report.
- **Renaming Images:** To ensure that the images could be easily matched with their corresponding reports, we developed a script that renamed each image based on the folder name. The folder name corresponded to the "CODE" in the CSV file, and each image within the folder was appended with a letter (e.g., 1302441_A, 1302441_B, etc.). This systematic naming convention ensured that each image could be uniquely identified and linked to its corresponding report.
- **Creating Text Files for Reports:** Another script was written to create a separate text file for each report. These text files were named using the corresponding "CODE" from the CSV file, making it easy to match the reports with the images during the training of the model.

4. Translation of Reports

- **Language Variability:** During the preprocessing phase, we discovered that some of the reports in the "Result" column were written in French, while others were in English. Since the model was primarily trained on English text, it was essential to have all reports in a consistent language.
- **Translation Process:** To address this issue, we utilized GPT's translation capabilities to translate all French reports into English. This ensured that all textual data fed into the model was in English, providing consistency in training and improving the model's ability to generate accurate medical reports.

You can find the csv file of data here.

And the images folder here

For Al Zahraa Hospital University Medical Center:

The team manually collected the data from the hospital's IT department. It was a tiring process because each image and report had to be exported one by one. After more than 100 hours of work over a week, the team gathered around 1200 samples. The data in the hospital is stored separately, with the images kept on the hospital's servers. These images can only be viewed using the PACS system provided by the hospital, specifically the Ultima system.

The Ultima system stores the modalities (x rays, CT scans, MRI scans...) as records containing the patient's id, accession number, study id, date of imaging, patient's name, study description, physician's name, number of images per imaging. We chose the studies identified as CHEST and having the DX (Digital X ray) modality, ranging from the year 2022 to June 2024. We also chose the chest X rays containing 1 image in the study, as our model takes as an input 1 image. From each patient we took 1 study only, to ensure uniformity and fairness in the sample and to cover as much cases as possible.

The process of the data collection was as follows: first the image was viewed by the dicom viewer of the hospital, then it was exported without the meta data containing the patient's information. The exported file is named by the patient's name. This was changed to the accession number to ensure the complete anonymity of the samples. Then the hospital's information system was accessed to the reports section, an archive for all reports written, and using the accession number we could get a full pdf containing all reports written for this patient (containing different modalities). By identifying the report by the date and time, we could match the image to its corresponding report. Note that many exported images didn't have any reports as they were either pre-admission images (the patient does the image and leaves the hospital without creating any record in the hospital information system) or didn't have an existing report due to database collision or data entry fault. Of course, these samples were removed from the dataset.

After collecting the data, our team worked on separate computers, which resulted in multiple folders of images, each containing subfolders with images. The main folders were named according to the entries in our Excel files, and we had several Excel files as well. To streamline everything, we developed a series of scripts to organize and standardize the dataset.

The first script was designed to extract all the images from their respective subfolders, rename them based on the main folder's name, and consolidate them into a single folder. Another script was created to ensure that every report had a corresponding image. If a report lacked a matching image, the script identified the report so we could locate the missing image or delete the report as necessary. Conversely, the script also checked that each image had an associated report, and any image without a corresponding report was flagged for deletion.

Once we processed each folder, we needed to merge all the content into one unified folder. To prevent duplicate file names from different folders, we wrote a script that detected identical names and automatically renamed one of the conflicting files. This allowed us to combine everything without losing any data due to name clashes.

Additionally, the dataset included reports in both French and English. We used Translate GPT to translate the French reports into English, ensuring consistency across the dataset.

In the end, we compiled a well-organized dataset consisting of one Excel file and a single folder containing all the images in JPG format, ready for further analysis.

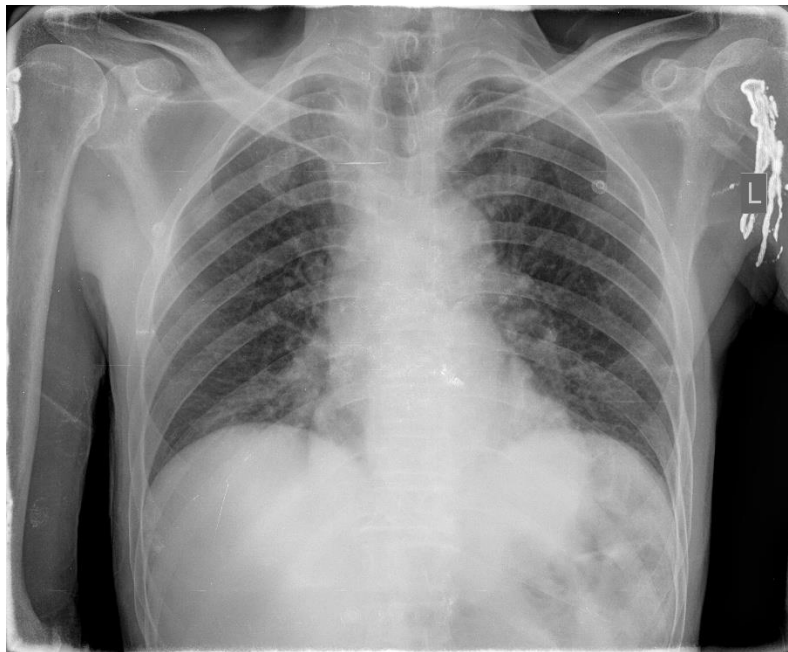
The csv file was filled with the patient's id, accession number, date of study, gender and report text.

Most of the reports were written in the French language so that was an extra preprocessing step.

The total size of the sample is 1218 image and report, to be split into training, development, and testing datasets.

A sample from the dataset before text preprocessing:

This is image: 22I12576.jpeg



A snapshot of the excel file of the 22I12576 record:

Patient ID	Accession Number	Study Date	gender	Reports
1.3.51.0.7.13031837681.17781.37192.46305.60232.49943.23492	22I12576	25/1/2023	m	canule trachéale en place. infiltrats intersticiels associés. coeur de taille normal.

You can find the csv file of the dataset [here](#)

And the folder of images [here](#)

Again, three approaches were applied to the collected data.

The first Approach:

The images were processed to a uniform size of [3, 224, 224] using standard preprocessing techniques, including resizing, center cropping, and normalization. The images were preprocessed by resizing them to 224x224 pixels and normalizing them to have values between 0 and 1. The dataset was divided into training, validation, and test sets, with careful consideration given to

maintaining a balanced distribution across the splits. The training set was used to fine-tune the models, the validation set was employed for hyperparameter tuning and model selection, and the test set was used for final evaluation. Here's the link to the [notebook](#) that handles this process for the approach.

In addition to the image data, the text data (medical reports) required preprocessing to ensure consistency and improve model performance. This involved translating the reports from French to English, cleaning the text by converting it to lowercase, removing punctuation, and stripping extraneous whitespace. Here's the [notebook](#) that handles this process. Google Drive was used as a storage unit, here's the link to the [dataset](#) on drive.

Here are some data samples after the image and text preprocessing:

Sample 1:

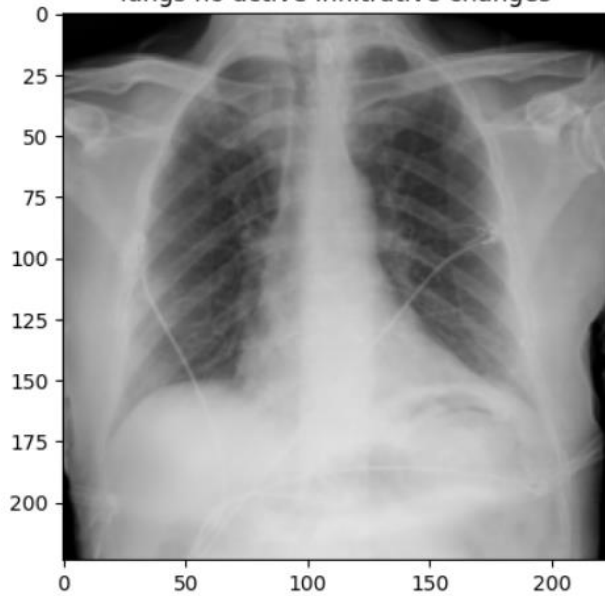
diffuse interstitial infiltrates of 2 pulmonary areas observable
free trachea no pleural reaction normal ict



```
normalized image shape torch.Size([3, 224, 224])  
image tensor shape torch.Size([3, 224, 224])
```

Sample 2:

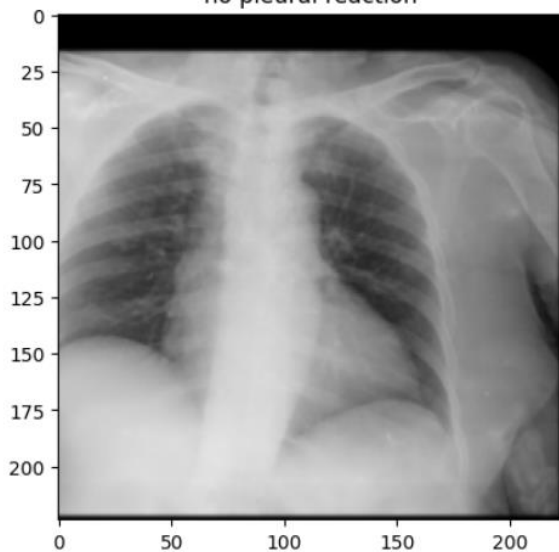
the heart is normal in size and shape
lungs no active infiltrative changes



normalized image shape torch.Size([3, 224, 224])
image tensor shape torch.Size([3, 224, 224])

Sample 3:

ict is at the limit of normal
unrolled aorta
bilateral interstitial infiltrates also observed
no consolidation
no pleural reaction



normalized image shape torch.Size([3, 224, 224])
image tensor shape torch.Size([3, 224, 224])

After the data is preprocessed, the training data was passed to the model for training.

The Second Approach:

In the second approach, the focus was on building a custom dataset loader and vocabulary system to better handle the complexities of medical image captions. This approach allowed for more granular control over the preprocessing of both images and text data.

Image Processing:

The images were processed to a uniform size of [3, 224, 224] similar to the first approach. The images were first converted to grayscale, and then each image was expanded into three channels to match the expected input format for the model. Standard preprocessing techniques, including resizing, center cropping, and normalization, were applied. This ensured that the images were consistent in size and format before being fed into the model.

Text Processing:

For text data, a custom Vocabulary class was created to tokenize, build a vocabulary, and numericalize the medical reports. The vocabulary was constructed by tokenizing the text data (medical reports) and filtering out words that appeared less frequently than a specified threshold. This process ensured that the most relevant words were retained while minimizing noise in the data.

