Report for the assignment of the "`Automatic Medical Report Generation`" project

Biamonte Salvatore 264177, Di Franco Federico 263678, Visciglia Domenico 263631

A.Y. 2024/2025

Contents

1 Introduction **2**

2 Dataset Analysis and Cleaning: Chest X-rays (Indiana University) **2**
2.1 Distribution of Problems into Dataset 3
2.2 Data Problems 3
2.3 Dataset Cleaning and Classification Strategy 4

3 Experimental Architectures and Model Development **4**
3.1 From Early Ideas to Final Solutions 5
3.2 Parametric Code, Modular Extensions and Custom Metrics 5

4 Final Models: Architectures, Performance and Comparison **6**
4.1 Data Splits 6
4.2 Binary Classification (Malato/Non Malato) + GPT-2 7
4.3 Multi-class Classification (Body Zones) + GPT-2 8
4.4 Bonus Model: Multi-class (Body Zones) + Custom Transformer 10
4.5 Comparison Models and Summary of Results 11

5 Discussion and Conclusions **12**

A Source Code **13**

1 Introduction

Medical imaging plays a pivotal role in patient diagnostics and treatment planning, yet radiology reports often depend on time-consuming, specialized medical expertise. This project aims to automate the generation of such reports by analyzing **Chest X-ray** data from the Indiana University repository. Our overarching objectives are:

- **Develop flexible architectures** capable of classifying X-rays (e.g., binary or multi-class) and generating radiology-style captions.
- **Improve data quality and consistency** by performing thorough dataset analysis, cleaning, and re-labeling steps, which proved essential due to issues like class imbalance and ambiguous labeling.
- **Compare multiple models and approaches**, including both pretrained (GPT-2) and custom (Transformer) text decoders, to identify trade-offs in accuracy and caption fluency.

Prior work in the state of the art often applies CNNs for image feature extraction and RNN/LSTM models for text generation. While these methods achieve solid results, they often over-describe normal findings or struggle with detailed abnormality localization. We therefore integrated *attention mechanisms* and risk-based categorization to focus on the most relevant pathological cues in each X-ray.

A central challenge we faced was the *dataset analysis and cleaning*: many labels were rare or inconsistently described, and large subsets of data were malformed. Ensuring coherent labeling and splitting the data suitably (80% train, 20% test, with an internal validation partition) thus became a keystone for reliable model performance. The next sections detail how we tackled these issues and iterated toward final architectures that balance classification accuracy and text-generation quality.

2 Dataset Analysis and Cleaning: Chest X-rays (Indiana University)

In this phase, we focused on analyzing the **Chest X-rays (Indiana University)** dataset to understand its structure, assess data quality, and identify potential challenges for subsequent modeling and analysis tasks. The dataset includes frontal and lateral chest x-ray images, accompanied by details and medical analyses provided in an associated CSV file. For our study, we selected the following columns:

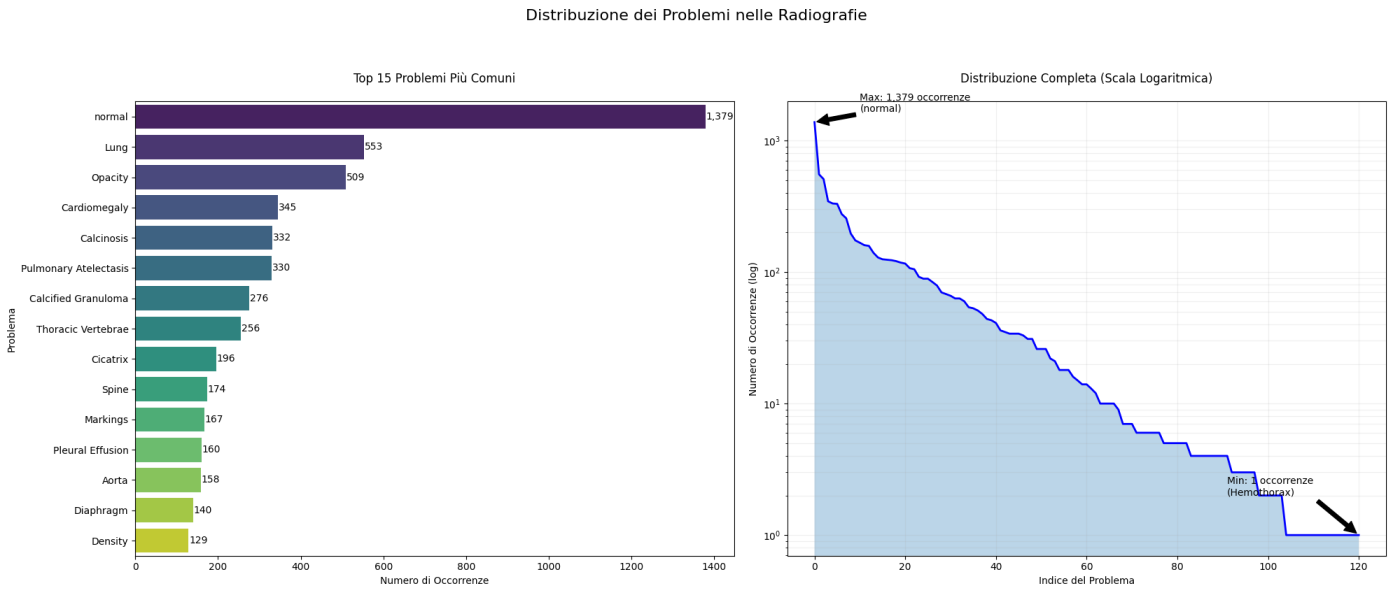
- **Problems**: Describes the medical issues identified from the x-rays.
- **Findings**: Provides the radiologist's detailed analysis.
- **Images**: Includes the frontal and lateral x-ray images.

