Hybrid ConVIRT - Enhancing Medical Image-Text Representation Learning of Vision Language Models

by

Abhinav Sharma

A thesis submitted to the
School of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Masters of Science in **Computer Science**

Faculty of Science
Ontario Tech University
Oshawa, Ontario, Canada

The University of Ontario Institute of Technology requires the Certificate of Approval (CoA) to be included as page (ii) of the PRINTED version. Check the source file on how to add the provided CoA to your thesis.

Abstract

Challenges in medical imaging are addressed by advancing image-text representation learning, as exemplified by Hybrid-ConVIRT, which builds on contrastive learning frameworks such as ConVIRT and MedCLIP. Traditional medical imaging models rely on costly, expert-annotated datasets, limiting scalability. Transfer learning from general datasets, like ImageNet, often fails to capture the domain-specific nuances critical in clinical applications. The Hybrid model improves upon this by incorporating domain-specific transformations and using CXR-BERT-specialized as the text encoder, tailored specifically for chest X-ray reports. To optimize feature alignment, it introduces a 'Fused Similarity Matrix (FSM)' which potentially serves as a better target, it leverages the learned capabilities of MedCLIP and COnVIRT. Evaluation across four datasets (COVID, Tuberculosis, Pneumonia/TB Mix, and CheXpert) using linear probe and zero-shot classification demonstrates Hybrid-ConVIRT's enhanced discriminative power and feature robustness, particularly on COVID and CheXpert. While MedCLIP shows a slight advantage on the Tuberculosis dataset, it is likely due to the larger training set. The Hybrid model's performance supports the potential of combining learnings of both the prior models.

Keywords: Contrastive Learning, Image Transformation, Linear Classifier, Zero-Shot Classification, Image-Text Representation, CXR-BERT-Specialized, ViT, Feature Extraction, Multi-Class Classification

Dedication

To my parents, whose unwavering emotional support has been the foundation of my journey. To my friends Kriti, Suzi, Sameer, Bidushi, Daniel and Carl, for their steadfast encouragement and companionship during this time.

A special dedication to my high school friend Josh, now pursuing his postdoctoral work in machine learning at the University of Nijmegen in the Netherlands, whose insights and suggestions were invaluable to this work.

Thank you all for your belief in me and for your enduring support.

Acknowledgements

My gratitude goes to Professor Mehran Ebrahimi and Professor Heidar Davoudi for their invaluable guidance, support, and mentorship throughout this research. Their insights and encouragement were instrumental in shaping this work, and their expertise greatly enriched my understanding of the field.

Thanks are also due to Ontario Tech University's Computer Science group for granting access to the Imaging Lab and providing the necessary GPU resources. Special appreciation is extended to Richard Drake, the computing lab technician, for his assistance in securing access to multiple GPUs. His technical support was crucial to the completion of this project.

I am grateful as well to my colleagues in the lab, whose collaboration, idea exchanges, and brainstorming sessions fostered a supportive and intellectually stimulating environment, proving invaluable throughout this journey. Finally, heartfelt thanks go to the open-source community for their dedication to making research and datasets publicly available, a commitment foundational to the exploration and advancement of ideas in this work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Context

The field of medical imaging has traditionally relied on high-quality, expert-annotated datasets to train machine learning models. However, the substantial cost and limited scalability of obtaining these annotations have prompted researchers to explore alternative methods, such as leveraging transfer learning from natural image datasets like ImageNet. Although effective to a degree, these methods are often suboptimal for medical imaging due to the high inter-class similarity of medical images and the unique, subtle features necessary for clinical tasks.

Recent advances in contrastive representation learning, such as SimCLR [9], have shown promise in learning visual features without supervision by maximizing agreement between augmented views of the same image while minimizing agreement with others. However, these approaches typically fall short in medical imaging contexts as well because they do not account for the nuanced differences in medical images that are critical for accurate diagnosis and classification.

To address these challenges, ConVIRT (Contrastive Visual Representation Learning from Text) [50] builds a framework which introduces a novel approach by leveraging the

naturally occurring pairing of medical images and their corresponding textual reports. By integrating cross-modal data (images and text) in a contrastive learning framework, the aim was to develop models that can learn more effective and nuanced visual representations. This approach not only enhances the quality of in-domain medical image representations but also demonstrates superior performance across various medical specialties without requiring additional expert input. Through this research, they further advanced the field of self-supervised learning in medical imaging, leveraging both visual and textual data to overcome current limitations and improve clinical outcomes. This work was crucial in the area of contrastive learning with image-text pairs, as it was an inspiration for one of the revolutionary works in this domain: CLIP (Contrastive Language-Image Pretraining) [38], it is the backbone of DallE-2. Details on how CLIP works can be seen in section 1.1.

Another recent work, MedCLIP [48], has addressed the shortcomings of ConVIRT by developing a more robust approach to contrastive learning with medical images and texts in an unpaired manner. MedCLIP introduces a soft semantic matching loss that uses the medical semantic similarity between each image label and report as the supervision signal, enabling the model to capture subtle yet crucial medical meanings (see section 2.2 for more details).

Building on these foundations (ConVIRT and MedCLIP), our work explores three novel approaches:

(1) We perform contrastive learning for the Chest X-rays image-text pairs by jointly training the encoders for both modalities and introducing prompt engineering and ensembling methods during this training. We replace the Clinical BERT text encoder used in ConVIRT with CXR-BERT-specialized [28], which is specifically trained on chest X-ray text data. This substitution ensures that the base text encoder, initialized with CXR-BERT weights before training, is better aligned with the domain-specific language of chest X-rays rather than general clinical text used in Clinical BERT [3].

(2) We leverage the knowledge of MedCLIP's trained encoders, combining it with contrastive learning for joint image-text pairs from (1) to create a more robust model. These innovations enhance the model's ability to learn a more nuanced medical representations. For a brief overview of our motivation, see section 1.4, and for detailed methodology of our training, refer to sections 2.1 to 2.4.

(3) Finally, we chose to incorporate the image transformation in every image while training the models. More details can be found in Chapter 4.

1.2 Contrastive Language Image Pretraining (CLIP) -

A high-level architecture of how CLIP works is shown in the Figure 1.1 below. Images and their corresponding captions (text) are trained together in a pair-wise manner.

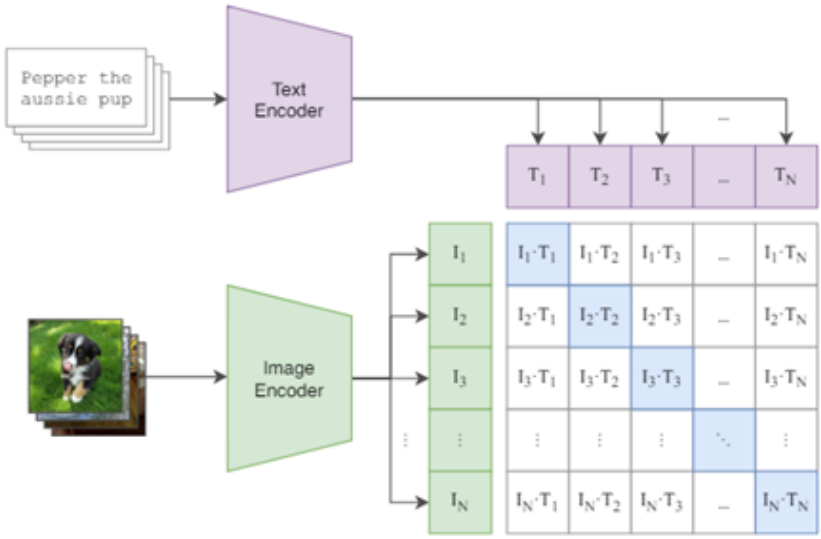


Figure 1.1: Contrastive Pre-training of CLIP

For the I_1 vector (Image 1 representation), the goal is to identify which text (T_1 - T_N) is most appropriate for that particular image. Image one (I_1) pairs with description one (T_1), and Image two (I_2) pairs with description two (T_2). We train the model such that

each image-description pair is maximally close to the correct one and minimally close to all incorrect ones. This approach is contrastive because it contrasts correct pairs (the diagonal elements in the similarity matrix) with incorrect pairs. [37]

We take the inner product of the image and text vectors to measure their similarity, maximizing the similarity for correct pairs and minimizing it for incorrect pairs. During training, this process pushes the diagonal elements of the similarity matrix towards 1, indicating high similarity, and the non-diagonal elements towards 0, indicating low similarity. This ensures that the model learns to distinguish between correctly and incorrectly paired images and descriptions effectively, enhancing its ability to perform tasks such as image classification, retrieval, and zero-shot learning.

1.3 MedCLIP addressing the False negatives pairs -

MedCLIP [48] identified and addressed the issue of false negatives pairs during the ConVIRT style of training by training their model in an unpaired manner. So the images and text are not being trained together as pairs. This way they get to leverage all the only-image and only-text datasets too. For the datasets that come with pairs like the ones used to train ConVIRT, they decouple the image-text pairs and collect all the texts together and all the images together and do the training in an unpaired manner.

They first obtain a semantic similarity (cosine similarity) between of every image label and every textual report, resulting in a semantic similarity matrix. This can be seen in the MedCLIP architecture in Figure 1.2 below (the left-hand side). The semantic similarity matrix serves as the target during training and is referred to as 'soft targets.' This matrix captures the semantic relationships between every image label and every text sample.

All the images and texts are then used to train an image encoder and a text encoder,

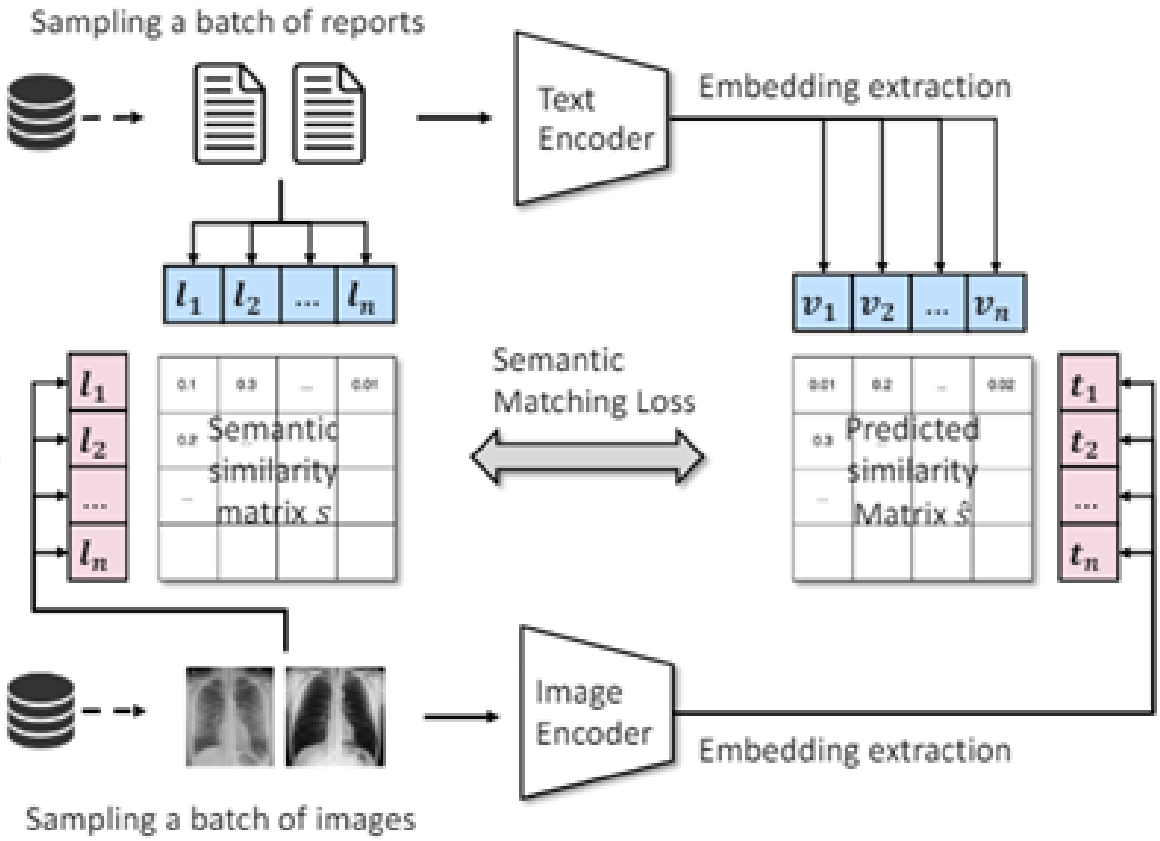


Figure 1.2: MedCLIP Architecture.

respectively. These encoders output the image and text embeddings, from which a similarity score is calculated using cosine similarity, resulting in a predicted similarity matrix (the right-hand side of Figure 1.2). These two encoders are trained to align this predicted similarity matrix towards the semantic similarity matrix through a semantic matching loss. Details of this is found in the MedCLIP paper.

1.4 Motivation and Contribution

While MedCLIP effectively addresses some issue present in the ConVIRT style of doing training (false negative pairs during training), there remains room for improvement in reinforcing the similarity of the true positive pairs. This work proposes a novel approach to fuse the Contrastive Similarity Matrix (CSM) (acquired from the ConVIRT

style training), with the Predicted Similarity Matrix (PSM) acquired from MedCLIP. By combining these matrices, the values of the cosine similarity for the true positive pairs will be higher, reinforcing the similarity scores of the true 'positive pairs', while still addressing the false negative pair issue (during the ConVIRT style training). This fusion aims to enhance the overall performance of the model, ensuring a more robust identification of relationships in medical images and texts.

Using CXR-BERT-specialized [28, 6] as the text encoder in the ConVIRT training framework is more beneficial than Clinical BERT [3] because CXR-BERT-specialized is pre-trained specifically on chest X-ray-related text. This specialization allows it to capture domain-specific language, terminology, and subtle nuances unique to chest radiology reports, leading to more accurate and contextually relevant text representations. In contrastive learning, where the alignment between image and text features is crucial, using CXR-BERT-specialized enhances the ability to learn meaningful relationships between chest X-ray images and their corresponding reports, resulting in improved performance and more robust medical image-text representations.

We also wanted to explore contrastive training on datasets that do not come with explicit radiology reports but instead have associated labels. Datasets like MIMIC-CXR and other radiology collections often require a lengthy and complex process to acquire the corresponding reports, which can be tedious, time-consuming and at times not possible to get. Additionally, access to such datasets is often restricted to individuals affiliated with research institutions, making it difficult for others to utilize them. To address this, I propose using contrastive training with multiple report templates where placeholders are dynamically filled with metadata from the datsets, such as patient information, imaging modality, and diagnostic labels etc. By generating synthetic but contextually relevant reports in this manner, anyone can effectively perform this style of training using readily accessible radiology image datasets with labels. Details of this can be seen in Chapter 4. This approach not only broadens the applicability of contrastive learning in scenarios

with limited or unavailable textual data but also makes advanced training methods more accessible to a wider range of researchers.

The implementation details are described below in the methodology section.