The text data was preprocessed by converting it to lowercase, removing punctuation, and stripping extraneous whitespace. A significant addition in this approach was the use of the `normalize_text` function to standardize the text further. The reports were then translated into numerical sequences using the custom vocabulary. Special tokens such as <SOS>, <EOS>, <PAD>, and <UNK> were incorporated to mark the start and end of sentences, pad sequences to the same length, and handle unknown words.

Dataset Creation:

A custom XRayDataset class was created to load the images and their corresponding captions. This class handled the pairing of images with their textual descriptions, ensuring that only valid and non-empty reports were included in the dataset. The images were loaded and converted to grayscale, then expanded to three channels. The dataset also provided the ability to return either the raw captions or the numericalized version, depending on the training needs.

Data Loader:

To manage the data during training, a CollateDataset class was implemented. This class ensured that all captions were padded to the same length within a batch, which is essential for efficient training. The data loader was set up to shuffle the data, drop the last incomplete batch, and pin memory for faster data transfer to the CPU.

After preprocessing and loading the data, the model was trained using the prepared dataset. The custom vocabulary and dataset classes allowed for better handling of the complex medical reports, which contributed to a more robust training process.

The Third Approach:

In the third approach, image preprocessing includes resizing images to 224x224 pixels and normalizing them using standard normalization values. This is consistent with common practices for ensuring uniform input dimensions. The dataset is split into training and validation sets, facilitating both model training and evaluation.

Text data preprocessing involves translating medical reports into English using the GoogleTranslator from the `deep_translator` library. This step ensures that all reports are in a consistent language, which is crucial for effective model training. To avoid repeated translation processes in future runs, the translated reports are saved to an Excel file. This saves time and computational resources by reusing the translated texts. The saved translations are then tokenized, with the tokenizer handling padding and truncation to fit the model's input requirements.

The `XrayReportDataset` class is utilized to manage both image and text data, incorporating preprocessing steps for both modalities. This custom dataset class returns tensors for images and tokenized text data, making it well-suited for training models on multimodal data.

Hyperparameter optimization is conducted using Optuna, which adjusts learning rates, batch sizes, and epochs based on the objective function's performance. During training, the best model checkpoints are saved, and metrics are logged with MLflow to monitor progress and performance.

3.3 Evaluation

Evaluating an image captioning model is a crucial aspect of ensuring that the model generates meaningful, relevant, and accurate descriptions of images.

For the **CLIP-GPT2** model, the evaluation criteria used in this project included BLEU, METEOR, ROUGE, and CIDEr scores, each offering a unique perspective on the performance of the model. The evaluation was performed on the test set, including 183 records. Below, I will discuss why these metrics were chosen, their potential shortcomings, and how these shortcomings can be mitigated.

BLEU Score:

BLEU (Bilingual Evaluation Understudy) is a precision-based metric that measures the overlap of n-grams between the generated captions and the reference captions. BLEU was chosen because it is a widely used metric in natural language processing, particularly for tasks like machine translation and image captioning. However, BLEU has limitations, such as its sensitivity to exact word order and its inability to account for semantic meaning, which can lead to misleadingly low scores if the generated captions are semantically correct but differ in phrasing from the reference captions.

Results:

The average BLEU score was 0.0575, indicating a relatively low overlap between the generated captions and the reference captions. The BLEU-1 score was higher (0.209), suggesting some degree of unigram (single word) overlap, but the scores dropped significantly for higher n-grams, highlighting the model's difficulty in generating longer, coherent sequences that match the reference captions.

METEOR Score:

METEOR (Metric for Evaluation of Translation with Explicit ORdering) considers precision, recall, synonymy, stemming, and word order. METEOR is more forgiving than BLEU when it comes to variations in word choice and order, making it a complementary metric to BLEU. However, it is still limited by the quality of the reference captions and the complexity of the task at hand.

Results:

The average METEOR score was 0.267, indicating moderate success in capturing relevant information in the generated captions. The model managed to generate text that was somewhat semantically aligned with the reference captions, but there was still significant room for improvement.

ROUGE Scores:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measures the recall of n-grams, word sequences, and word pairs. ROUGE was selected because it focuses on recall, which is important in medical contexts where missing critical information can be more problematic than generating redundant information. However, ROUGE also shares some of the same limitations as BLEU, particularly in its sensitivity to word order.

Results:

- **ROUGE-1** (unigram recall): 0.269
- **ROUGE-2** (bigram recall): 0.112
- **ROUGE-L** (longest common subsequence recall): 0.212

These results show that the model was better at recalling individual words but struggled with longer sequences, suggesting that the generated captions often missed key phrases or the correct structure needed for a coherent medical report.

CIDEr Score:

CIDEr (Consensus-based Image Description Evaluation) was chosen because it emphasizes the consensus among multiple reference captions, rewarding captions that are closer to the majority view. CIDEr is particularly useful in cases where multiple valid descriptions exist for the same image, as it accounts for this variability.

Results: The CIDEr score was 0.066, which is quite low. This indicates that the generated captions did not align well with the reference captions, reflecting a lack of consensus between what the model generated and what was expected.

Some Examples of the generated captions (Hypothesis) with respect to the ground truth (reference) , on the test dataset:

for higher n-grams suggests that the model struggles with generating longer, coherent sequences that closely match the reference captions.

METEOR Score:

METEOR (Metric for Evaluation of Translation with Explicit ORdering) is a metric that considers precision, recall, synonymy, stemming, and word order. It is more forgiving than BLEU regarding variations in word choice and order, making it a valuable complement to BLEU in image captioning tasks.

Results:

- **METEOR:** 0.2152

The METEOR score suggests that the model captures some relevant information in the generated captions, but there is still significant room for improvement in generating text that is semantically aligned with the reference captions.

ROUGE Scores:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) focuses on recall and is particularly useful in contexts where it is essential to capture critical information. ROUGE measures the recall of n-grams, word sequences, and word pairs, making it a comprehensive metric for evaluating text similarity.

Results:

- **ROUGE-1:** 0.3129
- **ROUGE-2:** 0.1601
- **ROUGE-L:** 0.2834

The ROUGE scores indicate that the model is better at recalling individual words but struggles with capturing longer sequences or the correct structure needed for coherent captions.

Comparison Based on Tests Made for Both Models

When comparing the DenseNet121-based model and the CLIP-GPT2 model based on the evaluation metrics and test results, several key differences and similarities emerge:

1. BLEU Scores:

- **DenseNet121-Based Model:**
 - **BLEU-1:** 0.2042
 - **BLEU-2:** 0.1377
 - **BLEU-3:** 0.0938
 - **BLEU-4:** 0.0492
- **CLIP-GPT2 Model:**
 - **Average BLEU:** 0.0575
 - **BLEU-1:** 0.209
 - **BLEU-2 :** 0.135
 - **BLEU-3:** 0.089

- **BLEU-4:** 0.056

Comparison: The BLEU-1 scores for both models are relatively close, indicating that both models achieve some level of unigram overlap with the reference captions. However, the DenseNet121-based model demonstrates a more gradual decline across BLEU-2, BLEU-3, and BLEU-4, suggesting slightly better performance in generating coherent n-grams sequences. In contrast, the CLIP-GPT2 model shows a sharper decline, indicating more difficulty in generating longer, coherent sequences.

2. METEOR Scores:

- **DenseNet121-Based Model:**
 - **METEOR:** 0.2152
- **CLIP-GPT2 Model:**
 - **METEOR:** 0.267

Comparison: The CLIP-GPT2 model achieves a higher METEOR score than the DenseNet121-based model. This suggests that the CLIP-GPT2 model is better at generating semantically relevant captions that account for variations in word choice and order. The DenseNet121-based model, while performing moderately well, shows room for improvement in capturing the semantic meaning of the reference captions.

3. ROUGE Scores:

- **DenseNet121-Based Model:**
 - **ROUGE-1:** 0.3129
 - **ROUGE-2:** 0.1601
 - **ROUGE-L:** 0.2834
- **CLIP-GPT2 Model:**
 - **ROUGE-1:** 0.269
 - **ROUGE-2:** 0.112
 - **ROUGE-L:** 0.212

Comparison: The DenseNet121-based model outperforms the CLIP-GPT2 model across all ROUGE metrics. The higher ROUGE-1 and ROUGE-L scores indicate that the DenseNet121-based model is more effective at recalling individual words and the longest common subsequences, which are critical for generating captions that capture essential information. The CLIP-GPT2 model, on the other hand, shows lower recall, particularly in bigram sequences (ROUGE-2), which might result in captions that miss key phrases or structure.

Hyperparameter Fine-tuning:

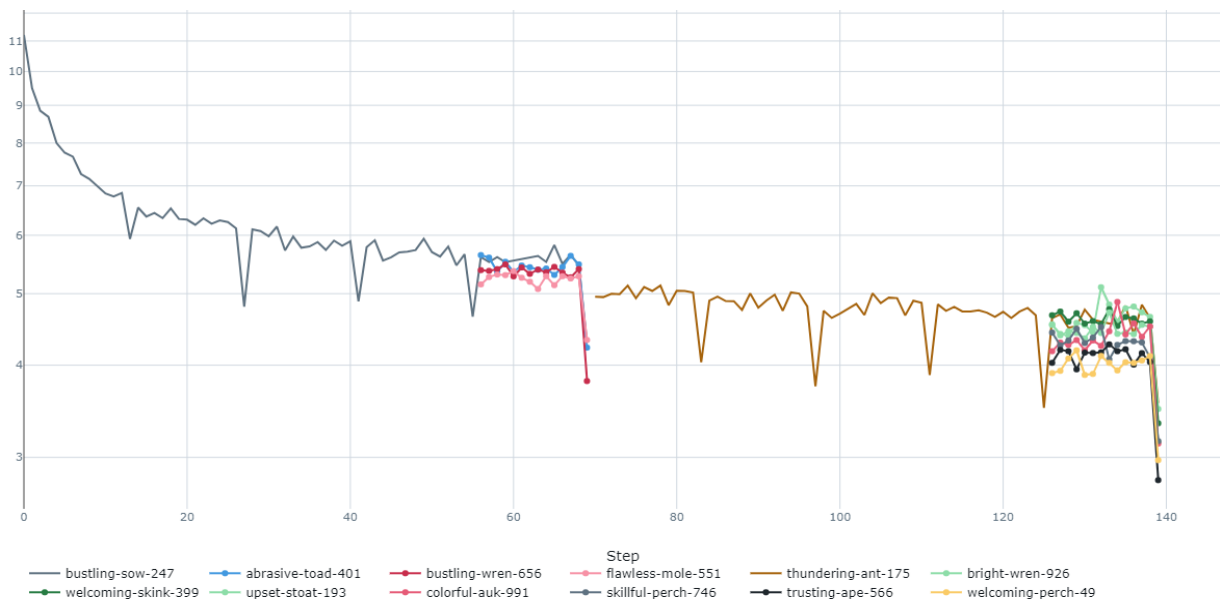
Hyperparameter fine-tuning is a critical component of model evaluation and is necessary to optimize the performance of the model.