The choice of columns was made after several iterations. Initially, we intended to use the **MeSH** column, which contains more specific terms for the problems. However, the MeSH labels introduced significant challenges:

- The dataset included over **1400 unique MeSH labels**, most of which occurred only once.
- These MeSH terms represented the same problems described in the **Problems** column but at a much finer granularity.

Given this, we opted to use the **Problems** column as an abstraction of the MeSH terms, allowing us to work with a more manageable set of labels while retaining the relevant information.

2.1 Distribution of Problems into Dataset



The dataset includes **121 unique problem labels**, distributed in a highly imbalanced manner. For instance:

- The label **"normal"** accounts for **1379 occurrences**, making it the most represented class.
- Intermediate conditions, such as **"Lung"** with **553 occurrences** or **"Opacity"** with **509 occurrences**, have a noticeable presence.
- Conversely, rare conditions like **"Hemothorax"** or **"Fibrosis"** appear only once, complicating classification due to insufficient data.

This imbalance poses a significant challenge for developing robust predictive models, as the dominance of certain classes skews the learning process.

2.2 Data Problems

Data Quality and Formatting Issues Out of the 3851 rows in the dataset: **520 rows** (13.5%) were identified as malformed, either due to missing information or corrupted entries. The presence of incomplete or inaccurate data compromises the reliability of any conclusions drawn without extensive cleaning.

Ambiguity in Problem Labels A notable issue is the inconsistency in the labeling of problems:

- **Anatomical regions** (e.g., "Lung," "Aorta") are labeled similarly to specific medical conditions (e.g., "Cardiomegaly"), leading to ambiguity.
- **Inconsistent terminology** results in potential duplication, as the same condition might appear under different labels.

For example, while "Lung" indicates an anatomical region, it is indistinguishable from pathological labels in the current format. This lack of clarity complicates downstream tasks like data categorization and model training.

2.3 Dataset Cleaning and Classification Strategy

After removing rarely occurring problems (which contributed little to training) and discarding rows with incomplete or corrupted information, we initially introduced a **four-level risk** classification (No Risk, Low, Medium, High). However, since the same pathology could imply different risk levels depending on the patient, this scheme proved too inconsistent. Consequently, we pivoted toward more coherent and more stable approaches:

- **Binary Division (Problems vs. No Problems):** Images were labeled either *Malato* (any pathology detected) or *Non Malato*. This straightforward method yielded strong performance metrics and a clear, high-level distinction.
- **Broader System Labels (Four Classes):** As an alternative multi-class setup, problems were grouped by body regions: *Pulmonary*, *Cardiovascular*, *Musculoskeletal*, or *No Problem*. This objective categorization aligned well with our final architectures and code parameterization.

These refined classification strategies, combined with the cleaned dataset, formed the basis for all subsequent modeling experiments.

3 Experimental Architectures and Model Development

Throughout the project, our goal was to build flexible architectures capable of both learning a latent representation from chest X-ray images and producing meaningful outputs, such as classification labels (binary or multi-class) and textual reports. Initially, we considered a *multi-class risk* strategy (i.e., no risk, low risk, moderate risk, high risk). However, this posed some inconsistencies: the same pathological condition in two different patients might imply distinct risk levels, depending on their clinical history or other factors. Consequently, we evolved towards two primary versions of our model, plus a bonus variant:

- **Binary Classification (Malato/Non Malato).** This approach provides a simpler and more coherent label: an image is either "ill" (presence of a pathological condition) or "healthy" (no detected condition). By collapsing the original four risk classes into a single binary decision, we avoid confounding risk factors that can vary per patient. This model is paired with our *GPT-2*-based text generation for producing descriptive radiology reports.
- **Multi-class Classification by Body System.** Recognizing that certain body systems (e.g. pulmonary, cardiovascular, musculoskeletal) represent a more objective descriptor of the location or nature of the detected issues, we introduced a second multi-class architecture. Specifically, the model classifies into "No Problem," "Pulmonary System," "Cardiovascular System," or "Musculoskeletal System," then employs GPT-2 for generating the final report. Because our code base was already written in a highly parameterized fashion (from earlier attempts at multi-risk classification), adapting it to multiple zones proved straightforward and clean.
- **Bonus Model (Zones + Custom Transformer).** As an extra demonstration of our framework's modularity, we implemented a bonus variant of the multi-class body-system model. In this version, instead of using GPT-2 for caption/report generation, we developed a custom Transformer decoder. The rest of the pipeline remains essentially the same—the classification head still predicts "No Problem," "Pulmonary," "Cardiovascular," or "Musculoskeletal"—but the text generation leverages our own Transformer.

3.1 From Early Ideas to Final Solutions

In the early stages of the project, we tested a range of approaches:

1. **Segmentation-Focused and Compact Networks.** Alongside DenseNet121, EfficientNetB0, and DeepLabV3 variants, we also developed a *U-Net* from scratch and trained on the our dataset. While these laid some groundwork, we encountered challenges in balancing overfitting, interpretability, and classification accuracy.

2. **Autoencoder Trained Entirely on Our Dataset.** We implemented a custom autoencoder, training all weights solely on the chest X-ray images at hand. While this approach offered the freedom to design a tailored latent space, it often failed to converge stably and yielded insufficient detail for reliable medical captioning.

3. **ResNet50 at Higher Resolution + Shared Latent Space.** Ultimately, we opted for a *ResNet50* encoder owing to its robustness. We increased the input size from the usual 224×224 pixels to 384×384, a supported ResNet standard, improving the network's ability to capture finer image details. A single latent space of dimension 4096 (concatenated frontal and lateral features) is then:

   - *Used directly by the classifier*, predicting the relevant categories.
   - *Reshaped and fed into GPT-2*, enabling caption generation via the same extracted features.

   This unified approach boosted both classification accuracy and descriptive quality.

3.2 Parametric Code, Modular Extensions and Custom Metrics

Not a Standard Binary Before delving into how we switch seamlessly between binary and multi-class outputs, it is worth highlighting that *even for the "malato/non malato" scenario*, we do **not** employ a single-neuron layer with a sigmoid. Instead, we treat this as a two-class *softmax* problem within the same multi-class framework, ensuring consistent parametric design. Although we briefly tested a more conventional "binary" approach (single neuron + sigmoid) in a separate model (referenced in our supplementary Kaggle notebooks), the results were essentially identical or slightly poorer than our softmax-based solution. This confirms that our multi-class coding approach for binary classification does not hinder performance, and indeed streamlines the overall architecture.

Having a parametric code base from the outset allowed us to easily configure the number of output classes and seamlessly switch from multi-class to binary classification (or vice versa). This design choice became a key advantage:

- **Binary *vs.* Multi-class Switch.** Transitioning from the original four risk categories to a simpler "malato/non malato" classification was mainly a change in the final classification layer—the rest of the pipeline (ResNet, GPT-2, etc.) remained unaffected.

- **Multi-class by Anatomical System.** Introducing zone-based classification demanded only an updated label-mapping strategy and the addition of a few lines of code to specify four output classes (no problem, pulmonary, cardiovascular, musculoskeletal).

- **Bonus Transformer.** Because GPT-2 was cleanly separated in our model code, we replaced it with our custom Transformer for the bonus version. This required minimal architectural changes, highlighting the versatility of our approach.

Not a Standard Loss We extended our model training with two custom loss functions that address the specific challenges of imbalanced classes and the need to emphasize critical medical words in captioning:

- **Focal Loss (Slightly Modified).** We adapted the standard Focal Loss to handle class imbalance more effectively by introducing two key hyperparameters: α, which biases the weight towards rarer classes if needed (e.g., $\alpha > 1$ penalizes false negatives more strongly), and γ, which adjusts the focus on difficult examples. Our version applies the cross-entropy in `reduction='none'` mode, then raises $(1 - \text{pt})^\gamma$ to re-weight the loss term. This allows us to mitigate the dominance of majority classes in the training process.