Chapter 2

Related Work

Vision-text representation learning has proven to be effective in capturing meaningful visual features. [44] demonstrated that incorporating caption annotations can lead to the learning of high-quality visual representations, highlighting the potential of using textual information to enhance image understanding in computer vision tasks . Building on this concept, more recent work by [4] introduced VLMo, a unified vision-language pre-training model employing a mixture-of-modality-experts approach, further advancing the integration of vision and language in representation learning . A widely adopted method for initializing encoders in medical imaging studies involves using weights pretrained on ImageNet, which serves as a benchmark dataset for various natural image categories. This technique exploits the advantages of transfer learning, where models initially trained on diverse visual content can be adapted for medical tasks. However, ImageNet's natural image features differ substantially from those found in medical images, which often contain intricate, subtle patterns that are critical for accurate clinical interpretation [41]. This contrast raises concerns about the effectiveness of such pretraining in capturing the domain-specific features necessary for medical image analysis, prompting the exploration of more tailored pretraining approaches suited to the medical field.

In the medical domain, methods like ConVIRT [50] and GLoRIA [21] have shown the

effectiveness of contrastive learning frameworks in improving medical image-text representation learning. ConVIRT, by aligning medical images with corresponding textual reports, demonstrated superior performance in learning visual representations without supervision. The ConVIRT paper highlights that most previous methods for learning visual features from paired text used a simple approach that treated it as a binary task, where the model tries to distinguish whether a text matches an image or not. In contrast, ConVIRT introduces a more advanced way of training that uses a new cross-modality objective called the Noise Contrastive Estimation (NCE), the loss function used. This approach, which considers a wider range of positive and negative pairs, leads to better learning of visual features, resulting in improved performance in capturing important details from images when paired with text.

While GLoRIA on the other hand extended this by leveraging global and local medical image-text alignments to enhance representation quality.

In [6] the authors show that focusing on better text understanding significantly improves how images and text can be linked. They introduce a new language model that performs exceptionally well with radiology texts and a method that combines image and text learning to achieve state-of-the-art results. They also provide a new dataset with detailed annotations to help further research in this field.

MedCLip [48] of course uses a slightly different approach from the others by doing a contrastive training in an unpaired manner addressing the issue of noise in the supervision during training due to the false negative pairs that could be present.

These studies highlight the increasing significance of contrastive and multi-modal learning approaches in medical imaging applications. Building on this, our work combines the strengths of ConVIRT with a new domain-specific text encoder, while also leveraging the learning from from MedCLIP. Below we shall look into the theoretical principles on which ConVIRT and MedCLIP are based off.

2.1 Background on ConVIRT

ConVIRT (Contrastive Visual Representation Learning from Text) [50] is a foundational model in the domain of medical image-text representation learning. It was developed to address the limitations of supervised learning in medical imaging, particularly the need for large, annotated datasets that are costly and time-consuming to create. ConVIRT leverages contrastive learning to bypass the reliance on manual annotations by utilizing the natural pairing of medical images (e.g., chest X-rays) with their corresponding textual reports (e.g., radiology reports) for representation learning.

In ConVIRT, the core idea is to train two encoders: one for images and one for text. During training, each image is paired with its corresponding medical report, forming a positive pair. The model's objective is to maximize the agreement (similarity) between the representations of the image and its corresponding text, while minimizing the agreement between non-paired image-text representations (negative pairs). This process is formalized using a *contrastive loss function*, specifically designed to push apart negative pairs and pull together positive pairs in the latent space.

2.1.1 Image and Text Encoders in ConVIRT

In ConVIRT, we start with paired inputs (x_v, x_u), where x_v represents one or more input images, and x_u is a text sequence that describes the imaging information (e.g., radiology reports). The original ConVIRT framework uses a convolutional neural network (CNN), such as ResNet50, as the image encoder f_v, initialized with ImageNet weights. This encoder transforms the input image x_v into a fixed-dimensional vector representation v_i. For textual reports, a language model like BERT, specifically ClinicalBERT [3] pre-trained on clinical text data, serves as the text encoder f_u. This encoder converts the text input x_u into a vector representation u_i of the same dimensionality as v_i. These vector representations, v_i and u_i, are then projected into a shared space using non-linear projection functions, allowing their similarity to be computed using cosine similarity.

2.1.2 Contrastive Loss and its Bidirectional Nature

The contrastive loss in ConVIRT aims to maximize the similarity between positive pairs, (x_v, x_u) (the image and corresponding text), while minimizing the similarity between negative pairs (an image and unrelated text or vice versa). This is achieved through the *InfoNCE loss* [32], where the loss function for an image-to-text positive pair (x_v, x_u) is defined as:

$$\mathcal{L}_i^{(v \to u)} = -\log \frac{\exp(\langle v_i, u_i \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle v_i, u_k \rangle / \tau)} \tag{2.1}$$

Here:

- v_i is the image representation vector after transformation $\tilde{x}_v$ of x_v, encoded by the image encoder.
- u_i is the text representation vector after transformation $\tilde{x}_u$ of x_u, encoded by the text encoder.
- $\langle v_i, u_i \rangle$ denotes the cosine similarity between the vectors v_i and u_i, which is calculated by:

$$\langle \mathbf{v}, \mathbf{u} \rangle = \frac{\mathbf{v}^\top \mathbf{u}}{\|\mathbf{v}\| \|\mathbf{u}\|} \tag{2.2}$$

- τ is a temperature scaling parameter that controls the sharpness of the distribution.
- N is the number of samples in the minibatch.

To ensure robust learning, ConVIRT also defines a text-to-image contrastive loss, where the text representation retrieves the corresponding image:

$$\mathcal{L}_i^{(u \to v)} = -\log \frac{\exp(\langle u_i, v_i \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle u_i, v_k \rangle / \tau)} \tag{2.3}$$

This bidirectional contrastive loss setup captures both the image-to-text and text-to-image relationships, allowing the model to learn rich, mutually aligned representations across modalities.

The final objective is a weighted combination of both losses over all positive pairs in the minibatch, ensuring the learned representations maximize similarity for true pairs while minimizing it for all other pairs.

Weight Updates through the Loss Function

After the loss is calculated, the weights of the image and text encoders are updated through backpropagation. Here's a step-by-step explanation of how this process works:

1. Forward Pass: The image x_v is passed through the image encoder to obtain v_i, and the text x_u is passed through the text encoder to obtain u_i. Cosine similarity is then calculated between the image and text embeddings.

2. Loss Calculation: The contrastive loss is calculated using the formulas above. The loss function tries to maximize the similarity between positive pairs (image and corresponding text) and minimize the similarity between negative pairs (image and incorrect texts).

3. Backpropagation: The gradients of the loss with respect to the weights of the encoders are computed using the chain rule. This provides the information on how much each weight in the encoder should change to reduce the loss.

4. Weight Updates: The weights of the image encoder and text encoder are updated using an optimizer (e.g., Adam). The optimizer adjusts the weights in such a way that: - The similarity between the true positive pairs (image and its corresponding report) is increased (pushing it towards 1). - The similarity between negative pairs (image and incorrect reports) is decreased (pushing it towards 0).

Pushing Similarities Closer to 1 for Positive Pairs and Closer to 0 for Negative Pairs

The contrastive loss structure in this model maximizes similarity for positive pairs while minimizing it for negative pairs:

- Maximizing Positive Pair Similarity: The numerator in the loss function contains the exponential of the positive pair's similarity:

$$\exp(\langle v_i, u_i \rangle / \tau)$$

The model increases this value by maximizing the cosine similarity $\langle v_i, u_i \rangle$, which happens when the embeddings v_i and u_i are close in the vector space, pushing the similarity towards 1.

- Minimizing Negative Pair Similarity: The denominator contains the sum of exponentials over all pairs (positive and negative):

$$\sum_{k=1}^{N} \exp(\langle v_i, u_k \rangle / \tau)$$

To minimize the loss, the model reduces the similarities between negative pairs v_i and u_k (for $k \neq i$), effectively pushing these similarities towards 0.

The softmax function, defined as:

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{N} \exp(z_j)} \tag{2.4}$$

where z_i represents the input score for a particular class, with N being the total number of classes, converts these scores into a probability distribution. This transformation allows each similarity score to be interpreted as the probability of a particular class, ensuring the probabilities sum to 1 [18, 5].

In equation 3.3, the softmax function normalizes the similarity scores for the positive pair (u_i, v_i) and all other pairs (u_i, v_k) in the batch, emphasizing the relative similarity of the true positive pair over the negative pairs.

- Numerator: The term $\exp(\langle u_i, v_i \rangle / \tau)$ calculates the exponential of the scaled similarity score for the positive pair (u_i, v_i), where $\langle u_i, v_i \rangle$ is the cosine similarity, and τ is a temperature parameter controlling distribution sharpness. This exponentiated value emphasizes the positive pair's score.

- Denominator: The term $\sum_{k=1}^{N} \exp(\langle u_i, v_k \rangle / \tau)$ sums the exponentials of all similarity scores (positive and negative) for u_i with each v_k in the batch, acting as a normalization term to create a probability distribution over all pairs.

By applying softmax, the model assigns higher probabilities to pairs with higher similarity scores (e.g., the true positive pair) and lower probabilities to less similar pairs (negative pairs). The loss function $\mathcal{L}_i^{(u \to v)}$ thus maximizes the log probability of the true positive pair while minimizing it for negative pairs, effectively guiding the encoders to learn representations that clearly distinguish positive from negative pairs.

Final Training Loss

The final training loss is calculated as a weighted combination of the image-to-text contrastive loss and text-to-image contrastive loss, averaged over all positive image-text pairs in a mini-batch:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \left(\lambda \mathcal{L}_i^{(v \to u)} + (1 - \lambda) \mathcal{L}_i^{(u \to v)} \right) \tag{2.5}$$

Here, $\lambda \in [0, 1]$ is a scalar weight that controls the balance between the two losses. This weighted average ensures that both directions (image-to-text and text-to-image) contribute to the final loss, making the model robust in capturing relationships in both modalities.

2.1.3 Impact of ConVIRT and it's shortcomings

ConVIRT made significant advancements in contrastive learning for medical imaging by leveraging the inherent pairing of medical images and their reports to learn effective visual representations without the need for costly manual annotations. However, a significant issue arises in the way the contrastive loss treats non-paired image-text samples as negative pairs during training.

In medical datasets, it is possible for a text report that was not paired with a specific image during training to still accurately describe the contents of that image. For example, two different patients might present with the same medical condition (e.g., pneumonia), and their radiology reports could be very similar. Ideally, the model should assign a higher similarity score to such image-text pairs. However, because of the way contrastive loss works in ConVIRT, all non-paired text reports are treated as negative samples, which means the similarity between such pairs is penalized rather than encouraged. This introduces noise into the supervision during training, as semantically similar but unpaired image-text pairs are incorrectly pushed apart in the embedding space.

MedCLIP and other later works have pointed out this issue, noting that the contrastive loss in ConVIRT does not account for semantic similarity across different image-text pairs that might be equally accurate descriptions. As a result, when the ConVIRT model is trained and used to generate similarity scores between images and reports (what we refer to as the Contrastive Similarity Matrix (CSM)), the model will show high similarity scores for true positive pairs (correct image-report matches) and low similarity scores for true negative pairs (incorrect matches). However, for certain image-text pairs where the text accurately describes the image but was not paired with it during training, the model will still output low similarity scores due to the way these pairs were treated as negatives during training.

This limitation highlights how the ConVIRT training paradigm introduces unnecessary penalization for accurate, non-paired descriptions, thus reducing the effectiveness of

the learned representations. Although ConVIRT does an excellent job of learning positive pair relationships, its tendency to push apart false negatives (semantically similar but unpaired samples) limits its potential. This issue laid the groundwork for models like MedCLIP, which introduced a semantic similarity-based approach to address the false negative problem. MedCLIP, by using a soft matching loss that leverages semantic similarity, ensures that image-text pairs with high semantic overlap are not penalized, even if they were not paired together during training.

Therefore, while ConVIRT demonstrated the power of contrastive learning in medical image-text representation tasks such as image classification and retrieval, its inability to handle semantically correct, non-paired descriptions emphasizes the need for more refined methods like those introduced by MedCLIP.

2.2 MedCLIP's training procedure and the loss

MedCLIP addresses key limitations in traditional contrastive learning for medical images and texts by introducing innovative methods that extend beyond paired data. Unlike earlier approaches that rely solely on paired image-report datasets, MedCLIP decouples images and text, enabling the use of vast unpaired datasets, thereby significantly increasing the amount of training data available. Additionally, MedCLIP tackles the issue of false negatives, where semantically similar but unpaired image-text pairs are incorrectly treated as negatives. By implementing a soft semantic matching loss based on medical knowledge, MedCLIP allows the model to recognize semantic similarities between unpaired data, capturing subtle but important clinical meanings. This approach enhances the model's ability to understand complex medical relationships while making better use of available data.

Semantic similarity between images and text is computed using multi-hot entity vectors extracted from the data via MetaMap. These vectors represent the semantics of

images l_{img} and text l_{txt}, and their cosine similarity defines the semantic similarity matrix (SSM). The soft targets for image-to-text matching are computed using a softmax function over these similarities:

$$y_{ij}^{v \to u} = \frac{\exp(s_{ij})}{\sum_{j=1}^{N_{batch}} \exp(s_{ij})} \tag{2.6}$$

where s_{ij} is the semantic similarity between image i and text j. Text-to-image soft targets are computed similarly. These soft targets guide the training of two encoders, one for images and one for text, which generate the predicted similarity matrix (PSM).

The semantic matching loss minimizes the difference between the PSM and SSM using a cross-entropy loss. For image-to-text, the loss is defined as:

$$\mathcal{L}^{v \to u} = -\frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} \sum_{j=1}^{N_{batch}} y_{ij}^{v \to u} \log \hat{y}_{ij} \tag{2.7}$$

where $\hat{y}_{ij}$ is the predicted similarity between image i and text j. The text-to-image loss is computed similarly, and the final training loss is the average of both:

$$\mathcal{L} = \frac{\mathcal{L}^{v \to u} + \mathcal{L}^{u \to v}}{2} \tag{2.8}$$

By pushing the predicted similarities toward the soft targets, MedCLIP learns to recognize semantically similar, unpaired image-text pairs, resulting in more robust and nuanced representations for medical data.

2.2.1 Difference in the Loss functions and the outcomes

The key difference between the losses used in ConVIRT and MedCLIP lies in how they approach the learning process, especially in handling false negatives. ConVIRT's contrastive loss does not account for this semantic overlap between the image labels and the textual reports, leading to the issue of false negatives—text reports that accurately describe the medical condition in the image but are treated as negative samples.

MedCLIP introducing a semantic matching loss, which uses soft targets based on the semantic similarity between image labels and text. This approach allows the model to capture subtle but meaningful relationships between image-text pairs, even if they were not explicitly paired during training. As a result, MedCLIP is able to handle semantically similar pairs more effectively, avoiding the penalization of false negatives and leading to a more robust model.

Let suppose we train a model with a specific dataset in the ConVIRT way and generate a similarity matrix (CSM) and use the same dataset to inference through the MedCLIP encoders and acquire the PSM. When comparing the CSM from ConVIRT and the PSM from MedCLIP (as discussed in Section 3.4 of the methodology and visualized in Figure **??**), a clear difference in similarity values emerges. In ConVIRT, the contrastive loss tends to push false negatives too far apart, leading to lower similarity scores for semantically related pairs in the CSM. However, the true positive pairs in ConVIRT still show higher similarity values compared to MedCLIP, as the contrastive loss is more focused on maximizing the similarity of these exact matches.

While MedCLIP's PSM explicitly learns to output high similarity scores for semantically related pairs through its soft target supervision, the similarity scores for true positive pairs in MedCLIP's PSM are still not as high as those produced by ConVIRT for its true positive pairs.