For approach 1 to finetune **CLIP-GPT2 models**, the key hyperparameters chosen are learning rate, number of epochs, temperature, and top-p (nucleus sampling). These hyperparameters influence the training process, the model's convergence, and the diversity of the generated captions. The best model having the lowest validation loss was saved on drive, and the monitoring process was done via MLflow framework. The chosen hyper parameter values are:

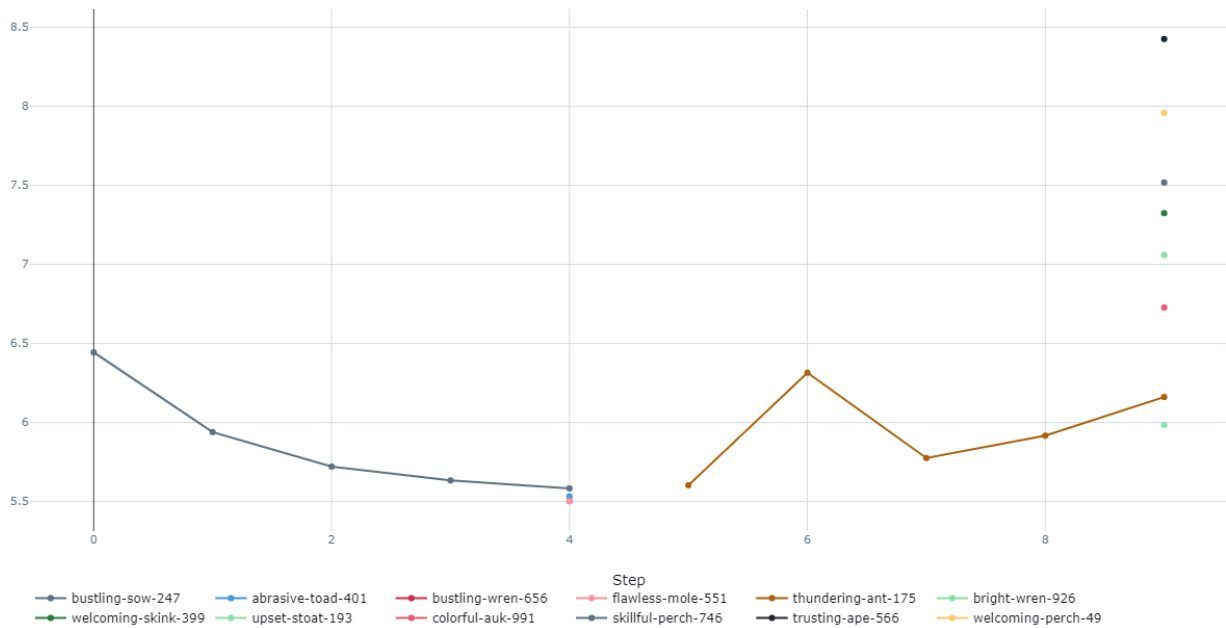
- **Learning Rate:** Affects how quickly the model adapts to the training data. Too high a learning rate may cause the model to converge too quickly to a suboptimal solution, while too low a learning rate may result in slow convergence. The chosen values are: [1e-4, 5e-5].
- **Number of Epochs:** Determines the number of times the model will pass through the entire training dataset. Too few epochs may lead to underfitting, while too many may cause overfitting. The chosen values: [5,10].
- **Temperature:** Controls the randomness of predictions by scaling the logits before applying softmax. Higher values result in more diverse predictions, while lower values make the model more confident in its predictions. The chosen values: [0.7, 1.0].
- **Top-p (Nucleus Sampling):** Limits the sampling to the smallest possible set of logits whose cumulative probability exceeds the threshold p. This helps in balancing between diversity and quality in generated captions. The chosen values: [0.8, 0.9].

Using MLflow, we can visualize the training and validation losses:

Train loss:



Validation loss:



Based on the validation set, we chose the following hyperparameters:

- Number of epochs:5
- Learning rate:0.0001
- Temperature:0.7
- Top_p: 0.8

Here's a snapshot of the runs:

Time created

State: Active

Datasets

+

New run

Sort: validation_loss

Columns

Group by

Table

Chart

Evaluation

Experimental

Traces

Experimental

				Metrics		Parameters			
	Run Name	Created	Duration	train_loss	validation_loss	learning_rate	num_epochs	temperature	top_p
<input type="checkbox"/>	bustling-wren-656	5 days ago	12.3min	3.806624889...	5.503201484...	0.0001	5	1.0	0.8
<input type="checkbox"/>	flawless-mole-551	5 days ago	10.3min	4.327245235...	5.503831386...	0.0001	5	1.0	0.9
<input type="checkbox"/>	abrasive-toad-401	5 days ago	12.3min	4.225388050...	5.532436211...	0.0001	5	0.7	0.9
<input type="checkbox"/>	bustling-sow-247	5 days ago	1.1h	4.290745258...	5.583560784...	0.0001	5	0.7	0.8
<input type="checkbox"/>	bright-wren-926	4 days ago	4.3min	3.488817691...	5.985260486...	0.0001	10	0.7	0.9
<input type="checkbox"/>	thundering-ant-175	4 days ago	40.8min	3.553564786...	6.161348978...	0.0001	10	0.7	0.8
<input type="checkbox"/>	colorful-auk-991	4 days ago	4.2min	3.132360219...	6.727114518...	5e-05	10	0.7	0.8
<input type="checkbox"/>	upset-stoat-193	4 days ago	4.7min	3.147760868...	7.059364318...	0.0001	10	1.0	0.9
<input type="checkbox"/>	welcoming-skink-399	4 days ago	4.3min	3.337025403...	7.324118614...	0.0001	10	1.0	0.8
<input type="checkbox"/>	skillful-perch-746	4 days ago	4.6min	3.155713558...	7.517853736...	5e-05	10	0.7	0.9
<input type="checkbox"/>	welcoming-perch-49	4 days ago	4.2min	2.974439144...	7.957759698...	5e-05	10	1.0	0.9
<input type="checkbox"/>	trusting-ape-566	4 days ago	4.6min	2.794298171...	8.425411542...	5e-05	10	1.0	0.8

16 matching runs

We can also notice based on the table and the above graph that the model starts to overfit for a higher number of epochs. However, this is not a problem since only the best model is saved and loaded later on.

Potential Shortcomings and Mitigation:

- **Metric Limitations:** Metrics like BLEU are precision-based and may not fully capture the semantic richness of the generated text. Metrics like METEOR and CIDEr, which consider synonyms and phrase structure, help address this limitation but may still miss nuanced contextual relevance.
 - **Mitigation:** To overcome these shortcomings, a combination of metrics is used to provide a more comprehensive evaluation. Additionally, qualitative analysis of generated captions through human evaluation can complement quantitative metrics.
- **Overfitting:** Fine-tuning hyperparameters without proper validation can lead to overfitting, where the model performs well on the training data but poorly on unseen data.
 - **Mitigation:** Cross-validation techniques, along with monitoring validation loss and other metrics during training, help in detecting and preventing overfitting.

Addressing Potential Shortcomings:

Validation with Limited Data: External datasets are limited in terms of medical reports, so we can mitigate this shortfall by employing data augmentation techniques. By applying transformations such as rotation, flipping, scaling, and cropping to the existing X-ray images, we can generate additional training data. However, since these techniques don't provide new medical insights, we should be cautious and ensure that the augmented data accurately represents real-world variations.

Transfer Learning: While transfer learning can be effective, it relies on pre-trained models that might not be optimized for medical imaging tasks. To address this, we can fine-tune the pre-trained models on our specific dataset. Additionally, since transfer learning requires sufficient similarity between the pre-training and target tasks, we should evaluate the performance of the fine-tuned models thoroughly to ensure they generalize well to our medical report generation task.

Regular Model Evaluation: Continuously monitoring model performance is crucial, but it may not be sufficient to overcome all shortcomings. To address this, we should implement robust validation procedures, such as cross-validation or train-test splits, to ensure that our evaluation results are reliable and representative of model performance on unseen data. Additionally, we should consider incorporating uncertainty estimation techniques to quantify the reliability of our model predictions and identify potential areas for improvement.

3.4 Baseline Model

The first baseline model was a straightforward implementation of **CLIP and GPT-2** without any fine-tuning. It served as a benchmark to understand the performance gains achieved through model adaptation. The baseline produced captions that were often generic or irrelevant, as well as a lot of meaningless captions and hallucinations that didn't capture the meaning of medical images. This

highlighted the need for fine-tuning to tailor the models to the specific task of generating medical captions.

The second baseline model **Florence 2** was also tested without any fine-tuning as a baseline. While Florence 2 is a powerful multimodal model designed primarily for extracting text from images, in its unadapted state, it focused primarily on providing raw quantitative data rather than generating comprehensive narrative reports. This approach often resulted in outputs that lacked contextual relevance and did not convey the nuanced information required in medical reports. This further emphasized the necessity of fine-tuning to adapt the model to the specific requirements of medical image captioning and report generation.

The third baseline model **DenseNet121-based model** was also tested as a baseline without any fine-tuning. DenseNet121 is well-regarded for its ability to extract rich features from images, particularly in medical contexts. However, when used without adaptation, the model's outputs were limited to basic feature extraction without translating these features into meaningful captions or reports. The results were often incomplete and failed to convey the necessary medical insights, demonstrating that fine-tuning was essential to enhance the model's capability to generate relevant and detailed medical reports.

The fourth baseline model **GPT-2** (vit-gpt2-image-captioning) was used as a baseline model without any fine-tuning. Although GPT-2 is known for its ability to generate coherent text, in this unadopted state, it produced captions that were often disconnected from the actual content of the medical images. The generated text was generic and lacked the specific medical terminology and detailed descriptions necessary for accurate report generation. This further underscored the importance of fine-tuning GPT-2 to better align its outputs with the specialized nature of medical data.

The fifth baseline model, **BioViL-T**, was tested without fine-tuning. BioViL-T is designed for vision-language tasks and can handle medical image and report analysis. However, in its unadopted state, it was unable to leverage its full potential for generating detailed medical reports from X-ray images. The outputs were often generic and did not effectively capture the nuanced information from medical images, reflecting the need for fine-tuning to adapt the model to the specific requirements of medical report generation. This baseline demonstrated that while BioViL-T has the capacity for multimodal tasks, it requires adaptation to produce high-quality and contextually relevant medical reports.