- **Medical Caption Loss (Increased Penalties for Critical Words).** When generating captions describing radiological findings, certain words (e.g., pathology names and anatomical locations) must be predicted accurately. Our *MedicalCaptionLoss* extends a standard cross-entropy computation by assigning higher weights to these crucial words:

  1. We maintain two internal regex patterns: one for medical problems (e.g. "cardiomegaly," "pneumothorax," etc.) and another for key anatomical locations (e.g. "lung," "rib," "vertebra").
  2. During the forward pass, for each caption in the batch, we identify which token IDs correspond to these words and create a *weight mask* that elevates their importance by a factor (e.g. `problem_terms_weight`=2.0).
  3. We then compute standard cross-entropy on the decoder logits *shifted* for language modeling. The selected tokens—i.e. those that match our medical or anatomical patterns—are penalized more if predicted incorrectly, thus encouraging the model to focus on critical terminology.

  This approach proved beneficial for highlighting key diagnoses and anatomical references in generated text, as it systematically penalizes omissions or mistakes in critical words at a higher rate than less important words.

In the following sections, we analyze these final solutions in detail, presenting experimental results and comparisons across both the classification tasks (binary and multi-class) and the alternative text-generation modules (GPT-2 vs. custom Transformer).

4 Final Models: Architectures, Performance and Comparison

In this section, we present the three final models developed for the project—two based on **GPT-2** (one using binary classification, the other using multi-class classification) and a **bonus model** that replaces GPT-2 with a custom Transformer. The results reported are the outcome of the average of the **about ten experiments** carried out. We also describe how the dataset was partitioned and provide an overview of each model's performance.

4.1 Data Splits

All three models use the same data-splitting strategy:

- We first separate *Train* (80% of the full dataset) from *Test* (20%).
- The Training portion is further divided into an internal *Train* (70% of that subset) and *Validation* (30%).

Hence, in total, about 56% of the data is for actual training, 24% for validation, and 20% for final testing. This approach is kept consistent across all three architectures, ensuring fair comparisons of classification accuracy and text quality.

4.2 Binary Classification (Malato/Non Malato) + GPT-2

Motivation. We have reduced the risk-based classification from four categories to two: "Sick" versus "Not sick" since the same pathology or problem does not necessarily indicate the same risk for two different patients, given that it also depends on the medical chart and other external factors, consequently this change has made the classification more useful and correct.

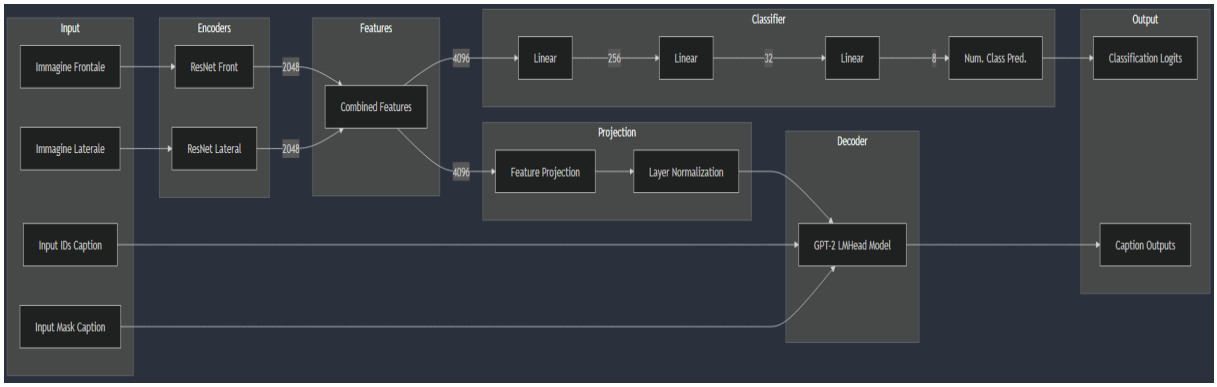


Figure 1: Architecture of Models using GPT2 exemplified

Architecture.

1. **Two Encoders (ResNet-50):** We use two ResNet-50 networks to process the frontal and lateral images, respectively. Each encoder outputs a 2048-dimensional vector. These two feature vectors are concatenated into a single 4096-dimensional representation.

2. **Binary Classifier:** The concatenated feature (size 4096) passes through a feed-forward block consisting of multiple layers (`Linear` → `BatchNorm` → `ReLU` → `Dropout`), ultimately producing logits for two classes (Malato/Non Malatos).

   - *Motivation:* This binary approach identifies whether there is any pathological finding on the X-ray.
   - *Loss Function:* We typically use either cross-entropy or *Focal Loss* to handle possible class imbalance.

3. **Feature Projection for GPT-2:** In parallel, the same 4096 feature is linearly projected (down to GPT-2's hidden size) and normalized by a `LayerNorm`. This output is interpreted as "context" or "encoder hidden states" for GPT-2's cross-attention mechanism.