2.3 Theory on the Zero-Shot Classification

2.3.1 Explaining Zero-shot Classification with the CLIP example

To perform zero-shot classification on a specific dataset to test a pre-trained model, we will follow the approach similar to that used by the authors of CLIP.

Embedding Class Descriptions: To utilize CLIP for zero-shot classification, they start

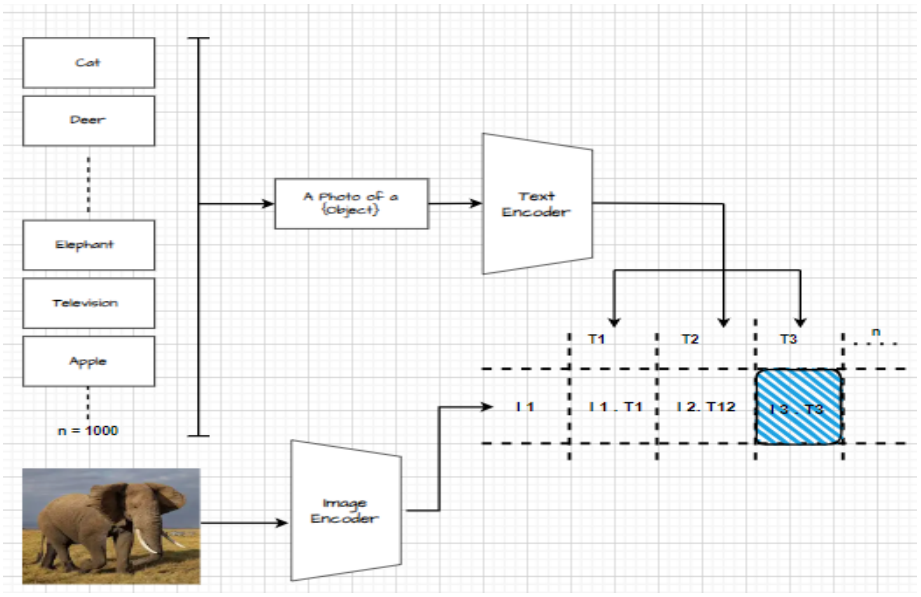


Figure 2.1: Zero-Shot Classification Process with the CLIP encoders

by embedding the descriptions of each class from the target dataset, which CLIP has not seen before, in an "image + label" format (a one or two-word label describing the image). If the target dataset is ImageNet with 1000 classes, CLIP has not seen these images, we want to generate text prompts like "a photo of a {object}" for each class. Examples include "a photo of an Elephant" or "a photo of a deer." The reason we do this is because we need to generate prompts from labels when testing the CLIP model because its text encoders are trained on more than just one or two-word labels. This is called prompt engineering - the efficacy of using "A photo of a {object}" rather than just the class name as the default prompt. This approach allows for better context and understanding of the image. CLIP is trained on descriptive phrases or sentences, which provide more context and semantic meaning. Using prompts like "a photo of a {object}" ensures that the text input aligns with the model's training data, enabling it to generate more accurate and meaningful embeddings for classification. So they use the text encoder of CLIP to convert these 1000 prompts into embeddings, resulting in a distinct embedding for each class. Look at Figure 2.1 as for reference.

Embedding the Image: Next, take the image you want to classify, such as a photo

of a dog, and embed it using CLIP's image encoder. This process transforms the image into a vector representation in the same embedding space as the text prompts, with the dimensions.

Computing Similarities: After obtaining the embeddings, compute the similarity between the image embedding and each of the 1000 class embeddings. This is done by taking the dot product of the image embedding with each text embedding. CLIP is designed so that both images and text share the same embedding space, and the dot product measures their similarity.

Predicting a class either with or without a Softmax Classfier: The class corresponding to the highest dot product value is considered the predicted label. For instance, if the dot product between the image embedding and the embedding for "a photo of an Elephant" is the highest, the model predicts that the image is a Elephant, this can be seen in Figure 2.1. To turn CLIP into a true classifier, you can apply a softmax function to the dot product values. This converts them into predicted probabilities for each class, providing a probabilistic interpretation of the classification results.

2.3.2 Prompt engineering and Ensembling during Evaluation

The CLIP paper shows the substantial benefits of using detailed prompts over simple class names to enhance context and comprehension in image classification. By utilizing prompts such as "A photo of a {class name}" instead of merely the label, the model acquires better contextual information, leading to improved performance. Tailored prompts to suit specific datasets also proves advantageous. Taking the Oxford-IIIT pet dataset as an example, employing "A photo of a {class name}, a type of pet", likewise "a type of food" for Food101 dataset, and "a type of aircraft" for the FGVC-Aircraft dataset resulted in better outcomes. [37]

Moreover, the paper examines the technique of prompt ensembling, where various prompts are averaged to create text embeddings. For example, merging prompts like "A photo of a very big {object}" and "A photo of a large {object}" showed performance

enhancements. In the case of ImageNet, using an ensemble of 80 prompts achieved a 3.5% performance increase, and when combined with prompt engineering, an overall gain of about 5% was observed.

CLIP, trained on a large dataset of 400 million image-text pairs from the internet, has demonstrated its effectiveness, particularly in zero-shot classification tasks across various datasets. However, its performance declined on some datasets like ObjectPose. According to [1], CLIP's performance on the ObjectPose dataset was evaluated using three different models (ViT-B/16, ResNet-50, and ResNet-101). These models were tested with three different prompt types: a single general prompt ("a photo of a {class name}"), 80 general ImageNet prompts for ensembling, and 12 customized prompts for ObjectPose that included terms like "flipped", "rotated", "upside-down", etc. The findings indicate that the ObjectPose-customized prompts outperformed both the single general prompt and the ensemble of 80 general prompts, suggesting that CLIP benefits from specific information about the object's rotation provided in the customized prompts.

In contrast, for the ObjectPose +-10 dataset, where objects are upright, the customized prompts were less effective and sometimes misleading, while the general prompts performed better. This underscores the importance of prompt engineering and ensembling in enhancing model performance across different datasets and conditions, emphasizing the necessity to tailor prompts to the specific characteristics of the dataset for optimal results [1].

It is clear that the effects of prompt ensembling and engineering is of great significance in these models. The type of prompt engineering and ensembling one would want to do during evaluation depends on the type of dataset you want to evaluate. At times a handful of customized prompts work well for a given dataset, like the 12 used for ObjectPose yields better results. Other times, when there is less variation in the object poses within the dataset, numerous general prompts could yield a better result.

2.3.3 Zero-Shot Performance and the effects of prompts ensembling

Now we will look at the effects of prompts ensembling on MedCLIP and compare the results before we move forward with the testing on the Hybrid-ConVIRT. Lets take the Covid dataset as an example for the testing, it has two classes; Pnumonia and normal.

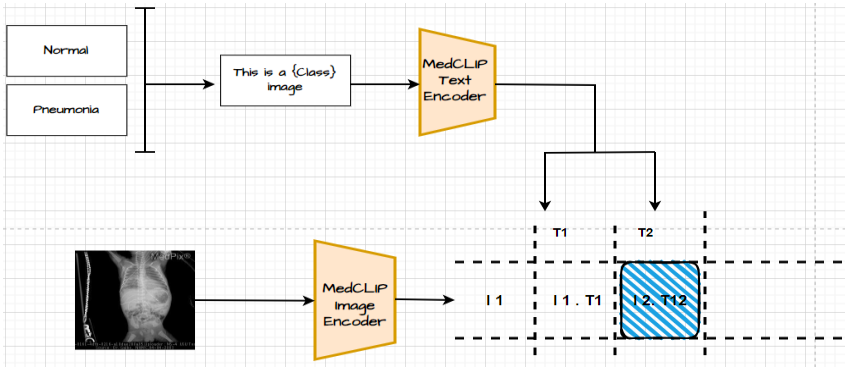


Figure 2.2: Zero-Shot Classification on the RSNA Dataset with MedCLIP Encoders without Prompt Ensembling

From Figure 2.2 we can see this Zero-shot classification process on the Covid Dataset. 'This is a {Class} image' is used to extract the two text embeddings for the two labels in this dataset. Then the cosine similarity of every image from the dataset will be computed with the two extracted embeddings, T_1 and T_2 and the similarity with the highest score will be class representing the image. So in Figure 2.2 it would be T_2.

Our zero-shot testing on binary datasets (COVID, Tuberculosis, RSNA) yielded suboptimal results due to the use of short prompts for generating text embeddings. Our encoders were trained with enriched prompts that have a larger token size, and during testing, the similarity scores were calculated between the images and the shorter prompts. This mismatch caused the model to generate less interpretable embeddings for classification.

To address this, we adopted an ensemble approach using eight longer prompts instead of just two short ones. For each class (positive and negative), we generated four different

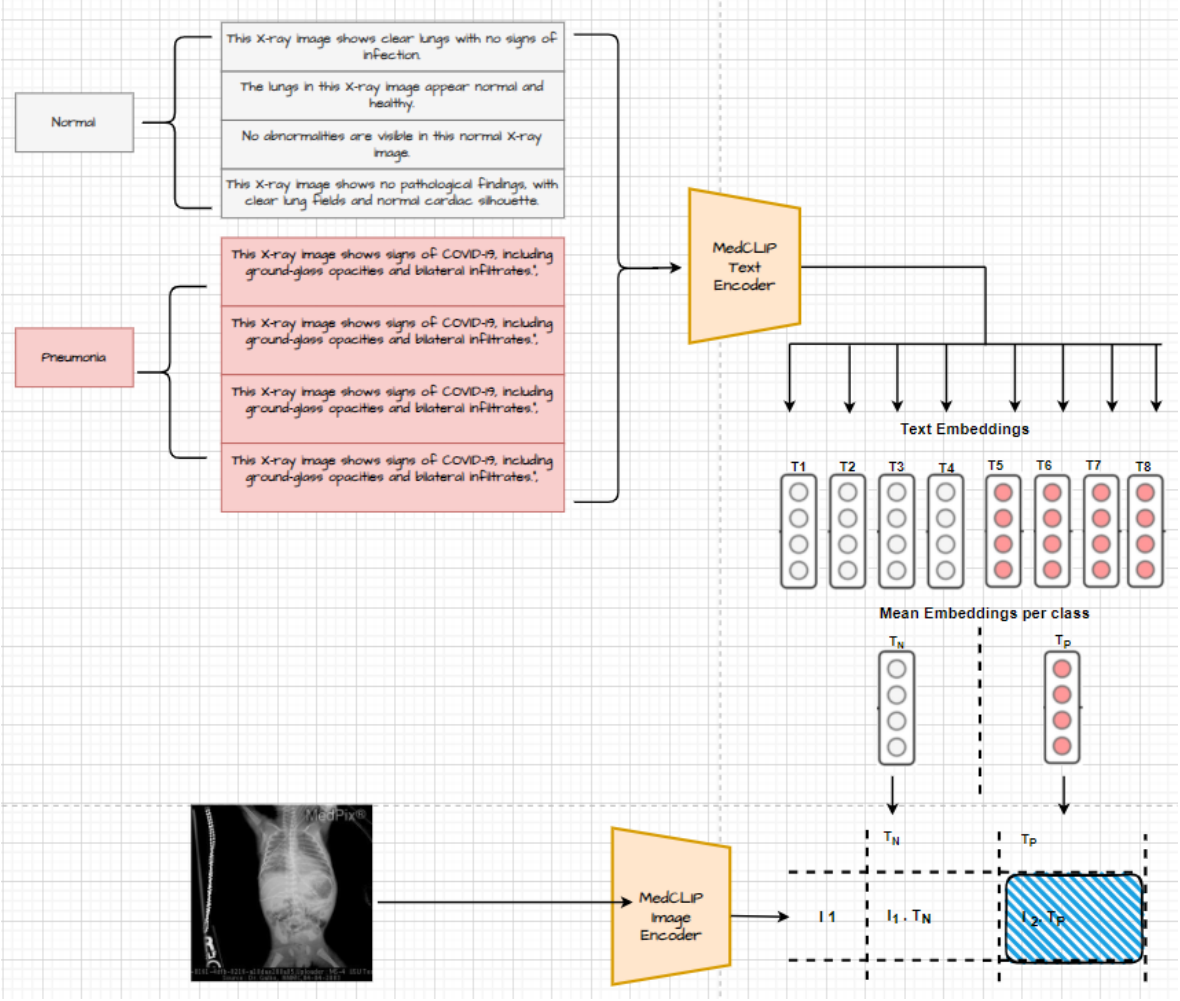


Figure 2.3: Zero-Shot Classification on the RSNA Dataset with MedCLIP Encoders with Prompt Ensembling

prompts, resulting in a total of eight prompts. We then calculated the embeddings for these prompts and averaged the four embeddings per class to obtain a mean embedding for each class, denoted as T_p (Mean Text embedding for the 'pneumonia' class) and T_n (Mean Text embedding for the 'normal' class), as illustrated in Figure 2.3.

During inference, the image is processed through the image encoder to compute its embedding, which is then compared to the mean embeddings T_p and T_n from Figure 2.3. The class with the highest similarity score is assigned to the image. As shown in Table 1, this prompt ensembling method significantly improved the classification results. We can see from these results that there is a clear significant difference in scores when using

prompt ensembling during zero-shot evaluation on datasets. Therefore, when evaluating the Hybrid-ConVIRT and the baseline models from this point onwards, we will be using prompt ensembling.

2.3.4 The Linear Probing Process

In the linear probing process, the frozen image encoder from a base model (either MedCLIP, ConVIRT, CLIP or others) outputs feature embeddings for each input image. These embeddings are passed to the linear classifier, which is trained using supervised learning on a labeled dataset specific to the new task. This linear classifier essentially learns to map the frozen features to the target labels of the new task. By only training the weights of this simple linear layer, we can quantify the usefulness of the representations learned by the pre-trained encoder. If the frozen encoder produces meaningful feature embeddings, the linear classifier will be able to perform well on the new task with minimal additional training.

The advantage of this approach is twofold. First, it provides a computationally efficient way to evaluate pre-trained models. Since the image encoder remains frozen, training time is significantly reduced, as we only need to adjust the parameters of the small linear layer. Second, it highlights the effectiveness of the learned representations without requiring any further updates or modifications to the pre-trained model itself [7] [36]. This makes linear probing particularly valuable when working with large, pre-trained models like the image encoder of models like MedCLIP, ConVIRT, CLIP and others, where fine-tuning the entire model can be resource-intensive.

Chapter 3

Methodology

The methodology for training the Hybrid-ConVIRT model, as seen in Figure 3.1 can be divided into three major procedures:

1. **ConVIRT-Style Training on CheXpert**: First, we perform a ConVIRT-style training on 100,000 images from the CheXpert dataset and their generated captions. After completing the training, we generate a *Contrastive Similarity Matrix* (CSM) per batch using the same dataset. This matrix captures the model's learned ability to separate true positive pairs from negative pairs in the embedding space, effectively reflecting the full potential of the model's learned embeddings.

2. **Inference through Pre-trained MedCLIP**: Next, we use the same 100,000 images (which MedCLIP was also trained on) and perform inference through the pre-trained MedCLIP model, utilizing its respective image and text encoders, to generate a *Predicted Similarity Matrix* (PSM) per batch.