The sixth baseline model, **Kosmos-2**, was tested without fine-tuning. Kosmos-2 is designed for multimodal tasks, leveraging both visual and textual information. In its unadopted state, however, the model struggled to produce meaningful and relevant medical reports. The outputs were often generic and lacked the specific medical terminology and detailed descriptions needed for accurate report generation. This further highlighted the necessity of fine-tuning to tailor Kosmos-2 to the specialized task of medical image captioning and report generation, ensuring that the model could fully leverage its multimodal capabilities.

The seventh baseline model, BLIP, was also tested without fine-tuning. BLIP (Bootstrapping Language-Image Pretraining) is designed to enhance image captioning and understanding by generating descriptive text from images. However, in its unadapted state, BLIP produced outputs that were often generic and lacked the detailed, contextually relevant information required for medical report generation. This baseline highlighted the need for fine-tuning BLIP to improve its capability in generating accurate and comprehensive medical reports from X-ray images.

3.5 Advanced Model

The first advanced model involved fine-tuning both **CLIP** and **GPT-2** on the chest X-ray dataset. Several hyperparameters were tuned, including the learning rate, temperature, and top-p value for GPT-2, to optimize the performance of the model. Additionally, different prompts were experimented with to guide GPT-2's text generation more effectively.

Despite these efforts, the evaluation metrics indicated that the model still struggled with generating accurate and relevant captions. However, the model produces medical captions and no longer hallucinate. The low BLEU and CIDEr scores pointed to a significant gap between the generated and reference captions, especially in capturing longer and more complex sequences.

One key learning was the importance of careful prompt engineering and the use of appropriate input formatting to ensure that the model remained grounded in the visual content provided by CLIP. Additionally, it became clear that further experimentation with different model architectures, such as transformers or attention mechanisms, could be beneficial in improving the model's ability to generate coherent and medically accurate captions.

The second advanced model involved fine-tuning **GPT-2** (vit-gpt2-image-captioning) which consistently produced the same report for different images. This repetition indicated that the model was not effectively learning to differentiate between various inputs, which is crucial for generating accurate and relevant medical reports. Despite adjustments to hyperparameters and prompt engineering, GPT-2's lack of variability in output highlighted its limitations in this context, underscoring the need for further refinement or alternative approaches to achieve the desired level of detail and accuracy.

You can find the fine-tuned model, along with some examples of its output, in the following [Colab](#).

The third advanced model involved fine-tuning the **Florence 2 model**. Despite being highly regarded for its capabilities in multimodal tasks, it did not perform as expected when fine-tuned on the medical dataset. Although literature suggested that Florence 2 would excel in generating reports related to percentages, measurements, and other quantitative data, such as determining the size of a lesion or the percentage of tissue damage, it struggled with our specific task. Instead of generating coherent, human-readable reports in English, the model often produced outputs limited to simple "yes" or "no" responses or extracted text directly from the image when available. In cases where it couldn't find relevant text in the image, it would output "unanswerable." This behavior demonstrated the model's unsuitability for tasks that require detailed textual explanations without a strong numerical focus, as it failed to perform the comprehensive report generation task it was intended for.

You can find the fine-tuned model, along with some examples of its output, in the following [Colab](#).

The fourth advanced model involved fine-tuning **DenseNet121-based model** which showed the most promise. After fine-tuning, it was able to generate reports that closely resembled the actual medical reports, capturing key details from the images. This model demonstrated a strong ability to translate visual data into meaningful textual descriptions, making it the most effective model in our experiments. We believe that with access to more data and longer, more detailed reports, DenseNet121 could achieve even greater accuracy and relevance in its outputs. However, the quality of the generated reports could be further improved if the training data included instances where doctors explicitly described every aspect of the image without relying on patient history, as some current reports are influenced by prior patient information. This adaptation could enhance the model's ability to generate fully detailed and accurate medical reports, making it a valuable tool in clinical settings.

Examples on Model Generated reports and real ones with comparison:

EXAMPLE ONE:

Real: Slight bronchovascular overload at the bases. No consolidation. Normal cardiothoracic index. Free pleural sinuses.

Generated: the heart is normal ict no consolidation no pleural reaction

The reports share some key points, such as the absence of consolidation and normal pleura, which makes them partially similar. However, the generated report lacks the detail about "slight bronchovascular overload" and the specific reference to "free pleural sinuses," which makes the similarity incomplete. Therefore, while there is some overlap, the reports are not fully similar in terms of the details provided.

EXAMPLE TWO:

Real: The heart is normal in size and shape. Lungs are clear showing no nodular or infiltrative changes. Hila and mediastinum are normal. The pleural space is free, rib cage is intact.

Generated: The heart is normal in size and shape lungs are clear showing no nodular or infiltrative changes hila and mediastinum are normal the pleural space

The generated report is very similar to the real report in terms of content, covering all major findings except for the incomplete sentence at the end and the missing mention of the rib cage. The core observations about the heart, lungs, hila, and mediastinum are the same. Overall, the reports are considered similar, with only minor differences in detail and completeness.

EXAMPLE THREE:

Real: Asymmetry of lung transparency. Basal infiltrates noted. No pleural reaction. Dilated and unfolded descending aorta. ICT is at the upper limit of normal. Free trachea. No pleural reaction.

Generated: excess transparency of the right lung fields no consolidation no pleural reaction

The reports share some similarities, particularly regarding lung transparency and the absence of pleural reaction. However, the real report includes significantly more details, such as basal infiltrates, aortic dilation, ICT status, and trachea condition, which are not captured in the generated report. Additionally, the generated report's mention of "excess transparency of the right lung fields" provides specificity not found in the real report. Overall, while there is some overlap, the reports are not considered fully similar due to the differences in detail and completeness. The generated report is missing several critical observations present in the real report.

EXAMPLE FOUR:

real: Asymmetric bronchitic infiltrate at the bases. No consolidation. Trachea is free. Symmetrical domes. No pleural reaction.

generated: the heart is normal in size and shape lungs are clear showing no nodular or infiltrative changes and mediastinum are normal the pleural space is

The reports are not considered similar. They describe different findings, with the real report focusing on the presence of an asymmetric bronchitic infiltrate and the condition of the trachea and domes. The generated report, on the other hand, describes normal heart, lung, and mediastinum conditions but lacks the specific details found in the real report, and its mention of the pleural space is incomplete. Therefore, the content and focus of the two reports differ significantly.

EXAMPLE FIVE:

Real: Lungs are clear. The cardio-thoracic silhouette is normal. Rib cage is intact.

Generated: the heart is normal in size and shape lungs are clear showing no nodular or infiltrative changes hilli and mediastinum are normal the pleural space

The reports are somewhat similar in that they both confirm clear lungs and normal conditions related to the heart or cardio-thoracic silhouette. However, the generated report provides additional details about the hilli and mediastinum, while the real report includes information about the rib cage, which is not mentioned in the generated report. The incomplete sentence in the generated report also makes it less complete. Overall, the reports share some key findings but differ in the details provided, leading to a partial similarity but not full equivalence.

You can find more in this [chatgpt](#) chat where there are more couples of real and generated reports with comparison between them.

The fifth advanced model, **BioViL-T**, was implemented with high expectations for integrating text and image data to produce comprehensive medical reports. Despite its sophisticated architecture, the model faced significant challenges during fine-tuning. One major issue was its tendency to output vectors rather than complete text, which severely impacted the quality of the generated reports. Efforts to address this included adjusting the maximum length settings and exploring different configurations to enhance text generation. However, these adjustments did not yield satisfactory results, as the model continued to produce outputs that lacked coherence and detail. This experience underscored the difficulty of adapting BioViL-T to generate accurate and

contextually rich medical reports and highlighted the need for further exploration and refinement to achieve the desired performance.

The sixth advanced model involved attempting to fine-tune **Kosmos-2** on the chest X-ray dataset. Persistent technical issues related to the model's large size and GPU compatibility, including difficulties with loading and managing resources, prevented us from reaching the testing phase. Despite multiple attempts to address these problems, we were unable to evaluate Kosmos-2's performance and need to explore further solutions.

The seventh advanced model, **BLIP**, was fine-tuned with the chest X-ray dataset. Despite its strong capability in generating descriptive text from images, BLIP did not perform well in this context. The model often produced the same report for different images, even after adjustments to hyperparameters like `max_length`, `num_beams`, and `early_stopping=True`. The evaluation scores for BLIP were as follows:

- **ROUGE Scores:** {'rouge1': 0.2382, 'rouge2': 0.0999, 'rougeL': 0.2082}
- **BLEU Score:** 0.0584
- **METEOR Score:** 0.1855

These scores were lower than those achieved by DenseNet121 and CLIP, highlighting the model's struggles with differentiating between images and generating varied, contextually relevant reports. The uniformity in its outputs could be attributed to overfitting and the relatively small size of the dataset, which may have limited the model's ability to generalize across different images. Further refinements, additional data, and tailored fine-tuning are needed to enhance BLIP's performance and enable it to generate more varied and contextually relevant medical reports. You could find the notebook [here](#).

4 User Interface

Our user interface will prioritize simplicity and ease of use, ensuring that anyone can interact with it effortlessly. Given our focus on model development and the time constraints of the project, the user interface may not include all the features of a full-fledged software application. However, it will offer essential functionality.

The user interface will be developed as a web application using a .NET API for the backend and Angular for the frontend. This setup will allow users to easily navigate the platform, take appointments with doctors, and upload X-ray images for analysis. Once an image is uploaded, our trained model will process it and generate a comprehensive medical report based on the findings. Users will have the option to download the generated report in PDF format, ensuring that the reports can be easily shared with healthcare professionals or added to patients' records, just as hospitals typically provide to patients.

This combination of features will create a user-friendly and practical interface that meets the needs of both patients and healthcare providers.