4. **GPT-2 Decoder:** A pretrained GPT-2 (with *cross-attention* enabled) receives:

   - *Textual Inputs:* The tokenized caption (e.g. from the radiology report).
   - *Projected Features:* From the concatenated ResNet encoders.

   During training, we utilize the pretrained weights of GPT-2 but **unfreeze all layers except the first one**. This strategy ensures efficient adaptation of the GPT-2 decoder to the medical captioning task. GPT-2 then generates the textual description (a radiology-style caption) by attending to the image features. During training, a *caption loss* (using `MedicalCaptionLoss`, the custom cross-entropy explained previously) is added to the binary classification loss.

5. **Combined Training:** Each training step computes:

- *Classification Loss* (for Malato/Non Malato), which backpropagates into the ResNet layers and the classifier.
- *Caption Loss*, which updates the GPT-2 decoder, the projection layer, and partially the ResNet if desired.

A total loss combines both. Different `param_groups` allow the GPT-2 part to have a higher learning rate than the classifier/resnet part.

Performance Overview. With the above data split, the classification accuracy on the test set it often reaches *values around* 95%. In the meantime, textual metrics such as BLEU-4 or ROUGE-L are also approaching the state of the art, although the exact numbers may vary and improve slightly by continuing to search for the best train by varying weights, epochs, etc. , these results indicate that GPT-2 can consistently produce short, clinically oriented captions (for example, mentioning possible pulmonary or cardiac findings).

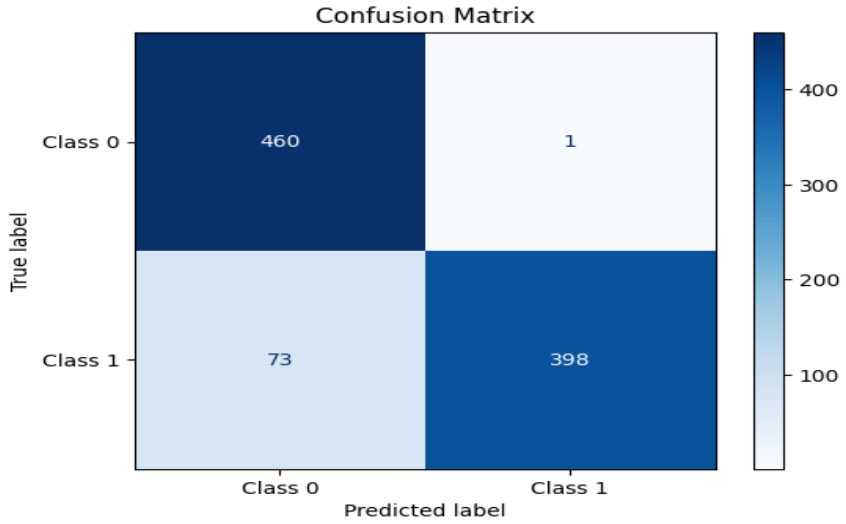


Figure 2: Confusion Matrix of the Binary Classification + GPT-2 Model

4.3 Multi-class Classification (Body Zones) + GPT-2

Motivation. Instead of assigning a "risk level" to a condition – which may differ from patient to patient – we decided to exploit the parametric nature of our model to classify pathologies into four specific "zones", classification that provides valuable information to the doctor. In this model, we also tested the effect of different learning rate strategies by grouping the model components based on their roles:

- **Group 1 (Higher Learning Rate):** `model.decoder` (GPT-2), `model.feature_projection` and `model.projection_layer_norm`
- **Group 2 (Original Learning Rate):** `model.classifier`, `model.encoder_front` and `model.encoder_lateral`

This strategy allowed us to independently tune the learning rates of the text-generation and image-classification components. However, we observed that allowing all weights to update together led to better generalization and slightly improved results overall.

Architecture. Structurally, this model retains exactly the same **ResNet-50 encoders**, **feature concatenation**, and **GPT-2 decoding** pipeline as the binary case. The *only* architectural change lies in **how classification is handled**:

- Instead of producing two logits (Malato/Non Malato), the final classifier layer generates *four* logits for the categories:

  Nessun Problema, *Sistema Polmonare*, *Sistema Cardiovascolare*, *Sistema Muscoloscheletrico*.

- The rest of the classification block is identical: we simply replace the output dimension of the last `Linear` layer from 2 to 4. No other code changes are required in the model itself, thanks to the parametric design.
- Training proceeds analogously, but the **classification loss** now uses a four-class cross-entropy instead of a binary version.