3. **Fusion and Training of Hybrid-ConVIRT**: Finally, we fuse the two matrices to generate a *Fused Similarity Matrix* (FSM), which is used as the target for training Hybrid-ConVIRT. During training, we incorporate image transformations for every image. We employ a Vision Transformer (ViT) with ImageNet initialized weights

as the base image encoder and a pre-trained, specialized CXR-BERT as the text encoder.

3.1 Acquiring the Contrastive Similarity Matrix (CSM)

To obtain a contrastive similarity matrix, we need to train a image/text encoder in a manner similar to how the original CLIP/ConVIRT models were trained. This requires true captions (a sentence or two describing the diagnosis of the X-Ray. Like the description in a report) for each image so that the encoders can be trained in a paired-wise manner (every image has an associated caption which describes that image). The goal is to ensure that positive pairs are close together and negative pairs are further apart in the embedding space.

Ideally it would have been beneficial to do to use the pre-trained model described in the ConVIRT paper to generate the CSM but since the authors did not make the weights of their model public, the ConVIRT style training would need to be done again.

3.1.1 Contrastive Training with Prompt Engineering and Ensembling

Crafting high-quality contextual prompts has been shown to improve the performance of CLIP and other vision-language models. Not just engineering the prompts but using a combination of prompts for the same image has proven to be effective [25, 49].

Many publically available chest X-ray datasets come with one or multi-word labels. There are more chest X-ray datasets that provide labels without accompanying radiology reports than those that include full reports. These labeled datasets—such as the NIH ChestX-ray14 [46], CheXPert [23], the RSNA Pneumonia Detection Challenge Dataset [39], CoVID [10] and Tuberculosis [42] — offer annotations indicating the presence or

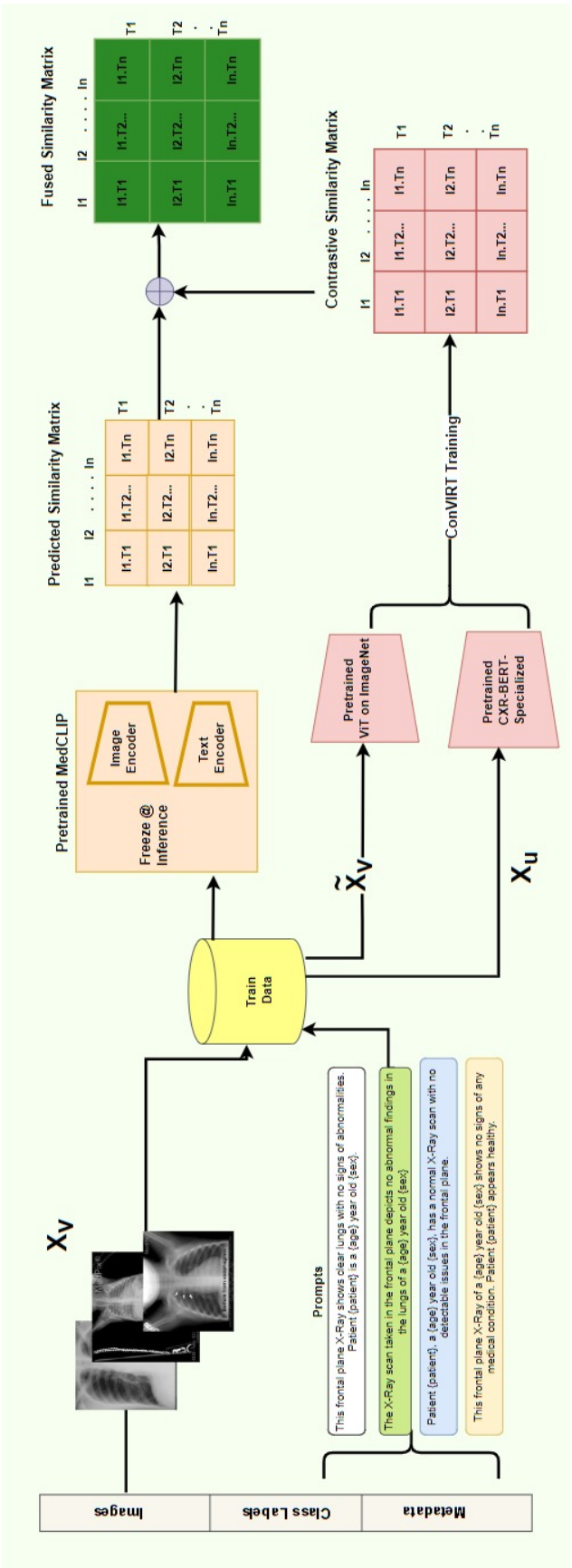


Figure 3.1: Hybrid-ConVIRT architecture: The PSM acquired from the pretrained MedCLIP on a dataset gets fused with the CSM with ConVIRT training to get a Final FSM. In the ConVIRT training (encoder in Pink), the image encoder, ViT, is initialized with ImageNet weights and the text encoder, a BERT model, is initialized with CXR-BERT-specialized.

absence of specific conditions or abnormalities. Though they are invaluable for training machine learning models focused on classification and detection tasks, these can also be used for a constrastive learning framework where captions (sentences from reports) are required. Multiple Caption templates can be used with placeholders as we've done below and hence datasets as such can be leveraged for contrastive learning as well.

The following 14 classes are from the Chexpert dataset which we are using for trianing: No Finding, Enlarged Cardiomediastinum, Cardiomegaly, Lung Opacity, Lung Lesion, Edema, Consolidation, Pneumonia, Atelectasis, Pneumothorax, Pleural Effusion, Pleural Other, and Fracture. Additionally, these datasets include other metadata like the path to the image, sex, age, frontal/lateral orientation, and AP/PA view.

To perform contrastive training with prompt ensembling, we will create four general prompts that include placeholders such as {age}, {sex}, and {get_positive_classes} to represent the metadata and positive classes associated with each image. These placeholders are designed to capture the unique attributes and medical findings of each X-ray. For each image, we will fill in these placeholders with the relevant metadata, such as the patient's age and sex, as well as the specific medical conditions detected in the scan. This approach will generate four distinct text samples per image, each tailored to reflect the detailed characteristics of the X-ray, thereby enhancing the training process through diverse and contextually relevant textual descriptions.

The following are the four customized prompt templates that will be filled with the corresponding metadata and positive classes:

General Templates -

1. This {frontal_lateral_plane} X-Ray in the {ap_pa} orientation reveals {get_positive_classes} in Patient {patient}, a {age} year old {sex}.

2. The X-Ray scan taken in the {frontal_lateral_plane} plane and {ap_pa} orientation shows {get_positive_classes} in a {age} year old {sex}.

3. Patient {patient}, a {age} year old {sex}, presents with {get_positive_classes} as seen in this {frontal_lateral_plane} X-Ray in the {ap_pa} orientation.

4. The {frontal_lateral_plane} X-Ray image in the {ap_pa} orientation displays {get_positive_classes} in a {age} year old {sex}, Patient {patient}.

The training will be conducted in a paired-wise manner, similar to the original CLIP model (Figure 3.1) and ConVIRT, which is designed for chest X-rays and their corresponding text samples. The novelty in our approach is the use of prompt ensembling during training. Each image will be associated with four customized prompts, resulting in four "true" image-text pairs per image as opposed to one "true" image-text pairs from previous methods. Consequently, we will feed the same image four times through the image encoder, paired each time with one of the four associated text samples through the text encoder.

We will employ a bi-directional contrastive loss function, similar to ConVIRT style of training. The loss is explained in detail in section 3.1.2 With ConVIRT, the visual representations were enhanced by increasing the alignment between correct image-text pairs and random pairs using a bidirectional contrastive objective across image and text modalities [50].

After the model is trained, we will use the same image-text pairs on the trained encoders and inference through them to acquire the image and text embeddings for each pair. This process can be seen in Figure 3.2 We will then calculate the cosine similarities to generate a contrastive similarity matrix. In this matrix, the diagonal elements should represent the 'true positive pairs,' with similarity values higher than those of the non-diagonal elements, as illustrated in Figure 1.1.

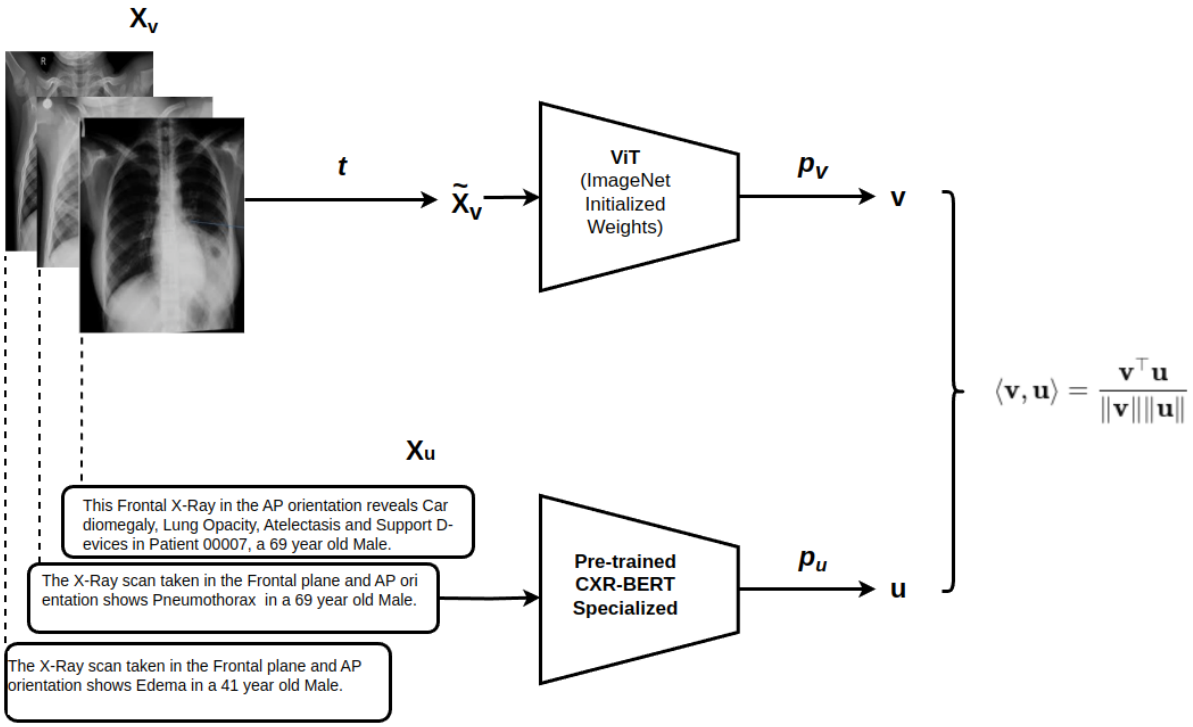


Figure 3.2: Training ConVIRT with CXR-BERT Speacilized

3.1.2 Image Transformation and ConVIRT stlyle of training Training

In Section 3, we described how the original ConVIRT model was trained using a ResNet-50 architecture initialized with ImageNet weights for the image encoder and a BERT model initialized with ClinicalBERT weights for the text encoder. ClinicalBERT is a variant of BERT that has been trained on a diverse corpus of clinical text, providing it with a broad understanding of medical language.

Since the introduction of Vision Transformers (ViT) in October 2020, [14] they have gained significant popularity in the computer vision community due to their ability to model long-range dependencies in images effectively. However, the authors of ConVIRT, Zhang et al., were already well advanced in their research by that time and did not have the opportunity to experiment with ViT in their work.

Subsequent studies after 2021 have explored the use of ViT as image encoders in sim-

ilar architectures. For instance, MedCLIP employs ViT for image encoding to leverage its superior performance in capturing image representations and even though by margin it outperforms its' restnet50 model. In our ConVIRT-style training, we adopt ViT initialized with ImageNet weights as our base image encoder to harness these advancements in transformer-based image modeling.

Additionally, instead of initializing our BERT model with ClinicalBERT weights, we chose to use CXR-BERT Specialized [28]. This model is a BERT variant explicitly trained on chest X-ray reports and other relevant text associated with chest radio-graphs. By utilizing CXR-BERT Specialized, we aim to enhance the text encoder's ability to comprehend domain-specific terminology and nuances inherent in chest X-ray interpretations. The architecture of this training procedure can be seen in Figure 3.2 above.

Image Transformation

For every X-ray image X_v, we have an associated textual report X_u. During training, we process them as paired inputs to learn meaningful cross-modal representations. The image X_v undergoes a series of transformations to produce $\hat{X}_v$. This transformed image is created by applying one or multiple transformations, including contrast enhancement, random affine transformations, horizontal flipping, color jittering, and normalization. These augmentations introduce variability in the training data, which helps the model become more resilient to changes in the input and focus on the salient features relevant for diagnosis.

By enhancing the, we aim to improve the visibility of anatomical structures and pathological findings in the X-ray images. The random transformations simulate real-world variations in image acquisition, such as different patient positions or imaging conditions. Lastly, the normalization step ensures that the input images have a consistent scale and distribution, which is crucial for effective training of the Vision Transformer initialized with ImageNet weights.

For the text data X_u, we utilize a tokenizer compatible with CXR-BERT Specialized, which is pre-trained on chest X-ray reports and related clinical text. The tokenization process includes padding and truncation to maintain uniform input lengths, facilitating efficient batch processing during training.

By jointly training on the transformed images $\hat{X}_v$ and their corresponding texts X_u, the model learns to associate visual features with textual descriptions. This approach leverages the strengths of both modalities and aims to improve performance on downstream tasks such as image-text retrieval and report generation.

Below the image transformation process has been explained in detailed. A series of augmentations are applied to every image to each X-ray image X_v to produce the transformed image $\hat{X}_v$. Pytorch library's torch.transforms module was imported to use all of the functions defined below.

- **Resize - Applied to every image**
  - **Function**: `transforms.Resize((224, 224))`
  - **Purpose**: Ensures all images have uniform dimensions suitable for the input requirements of the Vision Transformer (ViT) model.
- **RandomAffine Transformation - Applied to every image with randomly sampled parameters. The randomly sampled parameters are explained below**
  - **Function**: `transforms.RandomAffine(degrees=10, translate=(0.1, 0.1), scale=(0.9, 1.1))`
  - **Parameters**:
    - **Rotation**: Random rotation within $\pm 10°$.
    - **Translation**: Random horizontal and vertical shifts up to 10% of the image size.

    - ∗ **Scaling**: Random scaling between 90% and 110% of the original size.
  - **Purpose**: Introduces variability in orientation, position, and scale to enhance model robustness to such variations.

- **RandomHorizontalFlip - Applied with a 50% probability.**
  - **Function**: `transforms.RandomHorizontalFlip()`
  - **Purpose**: Augments the dataset by flipping images horizontally, helping the model generalize to different left-right orientations.

- **ColorJitter - Applied to every image with random adjustments.**
  - **Function**: `transforms.ColorJitter(brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1)`
  - **Parameters**:
    - ∗ **Brightness Adjustment**: Up to ±10%.
    - ∗ **Contrast Adjustment**: Up to ±10%.
    - ∗ **Saturation Adjustment**: Up to ±10%.
    - ∗ **Hue Adjustment**: Up to ±10%.
  - **Purpose**: Simulates variations in imaging conditions, making the model more robust to changes in lighting and color properties.

- **RandomRotation - Applied to every image with a random rotation angle**
  - **Function**: `transforms.RandomRotation(degrees=15)`
  - **Parameters**:
    - ∗ **Rotation**: Random rotation within ±15°.
  - **Purpose**: Provides additional rotational variance, helping the model to be invariant to slight rotational differences in the images.