User Name*

Password*

Login

Invalid Password [Close](#)

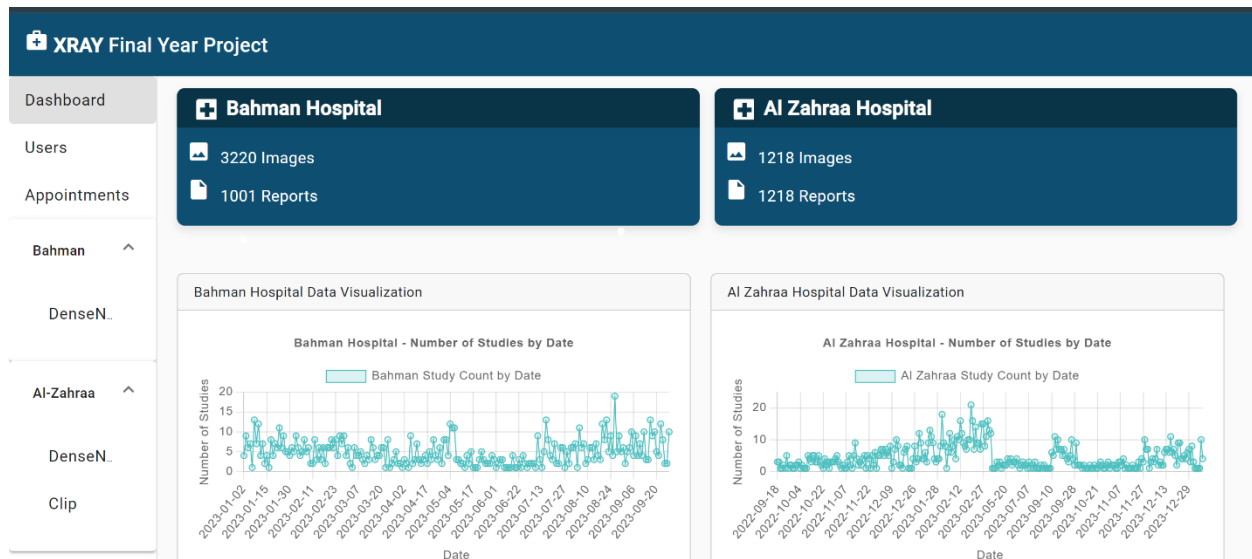


Password*

Password is required

Login

The project begins with a **login page** that prompts users to enter their username and password. This page is equipped with input validation to handle errors effectively. If a user attempts to log in without providing the necessary credentials, or if the input values are incorrect, the system will display appropriate error messages. For example, if the password field is left blank, a "Password is required" message will appear, guiding the user to correct their input before proceeding. This ensures that only valid login attempts are processed, maintaining the security and integrity of the system.



The **dashboard** provides an overview of the available data from Bahman and Al Zahraa Hospitals, focusing on the number of studies conducted over time, as the study date was the only variable suitable for visualization. While the data was limited, this visualization effectively highlights trends in study activity for each hospital. The dashboard's design is straightforward, displaying key metrics like the total number of images and reports, allowing users to quickly grasp the essential information despite the constraints of the dataset.

Dashboard

Appointments







Bahman ▾

Al-Zahraa ▾

Appointments

[+ Take Appointment](#)

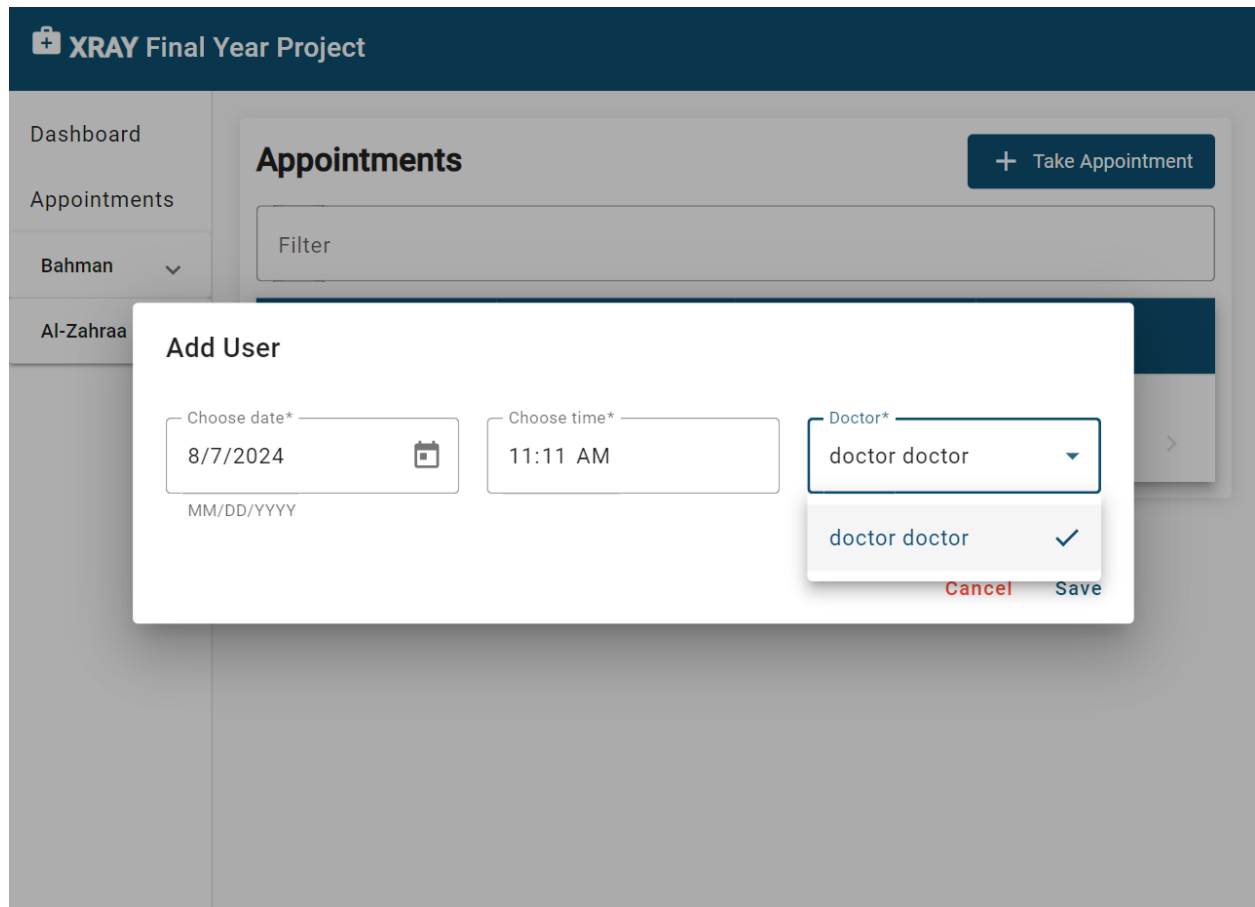
Filter

Patient Id	Doctor Id	Appointment Date	Actions
12	11	2024-08-07T08:11:00	 
12	11	2024-08-11T21:33:00	 
12	11	2024-08-08T23:02:00	 

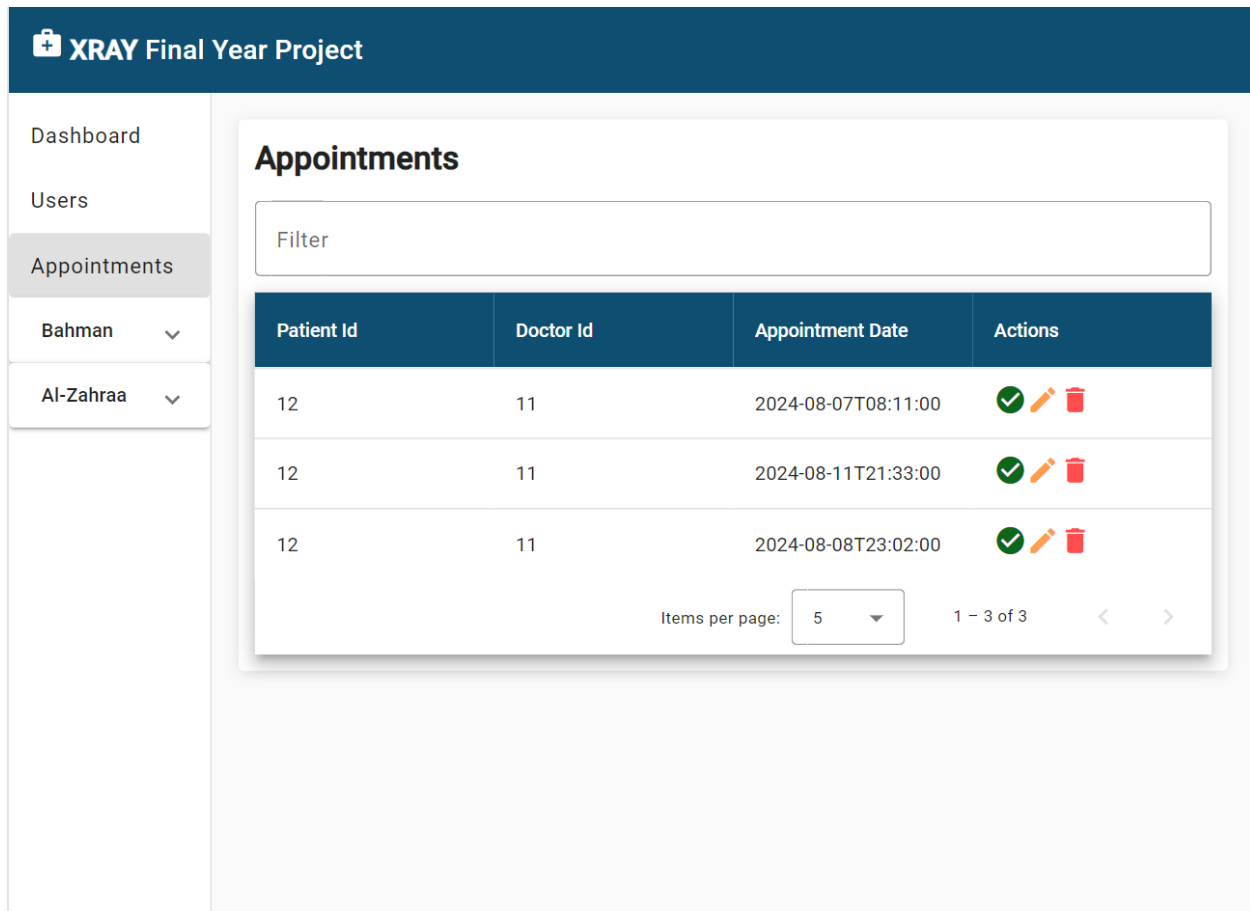
Items per page: 5 ▾

1 - 3 of 3

< >



The images above logged as a patient



The image above logged as a admin

The **appointments feature** is designed to be role-specific, providing different functionalities based on the user's role. If the logged-in user is an admin, they have access to view all appointments, with the ability to accept pending appointments and delete them if necessary. For doctors, the interface displays only the appointments scheduled with them, allowing them to view and accept their pending appointments. Patients see only their own appointments and have the exclusive ability to schedule new ones. This role-based functionality ensures that each user interacts with the system in a manner appropriate to their role, making the appointment process efficient and secure.