Why This Change is Possible in This Model. Because the code is parametrized via `num_class_pred`, switching from 2 classes to 4 requires:

1. Adjusting the dataset labeling (assigning each image to a body-system label).
2. Changing `num_class_pred = 4` in the model.
3. Using the same training loop but with four-class targets and a corresponding loss function (e.g. Focal Loss for multi-class).

Hence, aside from the classification head's final layer, the architecture for multi-class GPT-2 is identical to the binary version: **same double** ResNet encoder for frontal/lateral X-rays, **same linear + layer-normalization projection** into GPT-2's hidden dimension and GPT-2 **remains the text generator** with cross-attention.

Performance Overview. Accuracy values *generally fall around* 90% for the best classes in many series, although small variations may occur. In general, the multiclass approach is highly interpretable because each image is linked to a distinct one body system, and subsequent text indicates relevant findings. Furthermore, even for the ROUGE-L and BLUE-4 metrics the results are very similar, with small variations depending how the metrics are calculated (e.g. BLEU-4 penalizes empty predictions less than ROUGE-L) or the random distribution of the data for the train and the test.

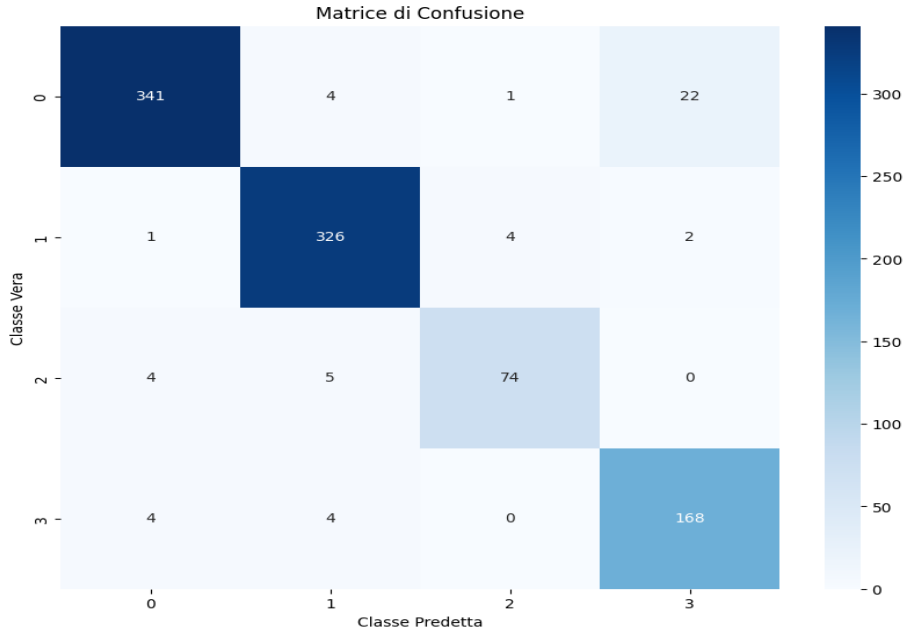


Figure 3: Confusion Matrix of the Multi-class Classification (Body Zones) + GPT-2 Model

4.4 Bonus Model: Multi-class (Body Zones) + Custom Transformer

Motivation. We developed a model with a customized transformer as requested, furthermore it was possible to highlight the modularity of our code and test the pre-trained capabilities of GPT-2. We have maintained the multiclass classification (4 zones) head but replaced GPT-2 with a hand-rolled Transformer decoder.

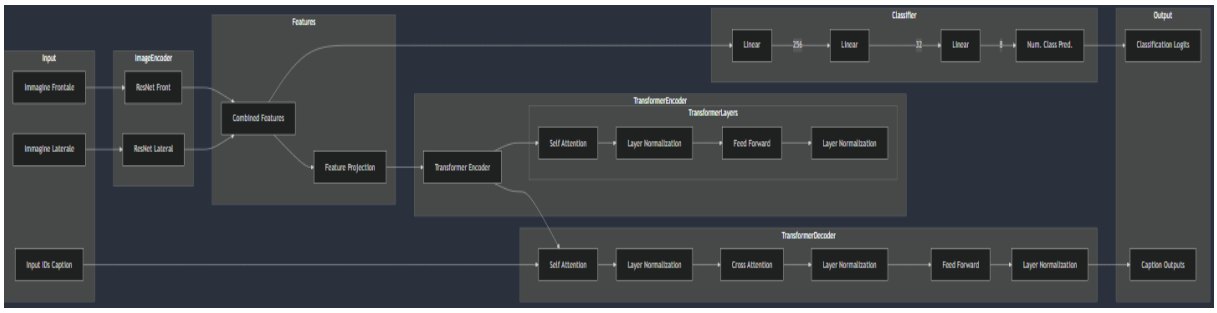


Figure 4: Architecture of Model using Custom Transformer exemplified

Architecture. Structurally, this model still relies on the **two ResNet-50 encoders** (frontal and lateral) plus a **multi-class classifier** exactly as in the GPT-2 versions. The major change is how *text generation* is done: instead of using GPT-2's pretrained components, we replace them with a **custom Transformer** built from scratch. Specifically:

1. **ResNet-50 + Classifier (Same as Multi-Class GPT-2).** We encode each image with ResNet-50 (2048-dim each), concatenate (4096-dim), and feed this vector into a multi-layer classifier that outputs four logits (*Nessun Problema*, *Sistema Polmonare*, *Sistema Cardiovascolare*, *Sistema Muscoloscheletrico*).