- **Conversion to Tensor - Applied to every image**
  - **Function**: `transforms.ToTensor()`
  - **Purpose**: Converts the PIL image to a PyTorch tensor, normalizing pixel values to the [0, 1] range.

- **Normalization - Applied to every image**
  - **Function**: `transforms.Normalize(mean, std)`
  - **Parameters**:
    - **Mean**: $[0.5015, 0.5015, 0.5015]$
    - **Standard Deviation**: $[0.2907, 0.2907, 0.2907]$
  - **Purpose**: Normalizes the image tensor using the mean and standard deviation of the 100,000 images of the Chexpert dataset. First, centering the data by subtracting the mean shifts the average value of the pixels in each channel to zero, ensuring that the data is centered around zero. Second, scaling the data by dividing by the standard deviation adjusts the data so that it has unit variance, making the range of pixel values consistent across different images and channels. This combination of centering and scaling facilitates convergence during training, as normalized data leads to faster and more stable convergence; when input features (in this case, pixel values) are on a similar scale, it helps the optimizer make more consistent updates to the model weights.

Transformation Vizualization

Lets look at how the transformations are happening for the purpose of visualization.

Image 1: RandomAffine [Angle 2.79°, Translations $(-21, -10)$, Scale 0.94, Shear 0.0]; RandomHorizontalFlip: Not Applied; ColorJitter [Brightness 1.03, Contrast 1.08, Saturation 0.9174, Hue -0.016]; RandomRotation: $-14.11°$.

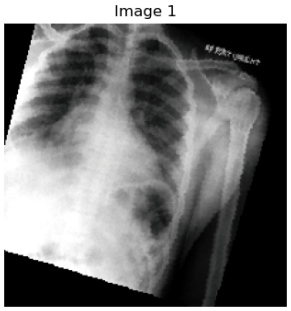


Image 2: RandomAffine [Angle $-5.63°$, Translations $(0, -21)$, Scale 0.94, Shear 0.0]; RandomHorizontalFlip: Not Applied; ColorJitter [Brightness 1.01, Contrast 0.94, Saturation 1.0179, Hue 0.062]; RandomRotation: $-14.81°$.

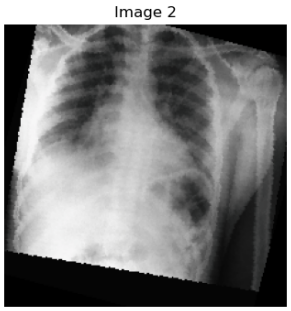


Image 3: RandomAffine [Angle 6.12°, Translations $(9, -7)$, Scale 0.93, Shear 0.0]; RandomHorizontalFlip: Not Applied; ColorJitter [Brightness 0.97, Contrast 0.92, Saturation 0.92, Hue 0.07]; RandomRotation: 3.11°.

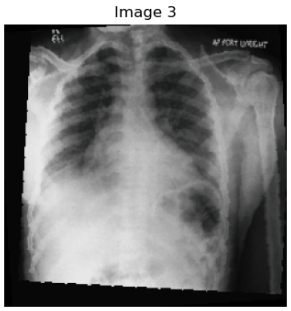


Image 4: RandomAffine [Angle 6.14°, Translations $(10, 2)$, Scale 1.09, Shear 0.0]; RandomHorizontalFlip: Applied; ColorJitter [Brightness 1.01, Contrast 1.07, Saturation 1.02, Hue 0.07]; RandomRotation: 2.32°.

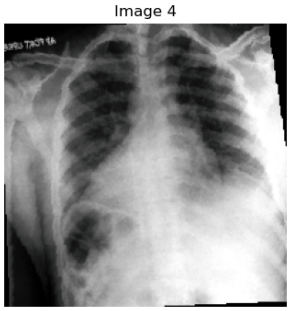
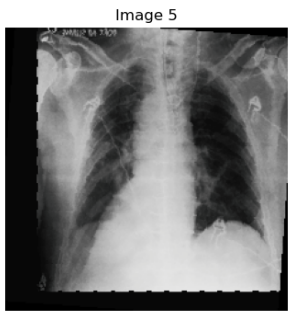


Image 5: RandomAffine [Angle 4.09°, Translations $(-20, -12)$, Scale 0.96, Shear 0.0]; RandomHorizontalFlip: Applied; ColorJitter [Brightness 0.95, Contrast 0.92, Saturation 0.96, Hue 0.027]; RandomRotation: $-4.01°$.



3.2 Acquiring the PSM through the Pre-trained MedCLIP

We will use the same image-text pairs employed to acquire the Contrastive Similarity Matrix (CSM) and perform inference through the image and text encoders of MedCLIP. Due to the semantic matching loss used to train the MedCLIP encoders, these pre-trained encoders output the image and text embeddings in such a way that the embeddings of the 'true' image-text pairs will be closer together in the embedding space. Additionally, the embeddings of pairs where the text does not necessarily belong to the patient's image but still accurately describes the image (the false negative cases from contrastive learning) will also be closer together in the embedding space. This occurs because during the training of MedCLIP, the encoders were trained to output embeddings of image-text pairs based on the semantic similarity between the reports (text samples) and the label of the image. The MedCLIP encoders were trained towards the semantic similarity matrix, which served as the 'target' during training [48]. This results in higher similarity scores for the 'true pairs' as well as for the pairs that were considered false negatives in the CSM.

Next, we will calculate the cosine similarity between every image embedding and every text embedding, resulting in a Predicted Similarity Matrix (PSM).

3.3 Fused Similarity Matrix (FSM)

The CSM should show higher values for 'positive pairs', improving upon the Predicted Similarity Matrix (PSM) we obtained previously - which will not show as high similarity scores for the 'positive pairs'. The way the contrastive loss works (Bidirectional contrastive objective), the positive pairs get pushed towards 1 (resulting in high values close to 1) and the negative pairs get pushed towards 0 (resulting in low values closer to 0).

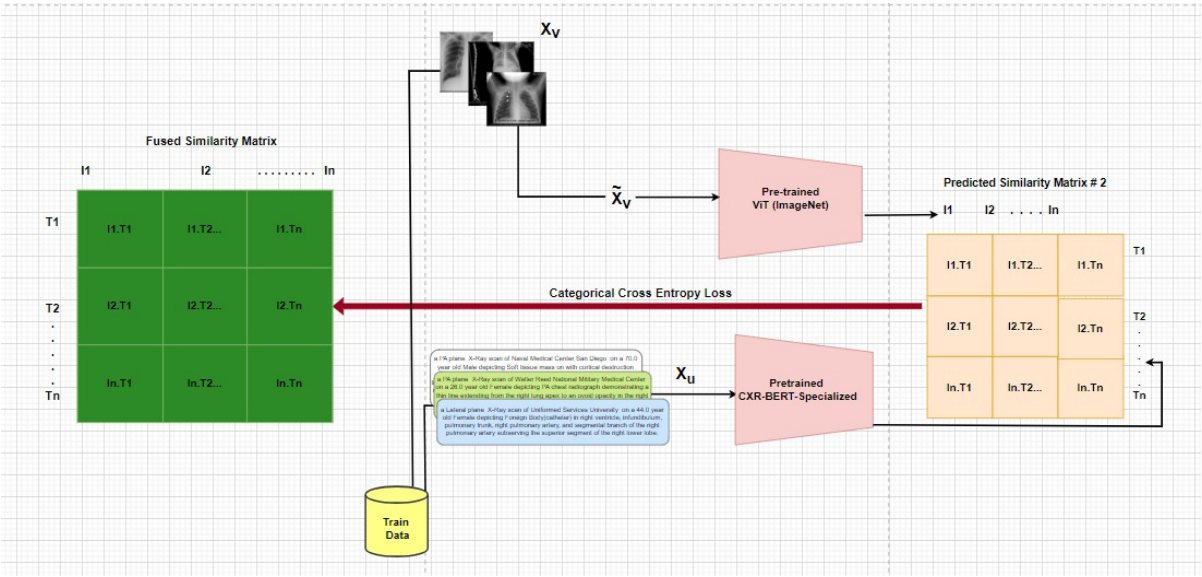


Figure 3.3: Training Encoders towards the FSM

As a result, captions that accurately describe the X-rays of different patients will also receive lower similarity scores, which is not desirable. This is the false negative issue that studies like ConVIRT [50] and Gloria [21] encountered.

To address this, we will combine the two matrices, the PSM and CSM. This fusion aims to balance the similarity scores, reinforcing the true positive similarity scores from the PSM (by incorporating the scores from the CSM) while still mitigating the impact of false negatives (which PSM already does). Details of this fusion can be in section 4.4 and a highlevel architecture of Hybrid-ConVIRT can be seen in Figure 3.1 and Figure 3.3.

After the fused matrix is ready, it will act as a target so we can train an image and a text encoder towards it. Similar to how MedCLIP was trained, now instead of using the semantic similarity matrix (targer) of MedCLIP, we will use the fused matrix to train these encoders towards it. The same set of images and text that were used to get the fused matrix will now be used to get a predicted similarity #2 (PSM#2) and with a cross entropy loss we will train the encoders. The architecture of this can be seen in Figure 3.3.

3.4 The method of fusion with the Addition Operation

Lets examine these Matrices in Figure 5 which are only for the purpose of illustration.

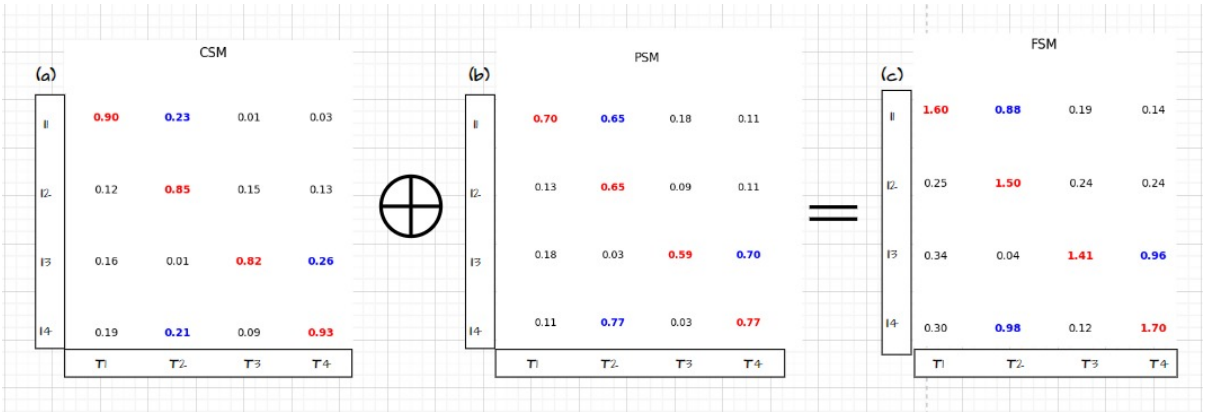


Figure 3.4: Addition of CSM and PSM to get the FSM. Scores in Red represent the 'true positive' pairs, 'flase negative' pairs in blue and 'true negative' pairs in Black.

After the ConVIRT style training is complete, we use five X-ray images and their corresponding captions to generate the CSM (Figure 5-a). The similarity scores for the true positive pairs (highlighted in red) in this matrix will be very high due to the nature of the contrastive objective, which pulls true positive pairs together in the embedding space and pushes negative pairs further apart. Consequently, we observe high similarity scores for true positives. In contrast, the negative pairs, including false negatives, will exhibit very low similarity scores; this includes cases where the image-text similarity score is low despite the text accurately describing the image of another patient (the false negative cases highlighted in blue).

Applying the same five images and texts to MedCLIP generates similarity scores akin to those in the PSM (Figure 3.4-b). The similarity scores for the true positive pairs in the PSM wouldn't be as high as those in the CSM due to the semantic matching loss employed in MedCLIP as opposed to the contrastive loss. For example, the similarity score for I1 and T1 is 0.70 in the PSM compared to 0.9 in the CSM. However, the PSM

shows improvement in addressing false negatives (highlighted in blue). For instance, the similarity score for I1 and T2 is 0.65 in the PSM versus a low 0.23 in the CSM. Thus, the PSM addresses false negatives while maintaining relatively high similarity scores for positive pairs, although these scores are not as high as those in the CSM. Similarly for the true negative case, the CSM shows the scores are very low, which is ideal but in MedCLIP's PSM the scores are indeed low but not as low as we'd like them to be. This observation suggests that a fusion approach might offer improvements.

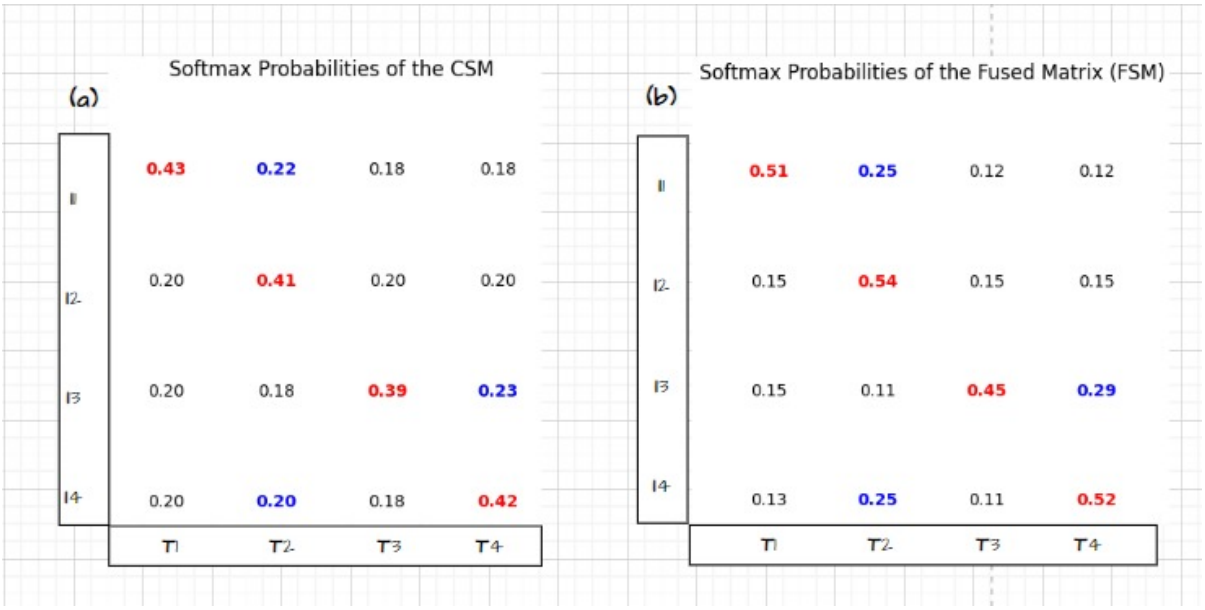


Figure 3.5: Normalzied CSM and FSM compared

To compare, let us examine the softmax probabilities of the two matrices (CSM and FSM) after addition operation (CSM+PSM =FSM). For the true positive pair (I1 and T1, highlighted in red), the softmax FSM (Figure 3.5-(b)) yields a higher probability than the softmax CSM (Figure 3.5-(a)) and we know it would be higher than that of the PSM too as we saw in Figure 3.4 (b) the scores for the true positive pairs (in red) were high but much lower than that from the CSM (Figure 3.4 (a)). In the case of false negatives (highlighted in blue), the softmax normalized FSM probabilities are higher than that of the CSM as we see, this is because of the addition operation, the higher scores for the flase negative cases from the PSM has been incorporated in the FSM. Specifically, for I1

- T2, the softmax FSM yields a higher probability of 0.25 as opposed to 0.22. Similarly, for I3 - T4 we get 0.29 which is greater than 0.23 and lastly for I4-T2 we have 0.25 which is greater than 0.20. So for the pairs that had a low similarity scores (false negatives pairs from ConVIRT, text describing an image belonging to another patient) giving a lower softmax probability (blue scores in Figure 3.5-(a)), now have a higher probability (blue scores in Figure 3.5-(b) to be training towards).