In the "Add Appointment" feature, the selector allows users to choose from a list of doctors that have been added to the database. Additionally, there are input fields for selecting the date and time of the appointment, ensuring that patients can schedule their appointments with specific doctors at their preferred time slots. This functionality is essential for creating a seamless and user-friendly appointment scheduling process.

Dashboard

Users

Appointm


Bahman

Al-Zahraa

Users

 Add User

Add User

First Name* doctor	Middle Name* doctor	Last Name* doctor
Name* doctor	Email* doctor@gmail.com	Choose a date* 8/13/2024 
Gender* Female	Roles* Doctor	

MM/DD/YYYY

Cancel

Save

Dashboard

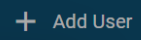
Users

Appointm

Bahman

Al-Zahraa

Users



Update User

First Name*

admin

Middle Name*

Last Name*

admin

Name*

Email*

admin@gmail.com

Choose a date*



MM/DD/YYYY

Gender*

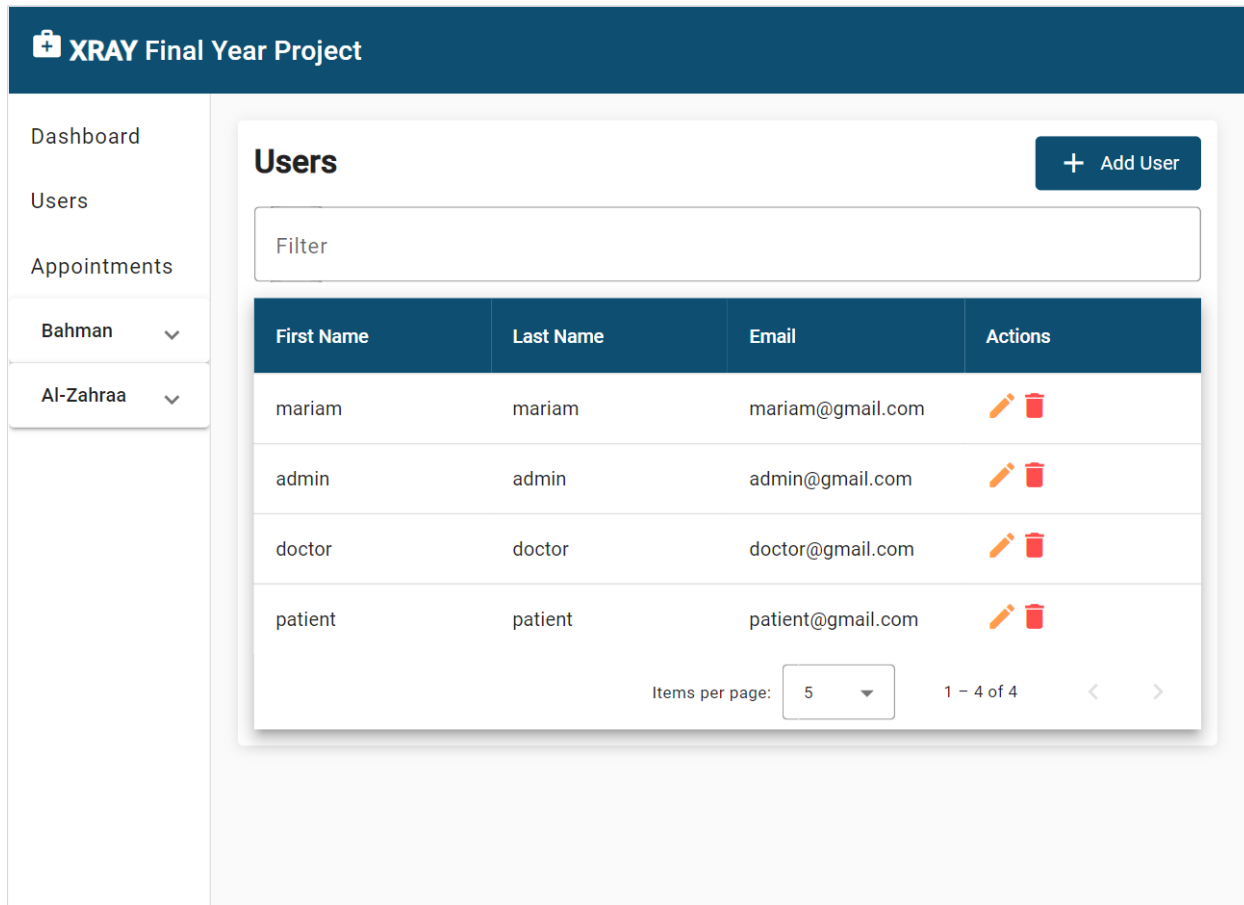


Roles*



Cancel

Save



The above images logged as an admin

The "Users" management feature is exclusively accessible to admins. Only admins can view the list of users, add new users, update existing users' details, or delete users from the system. Neither patients nor doctors have access to this feature. When adding a new user, the system automatically increments the user ID, and the default password for all new accounts is set to "P@\$\$w0rd." This ensures that the process of user management remains secure and controlled, with admins having the authority to manage user access and credentials within the system.

Additionally, all input fields in the user management feature come with built-in error messages to ensure data integrity. These validations include mandatory fields marked as required, ensuring that users cannot submit forms without providing necessary information. The date fields are validated to ensure the input is in the correct date format, and email fields require a valid email address format. The role selector is pre-populated with the three specific roles defined in our project: Admin, Doctor, and Patient, ensuring that users are assigned appropriate roles during the creation or update process. These validations and role restrictions help maintain the integrity and functionality of the system.

Dashboard

Users

Appointments

Bahman ^

DenseN_

Al-Zahraa v

Chest X-Ray Report Generator

Select Chest X-Ray Image:

Choose file No file chosen

Generate Report

Dashboard

Users

Appointments

Bahman ^

DenseN_

Al-Zahraa v

Chest X-Ray Report Generator

Select Chest X-Ray Image:

Choose file 22E7634.jpg

Generate Report

Http failure response for http://localhost:5000/report/generate: 0 Unknown Error

Chest X-Ray Report Generator

Select Chest X-Ray Image:

Choose file | 1000405_B.png

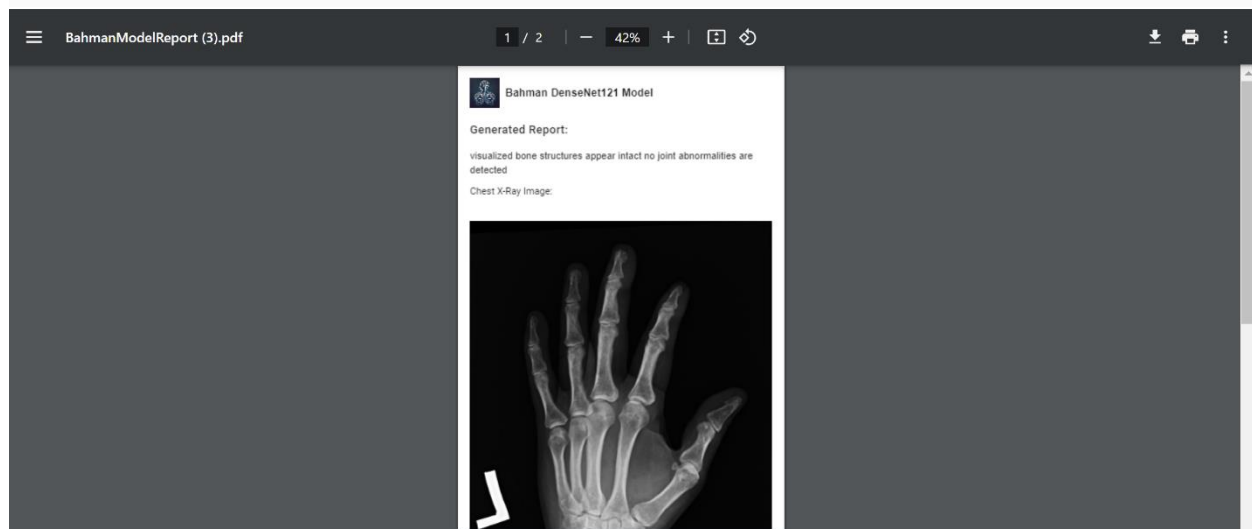
Generate Report

 Bahman DenseNet121 Model

Generated Report:
visualized bone structures appear intact no joint abnormalities are detected

Chest X-Ray Image:


Export as PDF



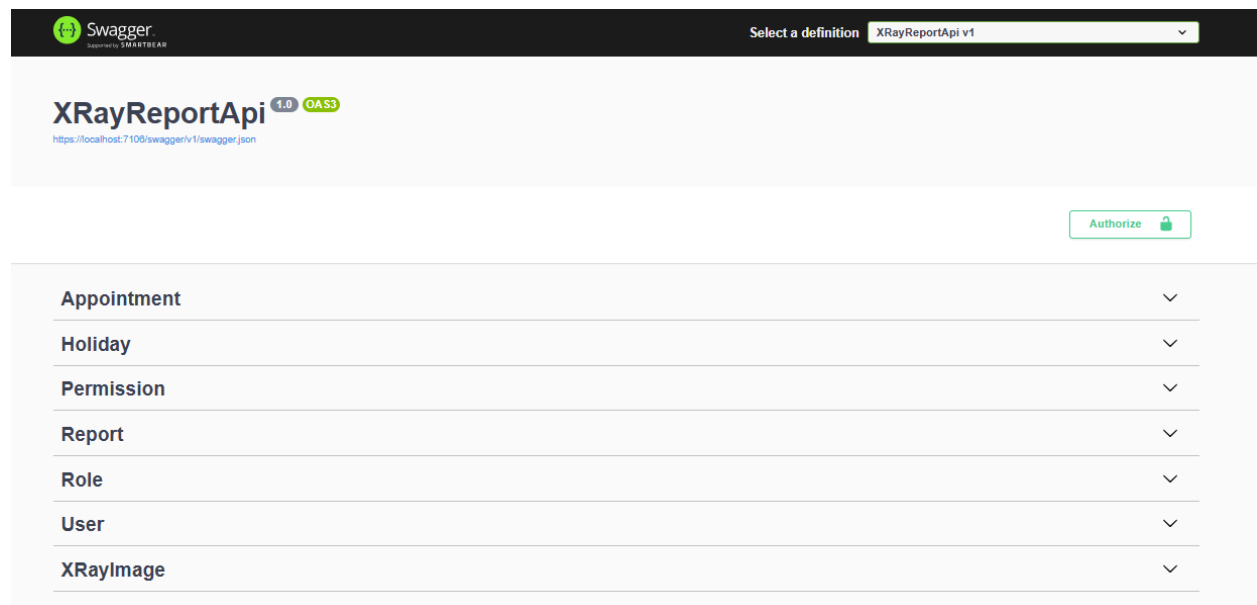
The "**X-Ray Report Generator**" feature allows users to upload a chest X-ray image and generate a corresponding medical report. When the "Generate Report" button is pressed, the system first checks if an image has been uploaded. If no image is uploaded, an error message is displayed, prompting the user to upload a file. Additionally, if there is an issue with the API during the report generation process, an error message is shown, indicating that something went wrong. If the process completes successfully, the generated report is displayed along with the uploaded image

underneath. Users can then export the report as a PDF, as illustrated in the image, making it easy to share or store the report for future reference.