2. **Custom Transformer Encoder.** Rather than a simple projection for GPT-2 cross-attention, we:

   - Project the 4096-dim image features to a d-dim "image token."
   - Embed any textual "problem description" (if available) via a small embedding layer + positional encoding.
   - Concatenate *image token* + *problem tokens* and pass them through a standard `Transformer Encoder` (multi-head self-attention + feed-forward layers).

3. **Custom Transformer Decoder.** We implement a decoder with *masked self-attention* and *cross-attention* to the encoder outputs (image token + problem text). This is trained from scratch (unlike GPT-2, which is pretrained) and uses:

   - Token embeddings (vocab-size-based).
   - Sinusoidal positional encoding.
   - Multiple decoder layers (each with self-attention, cross-attention, and feed-forward).

4. **Caption Generation and Loss.** Training includes:

   - *Classification Loss* on the 4-body-system prediction (updating the ResNet + classifier).
   - *Caption Loss* (similar to cross-entropy) for auto-regressive text generation, updating the custom encoder+decoder.

We optionally add teacher forcing during training (feeding ground-truth tokens into the decoder) and can decode with greedy or beam search at inference time.

In short, *we replace GPT-2's pretrained layers* with a fully custom Transformer encoder-decoder, learning everything from the image token and problem text embeddings up to the final output projection layer.

Performance Overview. Classification accuracy remains similar (*about the same* 95% range observed with GPT-2). While for the textual metrics, as regards ROUGE-L the *value remains around* 0.21 (as with GPT-2), however a big difference can be noticed with BLEU-4, where the *value is around* 0.05 - 0.06, significantly lower than the model with GPT-2 (*about* 0.1), probably because the custom decoder does not have large-scale pretraining. However, it can generate consistent results, short sentences referring to relevant body systems.

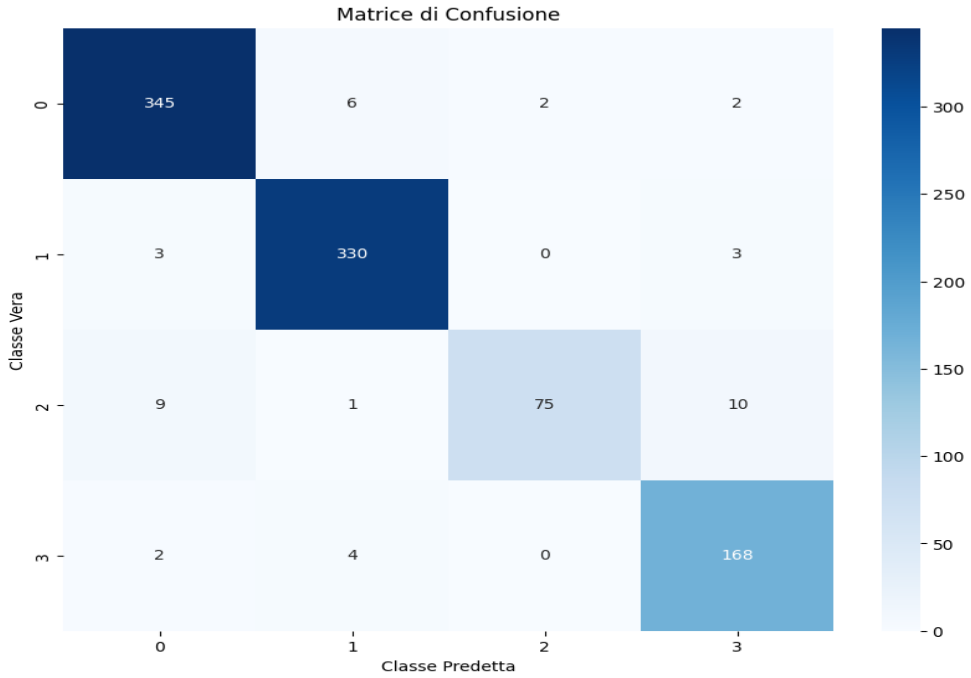


Figure 5: Confusion Matrix of the Multi-class Classification + Custom Transformer Model

4.5 Comparison Models and Summary of Results

All three architectures share the same ResNet-based pipeline, differing mostly in their classification outputs and text-generation backbones. In general:

- **Binary (Malato/Non Malato).** Yields simpler labels and tends to maintaining high test accuracies achieve higher classification accuracy. GPT-2 provides concise statements about pathology presence.
- **Multi-Class (Body Zones) + GPT-2.** Focuses on anatomical problem location, achieve higher classification accuracy and reasonably fluent radiological text.
- **Custom Transformer.** Achieves similar classification performance, though caption generation can be marginally less fluent than GPT-2, as it is trained from scratch.