For the true negative cases (highlighted in black), the probability scores from the FSM are consistently lower than those from both the CSM and PSM across all 'true negative' instances, which is whats desirable.

On a larger scale, when tested on an entire dataset, it remains to be seen whether encoders trained on the FSM will yield optimal results; however, the theory suggests that this will indeed be the case, as the Matrix which we're trainig our encoders towards is more robust than that used in MedCLIP and ConVIRT style of training.

Chapter 4

Experiments

4.1 Datasets and Baslines

The training of the Hybrid-ConVIRT model is conducted in two distinct stages. First, a ConVIRT model is trained on a selected dataset, which is then used to infer through the trained image and text encoders to generate the Contrastive Similarity Matrix (CSM). This same dataset is used to infer through the MedCLIP model, generating the Predicted Similarity Matrix (PSM). In the second stage, the primary training of the Hybrid-ConVIRT model takes place, where an image encoder (Vision Transformer (ViT) initialized with ImageNet weights) and a text encoder (CXR-Specialized BERT) are trained towards the Final Similarity Matrix (FSM), which is the combination of the CSM and PSM.

For this process, we utilized 100,000 images from the CheXpert dataset [23], which contains 14 classes. The corresponding reports for these images were generated using the metadata as described in Section 4.1.

For evaluation purposes, we employed four distinct datasets listed and described below.

COVID-19 Dataset: This is a binary classification dataset curated by a team from Qatar University, the University of Dhaka, and collaborators. The dataset comprises chest X-ray images, divided into COVID-19 positive cases, normal cases, and viral pneumonia images [45]. The second release of this dataset, used in this evaluation, contains:

- 3,616 COVID-19 positive cases
- 10,192 normal cases

The dataset includes contributions from various sources:

- 2,473 COVID-19 images from the Padchest dataset [8]
- 183 images from a German medical school [16]
- 559 images from SIRM, ML work-group GitHub [24] [30]
- 400 images from another GitHub source [11]

The normal images include:

- 8,851 images from the RSNA dataset [40]
- 1,341 images from Kaggle

Tuberculosis Dataset: This is another binary classification dataset, comprising 700 publicly accessible TB images and an additional 2,800 TB images available through the NIAID TB portal. These images were collected in collaboration with medical professionals from Hamad Medical Corporation and Bangladesh. The dataset also includes 3,500 normal images.

This dataset is associated with the publication: Rahman et al., 'Reliable Tuberculosis Detection using Chest X-ray with Deep Learning, Segmentation and Visualization' IEEE Access, 2020 [43].

Pneumonia/TB Mix Dataset: This multi-class dataset contains pneumonia, normal, and tuberculosis images. The pneumonia and normal images are sourced from the RSNA dataset, while the tuberculosis images are obtained from the NIAID TB portal [39, 31].

CheXpert Dataset (with 5 Classes): For the CheXpert dataset, we selected 9,347 images belonging to the following five classes for evaluation [23]:

- Atelectasis
- Cardiomegaly
- Edema
- Pleural Effusion
- Consolidation

This setup enables a comprehensive evaluation of the Hybrid-ConVIRT model across both binary and multi-class classification tasks.

4.1.1 Baslines

CLIP - Contrastive Language–Image Pretraining is a powerful model designed to learn visual representations from natural language supervision. It was trained on 400 million image-text pairs collected from the internet, but it was not trained on any medical images. CLIP includes two options for the image encoder: Vision Transformer (ViT) and ResNet50, while the text encoder is a BERT-based language model. The training process uses an InfoNCE contrastive loss, optimizing the model to match corresponding image-text pairs while pushing apart non-matching pairs. Despite its large-scale training, CLIP's performance may be suboptimal for specialized tasks like medical image interpretation due to the absence of domain-specific training data [38].

MedCLIP - MedCLIP was developed to address the need for domain-specific models in medical imaging. It is exclusively trained on medical datasets, including CheXpert and MIMIC-CXR, using an unpaired approach to align chest X-rays with radiological reports. The model uses either a ResNet50 or Vision Transformer (ViT) as its image encoder, with the text encoder initialized with BioClinical-BERT, a variant of BERT optimized for clinical language. MedCLIP is trained using a semantic matching loss, which encourages the model to align the representations of chest X-rays with the corresponding text descriptions of the findings. This focus on medical data makes MedCLIP more suitable for medical tasks compared to CLIP [48].

BioVIL - BioViL is another model specifically designed for medical imaging, using a contrastive learning approach with a contrastive loss similar to ConVIRT. It was trained using the MS-CXR dataset, which contains chest X-ray images paired with bounding box annotations and corresponding clinical findings they got from MiMIC-CXR. Each image in the dataset is annotated with multiple sentences describing different pathologies, and the bounding boxes highlight the corresponding regions in the images [29].

By comparing these baselines, it becomes evident that CLIP, although versatile and powerful, lacks the domain-specific training required for medical imaging tasks. On the other hand, both MedCLIP and BioViL were trained exclusively on medical datasets, with BioViL focusing on detailed medical annotations and bounding box information. This specialization provides MedCLIP and BioViL with a significant advantage in tasks involving chest X-ray interpretation and clinical text alignment. Our model, Hybrid-ConVIRT will be compared against these three baselines evaluated on multiple datasets listed above.

4.1.2 Pre-processing of the training data and the datasets used for evaluation

Exploratory Data Analysis of CheXpert

The ChexPert dataset contains 223414 images and meta data. We will not be using all of this data for training. The table below presents the class distribution for various chest x-ray related conditions as recorded in a CheXpert dataset.

Table 4.1: **Class Distribution of Medical Conditions**

Label	**'-1' Values**	**'1' Values**	**'0' Values**	**'NAN' Values**
No Finding	0	22381	0	201033
Enlarged Cardiomediastinum	12403	10798	21638	178575
Cardiomegaly	8087	27000	11116	177211
Lung Opacity	5598	105581	6599	105636
Lung Lesion	1488	9186	1270	211470
Edema	12984	52246	20726	137458
Consolidation	27742	14783	28097	152792
Pneumonia	18770	6039	2799	159806
Atelectasis	33739	33376	1328	154971
Pneumothorax	3145	19448	56341	144480
Pleural Effusion	11628	86187	35396	96023
Pleural Other	2653	3523	316	216922
Fracture	642	9040	2512	211220
Support Devices	1079	116001	6137	100197

Each row represents a specific condition, while the columns represent four different categories:

- **'-1' Values**: Represents uncertainty of the condition being present or not.
- **'1' Values**: Positive instances where the condition is present.
- **'0' Values**: Neutral or unclear instances.
- **'NAN' Values**: Missing or null values, indicating no mention of the observation by the labeler in the report.

The missing values (**nanVal**) mean that there was no mention of the observation being extracted by the labeler in the report. These cases are generally treated as negative in the model presented in the paper, meaning they do not belong to that specific class. [2]

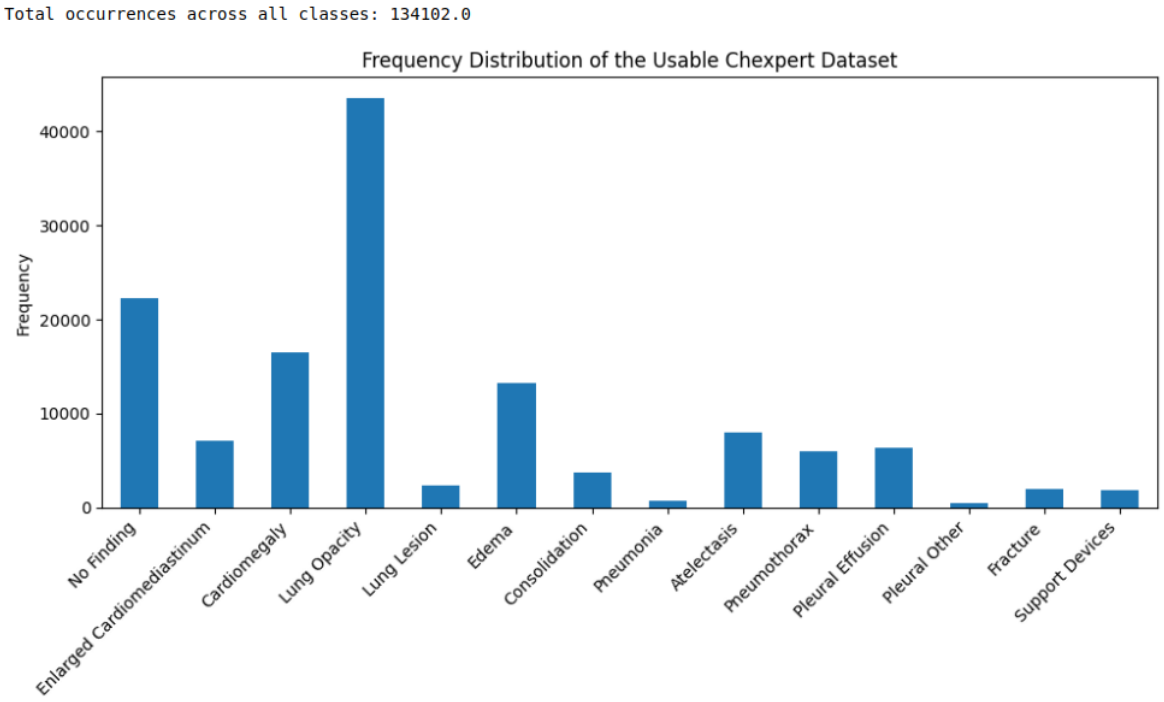


Figure 4.1: Frequency chart of the 14 classes in the CheXpert's trainable dataset

The **minusOneVal** represents uncertainty about whether the class is present or not. In some cases, the condition may be present, while in others, it is not. To avoid confusing the model and adding noise to the supervision, the approach in this dataset omits all rows with negative values and converts all the **nanVal** entries to zero. This strategy ensures clearer supervision during training by treating uncertain or missing values consistently.

We create a mask to filter out any rows that have `-1` values across the specified columns, ensuring that only rows with confident annotations are kept:

```
mask = (full_train_copy[cols] != -1).all(axis=1)
```

The resulting DataFrame contains data without uncertain labels, providing cleaner and more reliable training data for the model. The final step prints the number of

rows in the cleaned dataset. This approach maintains a substantial amount of data while ensuring the model is not exposed to uncertain or ambiguous annotations. The frequency class distribution from the resulting dataframe can be seen in Figure 4.1 above.

Handling Imbalanced Datasets with SMOTE

To address the class imbalance in the datasets used for evaluating the linear probe, we employ the Synthetic Minority Over-sampling Technique (SMOTE). This is applied across all datasets during the training of the linear probe during evaluation, which include the CoVID, Tuberculosis, Pneumonia, and CheXpert datasets. SMOTE is an established method used to mitigate the issue of classifiers being biased toward the majority class by generating synthetic samples for the minority class. This ensures that the classifier, in this case a linear probe, is trained in a balanced manner across all classes, thus preventing the model from leaning disproportionately towards the majority class [26].

In our experiments, SMOTE is integrated during the 5-fold cross-validation steps in the evaluation phase. After splitting the dataset into 5 sets, 4 sets are used for training and 1 set is reserved for validation. During the training process, SMOTE generates synthetic data for the underrepresented classes by creating new embeddings that are similar to the minority class examples. For instance, in the CoVID dataset, there are 10,192 normal images compared to only 3,616 COVID images. When the training set, comprising 4 out of the 5 sets, is used for training, SMOTE is applied to generate synthetic COVID embeddings to balance the dataset. This step ensures that the model does not overfit to the normal class due to the significant class imbalance. The same process is repeated for each of the datasets during their respective cross-validation training. This is an example of SMOTE increasing the training set size of the linear probe for ‘Fold 1’ during evaluation - Train size before SMOTE: 11046, Train size after SMOTE: 16308, Test size: 2762. The details of before and after size of the training data of all folds acrross all datasets can be seen in the appendix.

This technique is crucial because imbalanced datasets, such as those used in our evaluations, pose significant challenges to machine learning models, leading to poor generalization for minority classes. Without addressing this imbalance, the model would be prone to predict the majority class more often, neglecting the minority class, which can lead to suboptimal performance, especially in real-world scenarios where minority classes, such as disease cases, are of high importance.

Below is a summary of the class distributions for the evaluation datasets:

- **CoVID Dataset**:
  - Normal: 10,192
  - COVID: 3,616
- **Tuberculosis Dataset**:
  - Normal: 3,500
  - Tuberculosis: 700
- **Pneumonia/TB Mixed Dataset**:
  - Normal: 10,206
  - Pneumonia: 5,775
  - Tuberculosis: 1,489
- **CheXpert Test Set (5 Classes)**:
  - Atelectasis: 1,543
  - Cardiomegaly: 2,960
  - Consolidation: 700
  - Edema: 3,192

– Pleural Effusion: 952

In each case, SMOTE is applied to the training data to ensure that minority classes such as COVID, Tuberculosis, or specific medical conditions in CheXpert are adequately represented during training. By generating synthetic samples, SMOTE enables the classifier to better learn from the minority class, which is crucial for improving the overall performance of the model on imbalanced datasets.

4.2 Model Training and Evaluation

4.2.1 Hardware and Environment Setup

For the training of the Hybrid-ConVIRT model and the subsequent experiments, I utilized several key libraries and frameworks. The primary libraries for data management included `os`, `numpy`, and `pandas`, which were essential for handling file operations, numerical computations, and data manipulation. For the model development and training, I relied on `torch` [33] along with its submodules `torch.nn` and `torch.optim` to define the architecture and optimize the model. Batch processing and dataset handling were done using the `torch.utils.data` library, where custom `Dataset` and `DataLoader` classes were created.

For image processing, I employed the Python Imaging Library (`PIL`), which allowed me to manipulate and load image data. To evaluate the model's performance, I utilized a variety of metrics from the `sklearn.metrics` library [34], such as `accuracy_score`, `precision_score`, `recall_score`, `f1_score`, and `confusion_matrix`. Additionally, I used `StratifiedKFold` from `sklearn.model_selection` to perform cross-validation on the datasets. To visualize the results, including ROC curves and confusion matrices, I used `matplotlib.pyplot` [22], which provided a versatile platform for plotting.

Addressing class imbalance in the datasets was a key challenge, and to tackle this, I applied the Synthetic Minority Over-sampling Technique (SMOTE) [26] from the

`imblearn.over_sampling` library. This method helped in ensuring a balanced representation of classes during training.

Several pre-trained models were also integrated into the project. The CLIP model was sourced from the `transformers` library using `CLIPProcessor` and `CLIPModel` [38]. For the MedCLIP component, I cloned the MedCLIP repository andI utilized `MedCLIPProcessor`, `MedCLIPModel`, and `MedCLIPVisionModelViT` [48, 47] classes, which provided the necessary processing capabilities. Additionally, I integrated the BioViL model's image encoder from the `ImageModel` class, which I imported from Microsoft's `health_multimodal` GitHub repository [29].