This is implemented for the 3 models 2 for denseNet121 one for Bahman's data and the other for Alzahraa's data and the third is clip model on Alzahraa's data.

NOTE It's important to note that all permissions and role-based functionalities are enforced solely on the frontend of the application. There is currently no backend security implementation, meaning that role validation and access controls are managed through the user interface without backend verification. This leaves the application vulnerable to potential security risks, as the backend lacks mechanisms to authenticate or enforce permissions independently of the frontend.

Backend Implementation



Below is an overview of the key models and methods implemented:

1. User Model

User		^
POST	/api/User/Login	▼
POST	/api/User/register	▼
GET	/api/User/GetAllUsers	▼
GET	/api/User/GetUserById/{userId}	▼
PUT	/api/User/Update/{userId}	▼
DELETE	/api/User/Delete/{userId}	▼

The User model is central to the application's authentication and authorization mechanisms. It represents the users of the system, who can have different roles such as Admin, Doctor, or Patient.

- **Methods:**

- `Login(LoginDto request)`: Authenticates a user by validating their credentials and generating a token for session management. This method ensures that only authorized users can access protected resources.
- `Register(UserDto request)`: Registers a new user into the system. This method includes several key steps:
 - **Validation**: The user's input is validated to ensure the provided email, username, and password meet the required criteria.
 - **Duplicate Check**: Before creating a new user, the system checks if the email or username is already in use to prevent duplicate accounts.
 - **Password Hashing**: The password is hashed using a secure algorithm before storing it in the database, ensuring that plain text passwords are never saved.
 - **Role Assignment**: The user is assigned a role (Admin, Doctor, or Patient) based on the registration input. Role-specific permissions are then linked to the user.
 - **Token Generation**: Upon successful registration, the user may receive a token to automatically log them in, improving the user experience.
- `GetAllUsers()`: Retrieves all users from the database, typically used by admins for management purposes.
- `GetUserById(long userId)`: Retrieves user details by their ID, which is used to display user information in the user profile or by the admin.
- `UpdateUser(int userId, UserDto updatedUserDto)`: Updates user information, such as email, username, or password, with validation checks similar to those in the registration process.
- `DeleteUser(int userId)`: Deletes a user from the system. This method is used for account deactivation or removal by an admin.

2. Role Model

Role		^
GET	/api/Role/GetAllRoles	✓
POST	/api/Role/AddRole	✓
PUT	/api/Role/UpdateRole	✓
DELETE	/api/Role/Delete/{id}	✓
GET	/api/Role/GetRoleById/{id}	✓

The Role model defines the different roles a user can have within the system, such as Admin, Doctor, and Patient.

- **Methods:**

- `GetRoles()`: Retrieves all roles from the system.
- `AddRole(RoleDTO roleDTO)`: Adds a new role to the system, typically used by an admin to introduce new roles.
- `UpdateRole(RoleDTO roleDTO)`: Updates an existing role, allowing modification of role attributes or permissions.
- `DeleteRole(long id)`: Deletes a role by its ID. Before deletion, the system checks if the role is assigned to any users to prevent issues.
- `GetRoleById(long id)`: Retrieves details of a specific role by its ID, useful for role management interfaces.

3. Permission Model

Permission		^
GET	/api/Permission/GetAllPermissions	▼
POST	/api/Permission/AddPermission	▼
PUT	/api/Permission/UpdatePermission	▼
DELETE	/api/Permission/Delete/{id}	▼
GET	/api/Permission/GetPermissionById/{id}	▼
GET	/api/Permission/GetPermissionsByRole/{roleId}	▼

The Permission model defines the various permissions that can be assigned to roles. Permissions control what actions a role can perform.

- **Methods:**

- `GetPermissions()`: Retrieves all permissions.
- `AddPermission(PermissionDTO permissionDTO)`: Adds a new permission to the system, allowing the creation of new capabilities.
- `UpdatePermission(PermissionDTO permissionDTO)`: Updates an existing permission, adjusting the scope of what a role can do.
- `DeletePermission(long id)`: Deletes a permission by its ID. The system ensures that removing a permission doesn't leave roles without necessary access.
- `GetPermissionById(long id)`: Retrieves details of a specific permission by its ID.
- `GetPermissionsByRole(long roleId)`: Retrieves permissions assigned to a specific role, used in role management and access control checks.

4. RolePermission Model

The RolePermission model establishes the many-to-many relationship between roles and permissions. This model ensures that each role has a defined set of permissions.

- **Methods:** Typically, the CRUD operations for this model would be managed by associating roles with permissions using the PermissionController.

5. Holiday Model

Holiday		^
GET	/api/Holiday/GetAllHolidays	▼
POST	/api/Holiday/AddHoliday	▼
PUT	/api/Holiday/UpdateHoliday	▼
DELETE	/api/Holiday/Delete/{id}	▼
GET	/api/Holiday/GetHolidayById/{id}	▼

The Holiday model is used to define holidays within the system. These dates are excluded from the appointment booking process.

- **Methods:**
 - GetHolidays(): Retrieves all holidays.
 - AddHoliday(HolidayDTO holidayDTO): Adds a new holiday.
 - tUpdateHoliday(HolidayDTO holidayDTO): Updates an existing holiday, allowing modifications to holiday dates or descriptions.
 - DeleteHoliday(long id): Deletes a holiday by its ID. The system checks if any appointments are affected before allowing deletion.
 - GetHolidaybyId(long id): Retrieves details of a specific holiday by its ID.

6. Appointment Model

Appointment		^
GET	/api/Appointment/GetAllAppointments	▼
POST	/api/Appointment/AddAppointment	▼
PUT	/api/Appointment/UpdateAppointment	▼
DELETE	/api/Appointment/Delete/{id}	▼
GET	/api/Appointment/GetAppointmentById/{id}	▼
GET	/api/Appointment/GetAppointmentsByPatientId/{patientId}	▼
GET	/api/Appointment/GetAppointmentsByDoctorId/{doctorId}	▼

The Appointment model handles the scheduling of appointments between patients and doctors. This model includes special logic to ensure appointments are scheduled correctly and do not conflict with holidays or existing bookings.

- **Methods:**

- `GetAppointments()`: Retrieves all appointments.
- `AddAppointment(AppointmentDto appointmentDTO)`: Schedules a new appointment with the following special checks:
 - **Holiday Check**: The system verifies that the appointment date is not a holiday. If the chosen date is a holiday, the system prompts the user to select another date.
 - **Overlap Check**: The system checks for existing appointments to prevent double booking. If there is a conflict, the user is informed and asked to choose a different time slot.
 - **Doctor Availability**: The system ensures that the doctor is available at the requested time, taking into account their existing schedule and working hours.
- `UpdateAppointment(AppointmentDto appointmentDTO)`: Updates an existing appointment with similar checks as `AddAppointment`, ensuring that changes do not introduce conflicts or fall on holidays.
- `DeleteAppointment(long id)`: Deletes an appointment by its ID, freeing up the time slot for other patients.
- `GetAppointmentById(long id)`: Retrieves details of a specific appointment by its ID, allowing users to view or modify their bookings.
- `GetAppointmentsByPatientId(long patientId)`: Retrieves all appointments for a specific patient, providing a history of past and upcoming appointments.
- `GetAppointmentsByDoctorId(long doctorId)`: Retrieves all appointments for a specific doctor, helping doctors manage their schedules.

7. XRayImage Model

XRayImage		^
GET	/api/XRayImage/GetAllXRayImages	▼
DELETE	/api/XRayImage/Delete/{id}	▼
GET	/api/XRayImage/GetXRayImageById/{id}	▼
POST	/api/XRayImage/UploadXRayImage	▼

The XRayImage model manages the storage and retrieval of X-ray images.

- **Methods:**
 - `GetXRayImages()`: Retrieves all X-ray images.
 - `UploadXRayImage(IFormFile file, long userId)`: Uploads a new X-ray image for a specific user. The system ensures that the file format is valid and the file size is within acceptable limits.
 - `DeleteXRayImage(long id)`: Deletes an X-ray image by its ID, ensuring that the image is no longer needed before deletion.
 - `GetXRayImagebyId(long id)`: Retrieves details of a specific X-ray image by its ID, linking it to the associated report.

8. Report Model

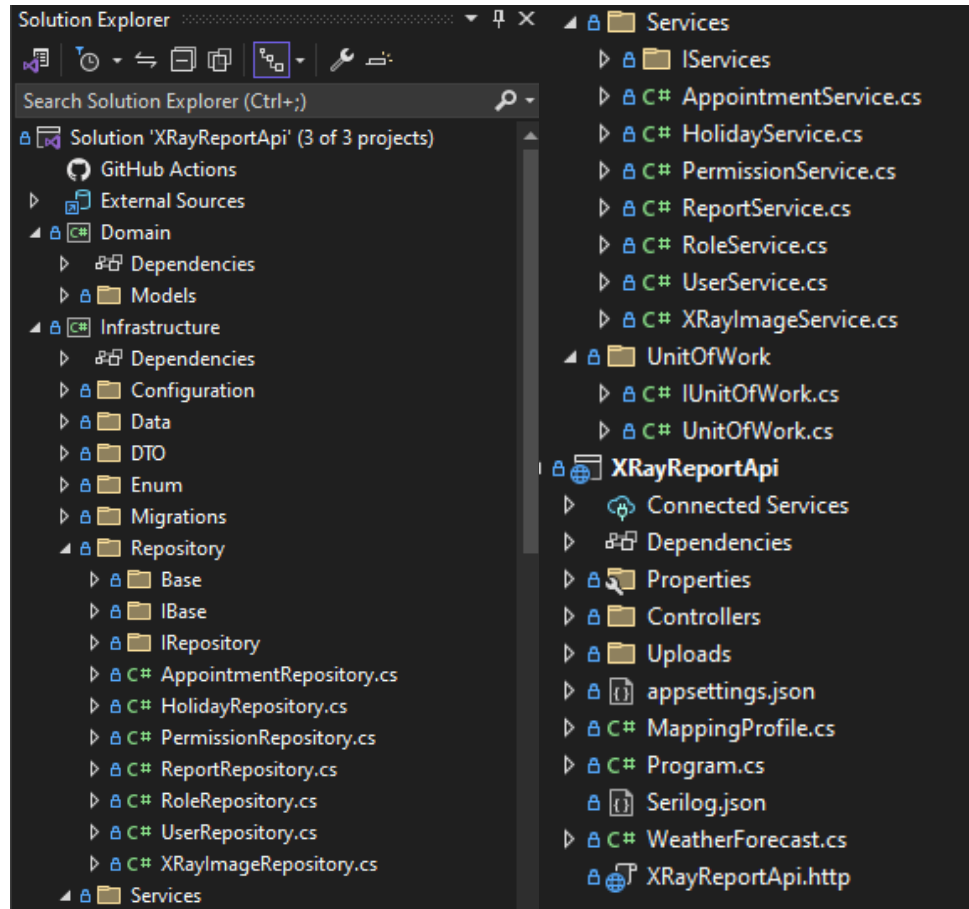
Report ^	
GET	/api/Report/GetAllReports
POST	/api/Report/AddReport
PUT	/api/Report/UpdateReport
DELETE	/api/Report/Delete/{id}
GET	/api/Report/GetReportById/{id}
GET	/api/Report/GetReportByXRayImageId/{id}

The Report model is responsible for storing and retrieving reports generated from X-ray images. The reports are automatically generated by the system based on the X-ray images uploaded by patients.