Reference	BLEU-4	ROUGE-L
State of the Art	0.177	0.372
Binary-GPT2	0.140	0.163
MultiClass-GPT2	0.124	0.201
MultiClass-CustomTrans	0.055	0.212

Table 1: Comparison of final results across our implementations and a reference state-of-the-art.

Reference	Acc. Class 0	Acc. Class 1	Acc. Class 2	Acc. Class 3	Overall Acc.
Binary-GPT2	97.8%	93.0%	-	-	95.4%
MultiClass-GPT2	87.0%	96.1%	86.3%	91.4%	90.9%
MultiClass-CustomTrans	97.2%	98.2%	78.9%	96.5%	95.6%

Table 2: Comparison between class and overall accuracies in our models.

Details regarding single-class accuracies, confusion matrices, loss curves, and extended metrics are available in the attached notebooks.

In conclusion, these final models demonstrate the flexibility of our pipeline: one can switch between binary or multi-class labeling, select GPT-2 or a custom decoder, and still train both classification and captioning objectives in a unified framework. This yields consistent results *around* the aforementioned accuracy ranges and text-generation scores without extensive rewriting of the underlying code.

5 Discussion and Conclusions

In summary, these results confirm that our architectural choices—particularly the use of dual ResNet encoders, flexible classification heads, and dedicated text-generation modules—significantly impact final performance. The shift from risk-based labeling to a simpler binary or zone-based strategy helped resolve inconsistencies in risk definitions, ultimately boosting classification reliability.

Looking ahead, there is clear room for refinement:

- **Augmenting the Dataset.** A more comprehensive, balanced, and clinically diverse set of X-rays would reduce overfitting and further validate the models' generalizability.
- **Exploring Larger Pretrained Transformers.** Integrating advanced models (e.g., GPT-3 or custom Vision-Language Transformers) could improve the richness and coherence of generated reports.
- **Full Integration with Medical Histories.** Risk levels could be revisited if correlated patient metadata becomes available, enabling a dynamic notion of severity tailored to each individual.

Given the report's size constraints, we could not detail every intermediate architecture or experimental trial. However, *all models and their training histories are documented* in the accompanying notebooks, where one can trace the entire evolution of this project. Overall, our pipeline demonstrates how careful dataset analysis and modular design can effectively unify classification tasks with automated captioning, paving the way for broader clinical applications.

A Source Code

Submitted Models

- **Binary-GPT2:** *https://www.kaggle.com/code/dom3n1co/binarymalatonon-gpt2*
- **MultiClass-ZoneCategories-GPT2:** *https://www.kaggle.com/code/dom3n1co/multiclasszonecategory gpt2*
- **MultiClass-ZoneCategories-CustomTrans:** *https://www.kaggle.com/code/fefe202/transformer-ksampling-nucleus*

Pretrained Models

- **Binary-GPT2-Pretrained:** *https://www.kaggle.com/code/dom3n1co/binarymalatonon-gpt2-pretrained*
- **MultiClass-ZoneCategories-GPT2-Pretrained:** *https://www.kaggle.com/code/dom3n1co/multiclas gpt2-pretrained*

Intermediate and History Models

- **FirstWithKeras:** *https://www.kaggle.com/code/dom3n1co/notebook63ce755794*
- **OnlyClassifierUNET:** *https://www.kaggle.com/code/dom3n1co/onlyclassifierunet*
- **OnlyClassifierResNet:** *https://www.kaggle.com/code/dom3n1co/onlyclassifierresnet*
- **AutoEncoder-Binary-GPT2:** *https://www.kaggle.com/code/dom3n1co/ae-binary-gpt2*
- **Binary-GPT2-Output:** *https://www.kaggle.com/code/fefe202/binarymalatonon-gpt2-5*
- **MultiClass-ZoneCategories-GPT2-Output:** *https://www.kaggle.com/code/fefe202/multiclasszoneca gpt2*
- **MultiClass-CustomTrans-Beams:** *https://www.kaggle.com/code/fefe202/transformer-beams*