The computational power required for this project was substantial, and I used two GPUs for all model training and experimentation: an `RTX 2080 Ti` and a `Titan V`. These GPUs help accelerate the training process, especially when handling the large datasets (100,000 images and text for the training of Hybrid-ConVIRT).

4.2.2 Hyper-parameter Tuning and Training

Hyperparameter tuning plays a critical role in optimizing model performance by identifying the best set of parameters that minimize the loss function while maintaining generalization. In this work, Optuna, an optimization framework designed for automated hyperparameter tuning, was selected for both ConVIRT and Hybrid-ConVIRT model training. Optuna is preferred over traditional hyperparameter tuning methods such as grid search or random search because of its flexibility, efficiency, and adaptive nature. Unlike grid search, which exhaustively evaluates all possible parameter combinations, or random search, which evaluates random combinations, Optuna uses a more intelligent approach based on Bayesian optimization. This allows it to explore the search space more effectively by focusing on promising regions, reducing computational overhead, and potentially achieving better results with fewer trials [19, 12].

ConVIRT Model's Tuning and Training

For the ConVIRT model, the text encoder was initialized using CXR-BERT specialized, the BERT language model specialized for chest X-ray related tasks. The hyperparameters selected for tuning included the batch size, learning rate, and the number of epochs. Using Optuna, the number of trials was set to 40, based on the size of the dataset, which consisted of 100,000 samples. The total time required for the tuning process spanned 96 hours.

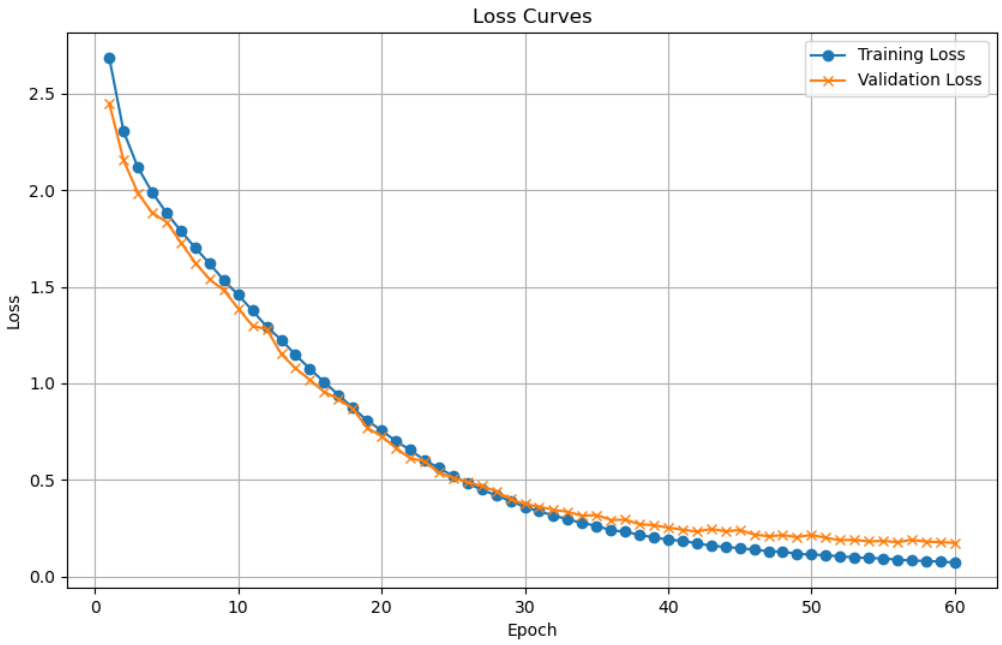


Figure 4.2: Training and Validation Loss Curve of ConVIRT

After examining the tuning results, the optimal hyperparameters were identified: a learning rate of 1.034×10^{-5} and a batch size of 16. The tuning results and logs are documented in the Appendix, under the section "Hyperparameter Tuning of ConVIRT." The ConVIRT model was then trained for 60 epochs, and the training and validation loss curves were closely monitored. As we can see from Figure 4.3 the validation loss started to rise slightly above the training loss after 30 epochs but did not indicate significant

overfitting, as the general trend of the validation loss was still decreasing. Therefore, the training was finalized at 55 epochs, as the validation loss plateaued after this point. This indicates that 55 epochs was the optimal cutoff for minimizing loss without overfitting, ensuring the model's generalization capability.

Hybrid-ConVIRT Model Tuning and Training

For the Hybrid-ConVIRT model, a similar hyperparameter tuning process was followed using Optuna. However, in this case, the number of epochs was excluded from the tuning parameters, as it was deemed unnecessary for initial tuning. The model was monitored for up to 60 epochs, allowing for observation of the loss trends. The hyperparameters chosen for tuning included the learning rate, batch size, and temperature. The temperature parameter, essential in contrastive learning, influences the scale of the logits in the softmax function, and thus plays a crucial role in controlling the sharpness of the probability distribution.

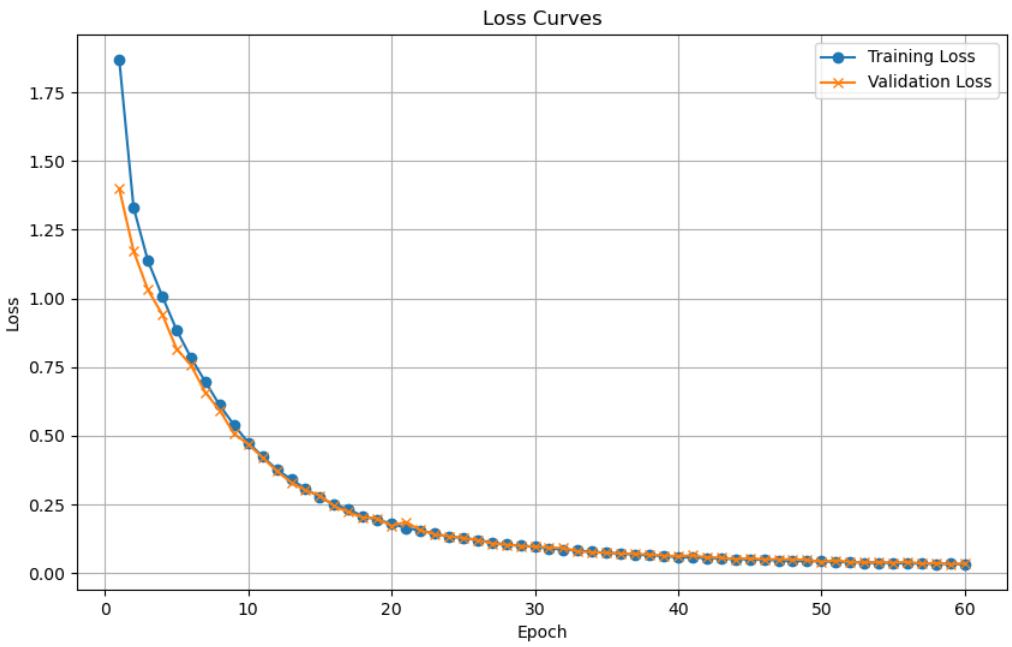


Figure 4.3: Training and Validation Loss Curve of ConVIRT

The tuning process for the Hybrid-ConVIRT model took approximately 165 hours. Upon reviewing the logs, the optimal hyper-parameters were found to be a temperature of 0.0578 and a learning rate of 1.7×10^{-5}, yielding the lowest loss of 0.239. These hyper-parameters were then used for training the image and text encoders towards the Final Similarity Matrix (FSM). Model checkpoints were saved every 5 epochs, starting from epoch 30. The training and validation loss curves were analyzed, and it was observed that the validation loss, along with the training loss, decreased simultaneously until around the 50th epoch, after which it began to plateau. Based on this analysis, the checkpoint from the 50th epoch was selected for testing, as it provided the most stable performance without further reduction in loss after that point. Refer to Figure 4.3.

The complete hyper-parameter tuning log for the Hybrid-ConVIRT model is detailed in the Appendix under "Hyper-parameter Tuning log of Hybrid-ConVIRT." This systematic approach to hyper-parameter optimization ensured that both the ConVIRT and Hybrid-ConVIRT models achieved optimal performance while minimizing the risk of over-fitting and under-fitting.

4.2.3 Evaluating Hybrid-ConVIRT with a Linear Probe

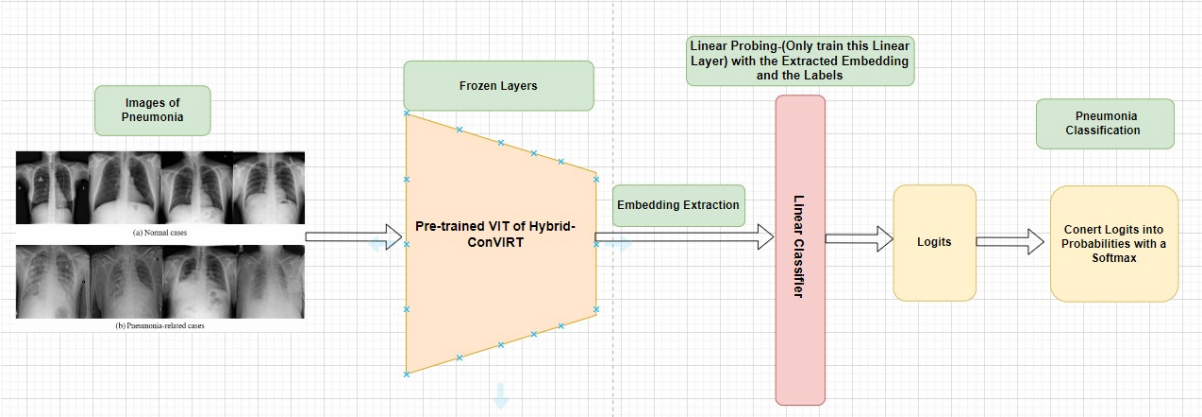


Figure 4.4: Architecture of Linear Probing done on the pre-trained Hybrid-ConVIRT with the RSNA dataset

The training of the linear layer, also known as "linear probing," [7] is a crucial step in evaluating the quality of the representations learned by a pre-trained model. In this

method, the base model (in our case, the image encoder from Hybrid-ConVIRT) is kept frozen, meaning that its weights are not updated during the training process. Instead, a simple linear classifier, often referred to as a fully connected layer, is appended to the output of the frozen model. This linear layer is supervised, relying on labeled data from a new task or dataset to adjust its weights. The core idea behind linear probing is to use this additional classifier to determine how well the features learned by the frozen model can transfer to a new task.

The main objective of linear probing is to assess the effectiveness of the features captured by the pre-trained model. If the pre-trained encoder has learned generalizable and robust features during its training on large-scale datasets (as in the case of Hybrid-ConVIRT, trained with medical image-text pairs), the linear classifier should perform well on the new task even without the need to fine-tune the base model. This indicates that the learned features are not specific to the original training data but rather transferable to new tasks, showcasing the generalization capabilities of the encoder.

Application to Hybrid-ConVIRT

In this work, the base image encoder used for linear probing was pre-trained as part of Hybrid-ConVIRT, a model designed for joint learning of medical image-text representations. The details of how the Hybrid-ConVIRT model was trained, including the contrastive learning approach and fusion of the Contrastive Similarity Matrix (CSM) with the Predicted Similarity Matrix (PSM), can be found in Chapter 3 (Methodology). The model's image encoder was trained to capture rich, domain-specific features from medical images in combination with their corresponding textual reports. These features are expected to generalize well to new tasks, even when only a linear classifier is trained on top of the frozen encoder.

The goal of using linear probing in this context is to evaluate the efficacy of the learned medical image representations without fine-tuning the Hybrid-ConVIRT model.

By training a linear layer on top of the frozen image encoder and using it to classify new medical tasks, we can measure how well the pre-trained encoder has captured useful, transferable features. If the linear classifier achieves high accuracy on these tasks, it indicates that the encoder has successfully learned generalizable representations during its pre-training.

This approach is not only computationally efficient but also highlights the potential of Hybrid-ConVIRT as a powerful feature extractor for medical imaging tasks. The diagrammatic representation of this evaluation process can be seen in Figure 4.4.

4.2.4 Evaluation Metrics

Let's discuss the metrics in the results. One key aspect of the evaluation is that we used macro-averaging to calculate the performance metrics, which ensures that each class, regardless of its size, is treated equally. This is especially important for imbalanced datasets, as it prevents the model from focusing too much on the majority class.

We will first explain how macro-averaging was applied [35, 27], both for binary classification and multi-class classification, and show how metrics like Accuracy, Precision, Recall, and F1-score were calculated. For binary classification, the performance metrics are computed separately for each class, and then the average is taken. For multi-class classification, the same principle applies, with the metrics averaged across all the classes, giving each one the same weight.

Macro averaging is a method used to compute performance metrics, such as Precision, Recall, and F1-score, in a way that gives equal weight to each class in a classification task, regardless of the class size or distribution. This approach is particularly useful in multi-class classification problems, where the class distributions are often imbalanced, as it ensures that the evaluation is not biased toward classes with a higher number of samples. In multi-class classification, metrics are first calculated individually for each class, treating each one as if it were the positive class, and then the average of these metrics is computed. This results in an overall metric that reflects the model's performance across

all classes, providing a balanced evaluation that is more informative than simply focusing on the majority class.

While macro averaging is most commonly associated with multi-class classification, it is equally valuable when evaluating binary classification tasks. In binary classification, macro averaging ensures that performance is not skewed toward the dominant class, which is particularly important when dealing with imbalanced datasets. By treating the positive and negative classes separately and then averaging the metrics, we obtain a clearer picture of how the model performs across both classes. This is crucial in real-world applications, especially in medical image classification, where the minority class often represents the cases of greatest interest, such as disease presence in the chest x-ray images. The authors of "Optimal Thresholding of Classifiers to Maximize F1 Measure" [27] higlight the importance of macro-averaging

Given that two of the four datasets in our study (CoVID, Tuberculosis) involve binary classification, while the remaining two (Pneumonia/TB Mix, CheXpert) involve multi-class classification, we opted to consistently use macro averaging for all evaluation metrics. This choice ensures a fair and consistent comparison across datasets, allowing us to avoid bias introduced by class imbalances in both the binary and multi-class cases. Macro averaging provides a more holistic view of model performance, enabling us to better understand the strengths and weaknesses of each model in classifying both the majority and minority classes across a range of datasets. By applying this method, we ensure that the evaluation reflects the model's ability to generalize across all categories, rather than focusing solely on the classes with more samples.

This is the small code snippet that is used when calculating the macro-average throughout the for all models across all datasets.

```
precision = precision_score(all_labels, all_predictions, average='macro',
    zero_division=0)
recall = recall_score(all_labels, all_predictions, average='macro',
    zero_division=0)
```

```
f1 = f1_score(all_labels, all_predictions, average='macro', zero_division=0)
```

Of course, there is no need to initialize the function with 'macro' or 'binary' for accuracy because it is always calculated the same way, all the correct predictions divided by all the predictions, regardless of the number of classes.

An Example of the Macro-Average calculation of Metrics for a binary case

Below is how the calculation would have been done by sci-kit learn on Fold 1 with a 0.4 threshold of Hybrid-ConVIRT on the CoVID dataset. The entire table of all the thresholds across all 5 folds for all the models can be seen in the Appendix. The average scores can are in Table 5.2 above.