- **Methods:**

- `GetReports()`: Retrieves all reports.
- `AddReport(ReportDTO reportDTO)`: Adds a new report to the system. The report is typically generated by the system using the X-ray image and stored for future reference.
- `UpdateReport(ReportDTO reportDTO)`: Updates an existing report, allowing corrections or updates based on new information.
- `DeleteReport(long id)`: Deletes a report by its ID. The system ensures that the report is no longer needed before allowing deletion.
- `GetReportbyId(long id)`: Retrieves details of a specific report by its ID, linking it to the X-ray image and patient.
- `GetReportByXRayImageId(long id)`: Retrieves a report based on the associated X-ray image ID, enabling doctors and patients to view the results of the analysis.

Backend Architecture



The backend of the X-Ray Report Generation System is implemented using Clean Architecture principles. This architectural approach divides the system into several layers, ensuring a clear separation of concerns, promoting maintainability, scalability, and testability.

1. Domain Layer

The Domain layer is the core of the application, containing the business logic and entities (e.g., User, Role, Permission). This layer is independent of any external frameworks, ensuring that the core business rules are isolated and can evolve without being affected by changes in the infrastructure or presentation layers.

2. Infrastructure Layer

The Infrastructure layer handles the technical implementation details that support the domain layer. It includes:

- **Repositories:** These classes manage data access and persistence, implementing the necessary CRUD operations for each entity.
- **Configuration:** Manages setup details such as database connections.
- **Data:** Contains the database context.
- **Migrations:** Contains the migrations for managing the database schema.
- **DTOs and Enums:** Facilitates data transfer between layers and defines constants used throughout the application.

3. Services Layer

The Services layer encapsulates the application's business logic, implementing the use cases that orchestrate operations across different repositories. It ensures that the business rules are consistently applied and handles the interaction between the domain and infrastructure layers.

4. UnitOfWork Layer

The UnitOfWork pattern is employed to ensure that all operations performed in a single transaction are either fully completed or rolled back, maintaining data consistency and integrity.

5. XRayReportApi Layer

This is the entry point of the application, where the API controllers are defined. It includes:

- **Controllers:** These manage incoming HTTP requests, interacting with the service layer to process and respond to them.
- **Program.cs and Configuration Files:** They are responsible for configuring the application, including dependency injection, logging, and environment-specific settings

5 Challenges

Throughout our project, we encountered several significant challenges that tested our problem-solving skills and resilience:

- Obtaining real data from the hospital for training our model proved time-consuming due to legal and ethical considerations, causing substantial delays in our project timeline. Although we initially expected to collect more extensive data, we faced unexpected constraints. Despite coordinating with Al Zahraa hospital, where the maximum number of team members allowed to work on data extraction was four, we were only able to gather 1,200 reports. This was less than anticipated due to delays in receiving data from Bahman hospital, which ultimately informed us late in the process that they could not provide the data we needed. Fortunately, Al Zahraa hospital welcomed our continued efforts, allowing us to proceed, but within the limited scope.
- Selecting the appropriate model architecture was also a significant challenge due to the novelty of our idea and the limited existing resources. Extensive research and consultation

with experts were required to identify suitable models, highlighting the importance of adaptability and the exploration of unconventional solutions.

- Accessing sufficient computational resources for training our model, particularly GPUs, proved to be a logistical hurdle. The university's GPU resources were difficult to access on time, and their usage was constrained by time limits and attendance requirements, which did not align well with our project needs. As a result, we resorted to using Google Colab, but this came with its own set of limitations, including restricted session durations and frequent interruptions due to RAM and GPU overloads. These constraints forced us to create multiple accounts to continue our work, which was a cumbersome process. Given these challenges, we decided to train one of the models locally on a personal computer. This process took more than 7 hours of continuous training, causing the laptop to overheat, which further complicated the process.
- Model deployment using ONNX to be integrated to ml.net. The exported model using ONNX yielded inaccurate results in testing after deployment. This included hallucinations and incoherent output.

6 Future Work

Based on the findings and outcomes of our project, several potential avenues for future research, development, and enhancement emerge:

- **Retraining and Fine-Tuning Models:** Enhancing model performance by retraining on more specific medical imaging tasks and larger, diverse datasets. Using transfer learning to leverage pre-trained models could also improve results.
- **Semantic Understanding and Contextual Adaptation:** Boosting the model's ability to understand medical terms and context through advanced techniques like attention mechanisms and contextual embeddings to produce more accurate reports.
- **Integration of Multimodal Information:** Adding more types of data, like clinical metadata or patient history, could help the model understand images better and create more detailed reports. Using structured data from electronic health records (EHRs) or imaging metadata can offer useful context. However, testing reports that use only the current image and exclude patient history might reduce bias and help the model generalize better.
- **User Interface Refinement and Deployment:** Enhancing the interface for better usability by healthcare professionals, incorporating feedback, and exploring real-world deployment to improve workflow and patient care.
- **Collaborative Research and Validation Studies:** Collaborating with healthcare institutions and domain experts to conduct validation studies and clinical trials could validate the effectiveness and utility of the system in real-world settings. Collecting feedback from radiologists, clinicians, and other stakeholders through user studies and surveys could help identify areas for improvement and guide future development efforts.
- **Involvement of Experts for Formal Report Writing:** Collaborating with experts to create detailed reports based solely on image findings (without patient history) could improve the

model's training and accuracy. This method helps the model focus on the image data itself, enhancing its generalization and report quality.

- **Incorporation of Reinforcement Learning**, where the model could be trained to optimize its outputs based on specific reward signals, such as the accuracy of medical terminology or alignment with expert-generated reports. This approach would allow the model to learn from feedback, continually refining its ability to produce high-quality medical descriptions tailored to specific diagnostic needs. Incorporating RL could lead to more sophisticated and reliable report generation, ultimately contributing to better clinical decision-making and patient care.
- **Implementing a continuous training pipeline**, where this pipeline would involve periodically retraining the model using newly available user data, including images, the generated reports, and corresponding ground truth reports provided by doctors. By comparing the generated reports to the expert-written reports, the model can learn from its mistakes and adjust its outputs accordingly. This continuous feedback loop would enable the model to stay up-to-date with the latest medical knowledge and reporting standards, ensuring that the generated reports remain accurate, relevant, and aligned with the expectations of healthcare professionals.

7 Conclusion

In conclusion, this project highlights the efficacy of leveraging advanced transformer-based architectures and multimodal models like CLIP and GPT-2 for generating detailed medical reports from chest X-ray images. While challenges such as data limitations and model selection complexities were encountered, the project successfully bridged the gap between visual data and natural language generation. The findings emphasize the need for robust data preprocessing, thorough evaluation metrics, and the importance of iterative model refinement.

Future work could focus on retraining models on more extensive and diverse datasets, optimizing user interfaces for clinical integration, and conducting extensive validation studies to ensure reliability in real-world medical applications. By addressing these areas, the system's potential to enhance diagnostic accuracy, streamline clinical workflows, and ultimately improve patient care outcomes in medical imaging becomes more tangible. The project's outcomes lay a solid foundation for further advancements in the intersection of AI and healthcare, contributing valuable insights for the development of more sophisticated medical imaging solutions.

8 Bibliography

1. Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2014). Show and tell: A neural image caption generator. *arXiv*. <https://doi.org/10.48550/arXiv.1411.4555>

2. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *arXiv*. <https://doi.org/10.48550/arXiv.1502.03044>
3. **Hugging Face.** (n.d.). *microsoft/BiomedVLP-BioViL-T*. Hugging Face. <https://huggingface.co/microsoft/BiomedVLP-BioViL-T>
4. **Hugging Face.** (n.d.). *microsoft/kosmos-2.5*. Hugging Face. <https://huggingface.co/microsoft/kosmos-2.5>
5. Salesforce. (2024). *BLIP image captioning base*. Hugging Face. <https://huggingface.co/Salesforce/blip-image-captioning-base>
6. Ghandi, T., Pourreza, H., & Mahyar, H. (2022). Deep learning approaches on image captioning: A review. *arXiv*. <https://doi.org/10.48550/arXiv.2201.12944>
7. Liu, S., Bai, L., Hu, Y., & Wang, H. (2018). Image captioning based on deep neural networks. *MATEC Web of Conferences*, 232, 01052. <https://doi.org/10.1051/mateconf/201823201052>
8. Bennani, S., Regnard, N.-E., Ventre, J., Lassalle, L., Nguyen, T., Ducarouge, A., Dargent, L., Guillo, E., Gouhier, E., Zaimi, S.-H., Canniff, E., Malandrin, C., Khafagy, P., Koulakian, H., Revel, M.-P., & Chassagnon, G. (2023). Using AI to improve radiologist performance in detection of abnormalities on chest radiographs. *Radiology*. <https://doi.org/10.1148/radiol.230860>
9. Wang, J., Yang, Z., Hu, X., Li, L., Lin, K., Gan, Z., Liu, Z., Liu, C., & Wang, L. (2022). GIT: A generative image-to-text transformer for vision and language. *arXiv*. <https://paperswithcode.com/paper/git-a-generative-image-to-text-transformer>
10. Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., & Wu, Y. (2022). CoCa: Contrastive captioners are image-text foundation models. *arXiv*. <https://paperswithcode.com/paper/coca-contrastive-captioners-are-image-text>