Performance metric values of Fold 1 with a 0.4 threshold:

- Accuracy:0.9345
- Precision: 0.9236
- Recall: 0.9044
- F1: 0.9134
- True Positives (TP): 609
- True Negatives (TN): 1972
- False Positives (FP): 66
- False Negatives (FN): 115

Accuracy - Accuracy is calculated for the entire dataset and does not change between macro and binary averaging.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Substituting the given values:

$$\text{Accuracy} = \frac{609 + 1972}{609 + 1972 + 66 + 115} = \frac{2581}{2762} \approx 0.935$$

Precision (Macro-Averaging)- Precision is calculated separately for the positive and negative classes and then averaged.

For the positive class:

$$\text{Precision}_{\text{positive}} = \frac{TP}{TP + FP} = \frac{609}{609 + 66} = \frac{609}{675} \approx 0.902$$

For the negative class:

$$\text{Precision}_{\text{negative}} = \frac{TN}{TN + FN} = \frac{1972}{1972 + 115} = \frac{1972}{2087} \approx 0.945$$

Macro-averaged Precision:

$$\text{Precision}_{\text{macro}} = \frac{\text{Precision}_{\text{positive}} + \text{Precision}_{\text{negative}}}{2} = \frac{0.902 + 0.945}{2} \approx 0.924$$

Recall (Macro-Averaging)- Recall is also calculated separately for the positive and negative classes and then averaged.

For the positive class:

$$\text{Recall}_{\text{positive}} = \frac{TP}{TP + FN} = \frac{609}{609 + 115} = \frac{609}{724} \approx 0.842$$

For the negative class:

$$\text{Recall}_{\text{negative}} = \frac{TN}{TN + FP} = \frac{1972}{1972 + 66} = \frac{1972}{2038} \approx 0.968$$

Macro-averaged Recall:

$$\text{Recall}_{\text{macro}} = \frac{\text{Recall}_{\text{positive}} + \text{Recall}_{\text{negative}}}{2} = \frac{0.842 + 0.968}{2} \approx 0.904$$

F1 Score (Macro-Averaging)- F1 Score is calculated separately for the positive and negative classes, and then averaged.

For the positive class:

$$\text{F1}_{\text{positive}} = 2 \times \frac{\text{Precision}_{\text{positive}} \times \text{Recall}_{\text{positive}}}{\text{Precision}_{\text{positive}} + \text{Recall}_{\text{positive}}} = 2 \times \frac{0.902 \times 0.842}{0.902 + 0.842} \approx 0.871$$

For the negative class:

$$\text{F1}_{\text{negative}} = 2 \times \frac{\text{Precision}_{\text{negative}} \times \text{Recall}_{\text{negative}}}{\text{Precision}_{\text{negative}} + \text{Recall}_{\text{negative}}} = 2 \times \frac{0.945 \times 0.968}{0.945 + 0.968} \approx 0.956$$

Macro-averaged F1 Score:

$$\text{F1}_{\text{macro}} = \frac{\text{F1}_{\text{positive}} + \text{F1}_{\text{negative}}}{2} = \frac{0.871 + 0.956}{2} \approx 0.913$$

An Example of the Macro-Average calculation of Metrics for a multi-class case

Below is how the calculation would have been done by sci-kit on Fold 1 with a 0.4 threshold of Hybrid-ConVIRT on the Pneumonia/TB dataset (3 classes). The entire table of all the thresholds across all 5 folds for all the models can be seen in the Appendix. The average scores can are in Table 5.2 above. Similarly for the CheXpert dataset with 5 classes, there would be a 5 x 5 matrix as the confusion matrix.

The given confusion matrix is:

$$\begin{bmatrix} 1603 & 341 & 98 \\ 123 & 1000 & 32 \\ 18 & 12 & 267 \end{bmatrix}$$

True Positives (TP), False Positives (FP), and False Negatives (FN) for each class

- **Class 1 (first row)**:
  - TP = 1603
  - FP = 141 (sum of column 1 minus TP)
  - FN = 439 (sum of row 1 minus TP)
- **Class 2 (second row)**:
  - TP = 1000
  - FP = 353 (sum of column 2 minus TP)
  - FN = 155 (sum of row 2 minus TP)
- **Class 3 (third row)**:
  - TP = 267
  - FP = 130 (sum of column 3 minus TP)
  - FN = 30 (sum of row 3 minus TP)

Macro-Averaged Precision: Precision calculated for each class

$$\text{Precision}_1 = \frac{TP_1}{TP_1 + FP_1} = \frac{1603}{1603 + 141} = 0.919$$

$$\text{Precision}_2 = \frac{TP_2}{TP_2 + FP_2} = \frac{1000}{1000 + 353} = 0.739$$

$$\text{Precision}_3 = \frac{TP_3}{TP_3 + FP_3} = \frac{267}{267 + 130} = 0.672$$

$$\text{Precision}_{\text{macro}} = \frac{0.919 + 0.739 + 0.672}{3} = 0.777$$

Macro-Averaged Recall Recall calculated for each class

$$\text{Recall}_1 = \frac{TP_1}{TP_1 + FN_1} = \frac{1603}{1603 + 439} = 0.785$$

$$\text{Recall}_2 = \frac{TP_2}{TP_2 + FN_2} = \frac{1000}{1000 + 155} = 0.866$$

$$\text{Recall}_3 = \frac{TP_3}{TP_3 + FN_3} = \frac{267}{267 + 30} = 0.899$$

$$\text{Recall}_{\text{macro}} = \frac{0.785 + 0.866 + 0.899}{3} = 0.850$$

Macro-Averaged F1-Score: F1 calculated for each class

$$\text{F1}_1 = 2 \times \frac{0.919 \times 0.785}{0.919 + 0.785} = 0.847$$

$$\text{F1}_2 = 2 \times \frac{0.739 \times 0.866}{0.739 + 0.866} = 0.797$$

$$\text{F1}_3 = 2 \times \frac{0.672 \times 0.899}{0.672 + 0.899} = 0.767$$

$$\text{F1}_{\text{macro}} = \frac{0.847 + 0.797 + 0.767}{3} = 0.804$$

4.3 Results

In this section, we present the results of evaluating the performance of the four models—CLIP, MedCLIP, BioViL, and Hybrid-ConVIRT—on four different datasets: CoVID, Tuberculosis, Pneumonia/TB, and CheXpert (5 classes). These models were used as feature extractors, and the extracted features, which are 512-dimensional, were fed into a linear probe classifier to assess the overall model performance. The results provide an in-depth comparison across several important evaluation metrics: Accuracy,

Precision, Recall, F1 score, ROC (Receiver Operating Characteristic) curves, and AUC (Area Under the Curve).

Evaluating the model performance on different threshold values (0.4, 0.5, and 0.65) allows us to gain insights into how the models behave when the decision boundary is adjusted. A fixed threshold may not always provide the best indication of model performance, especially in tasks with imbalanced datasets or when the cost of false positives and false negatives is significant. By evaluating at different thresholds, we can observe how sensitive the models are to these changes, helping us understand their robustness and adaptability across various decision-making criteria. This ensures that the model is not only performing well at a single threshold but also across a range of possible scenarios.

Moreover, ROC curves and AUC are particularly useful in this context, as they provide a comprehensive measure of the classifier's performance across all possible thresholds, rather than a single fixed point. The ROC curve shows the trade-off between true positive and false positive rates, while the AUC gives an aggregate measure of model performance, irrespective of the chosen threshold. A higher AUC indicates that the model is generally better at distinguishing between classes, making it one of the most reliable metrics for comparing models, especially when dealing with imbalanced datasets. Thus, focusing on both ROC and AUC gives us a clear understanding of how the model would perform across varying threshold levels and in different real-world scenarios.

The tables below summarize the results for each dataset at three threshold values—0.4, 0.5, and 0.65 for the binary datasets. Of course for the multi-class datasets the thresholds are not considered. The reported numbers represent the average performance across the five folds of cross-validation. For a detailed breakdown of the metric scores for each fold, refer to the Appendix. This approach ensures a thorough evaluation of each model, highlighting the models that converge to a lower loss the quickest in the 10 epocs the linear probe was trained.

4.3.1 Assessing Linear Probe Performance on CoVID

Table 4.2: **Average Metrics across Five Folds for Various Models on COVID**

Model	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
BioVil	0.40	0.827	0.790	0.858	0.804	667.6	1616.8	421.6	55.6
	0.50	0.880	0.839	0.872	0.852	618.4	1811.8	226.6	104.8
	0.65	0.895	0.889	0.830	0.853	501.4	1969.8	68.6	221.8
CLIP	0.40	0.746	0.723	0.719	0.694	479.4	1580.6	457.8	243.8
	0.50	0.773	0.748	0.702	0.698	400.2	1734.4	204.0	323.0
	0.65	0.788	0.784	0.663	0.669	289.2	1886.4	152.0	434.0
Hybrid-ConVirt	0.40	**0.914**	**0.889**	**0.910**	**0.895**	651.4	1874.0	457.8	71.8
	0.50	**0.925**	**0.906**	**0.910**	**0.905**	635.6	1918.8	204.0	87.6
	0.65	**0.933**	**0.925**	**0.902**	**0.911**	606.2	1969.6	152.0	117.0
MedCLIP	0.40	0.872	0.830	0.879	0.847	645.8	1762.2	276.2	77.4
	0.50	0.895	0.858	0.885	0.869	623.6	1848.4	190.0	99.6
	0.65	0.913	0.893	0.878	0.885	581.8	1938.8	99.6	141.4

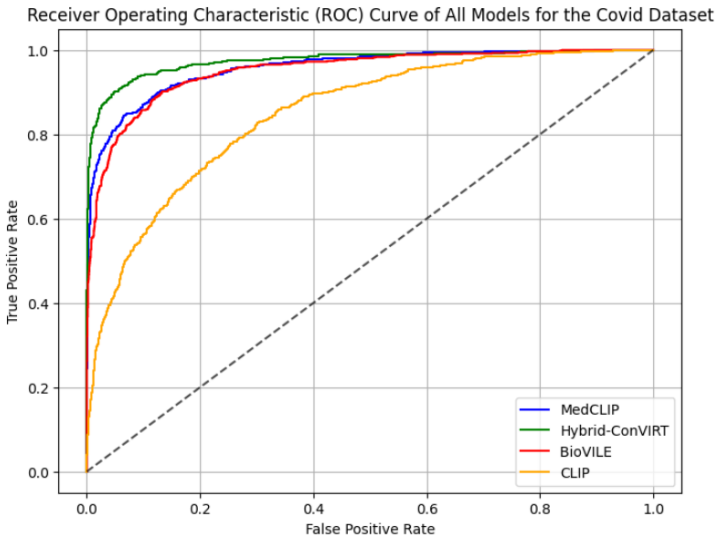


Figure 4.5: ROC Curve of All Models for CoVID

4.3.2 Assessing Linear Probe Performance on Tuberculosis

Table 4.3: **Average Metrics across Five Folds for Various Models on Tuberculosis**

Model	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
BioVil	0.40	0.937	0.935	0.829	0.871	74.8	554.6	5.4	37.2
	0.50	0.927	0.956	0.784	0.840	63.8	559.4	0.6	48.2
	0.65	0.919	0.956	0.758	0.817	57.8	560.0	0.0	54.2
CLIP	0.40	0.899	0.854	0.763	0.794	62.6	541.8	18.2	49.4
	0.50	0.880	0.925	0.643	0.687	32.2	559.0	1.0	79.8
	0.65	0.856	0.926	0.568	0.579	15.2	560.0	0.0	96.8
Hybrid-ConVirt	0.40	0.960	0.949	0.906	0.925	92.4	553.0	7.0	19.6
	0.50	0.956	0.956	0.881	0.913	86.2	556.0	4.0	25.8
	0.65	0.951	0.959	0.861	0.901	81.4	557.4	2.6	30.6
MedCLIP	0.40	**0.963**	**0.956**	**0.908**	**0.929**	92.4	554.6	5.4	19.6
	0.50	**0.957**	**0.962**	**0.881**	**0.915**	86.0	557.2	2.8	26.0
	0.65	**0.952**	**0.964**	**0.863**	**0.903**	81.6	558.2	1.8	30.4

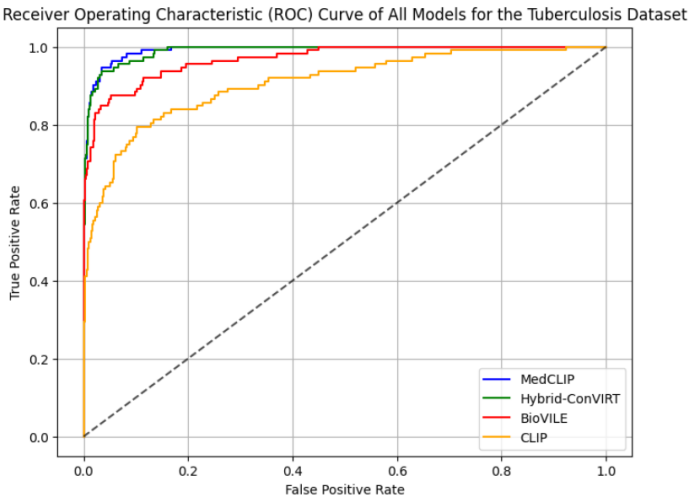


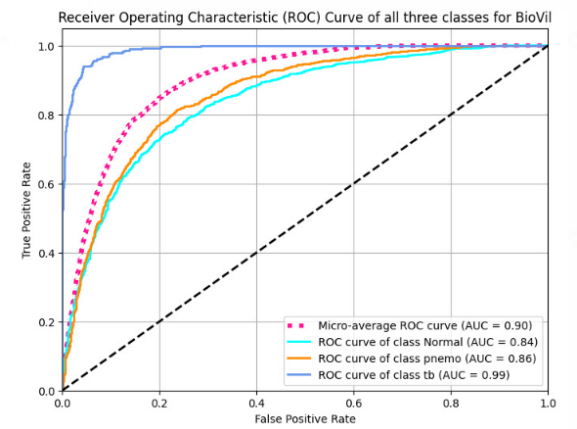
Figure 4.6: ROC Curve of All Models for Tuberculosis

4.3.3 Assessing Linear Probe Performance on Pneumonia

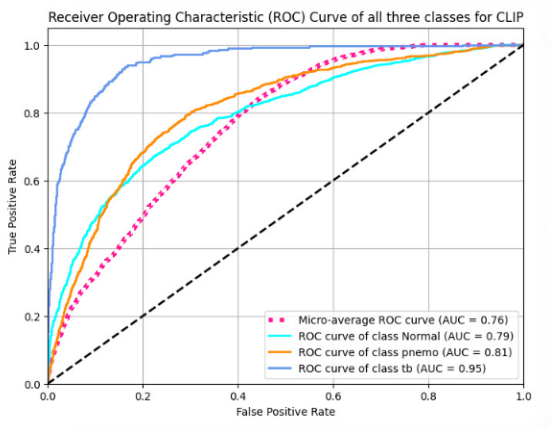
This dataset has 3 classes, so this classification becomes a multi-class classification. The way to interpret the results and how the metrics have been calculated can be seen in Section 5.5

Table 4.4: **Average Metrics across Five Folds for Various Models on Pneumonia**

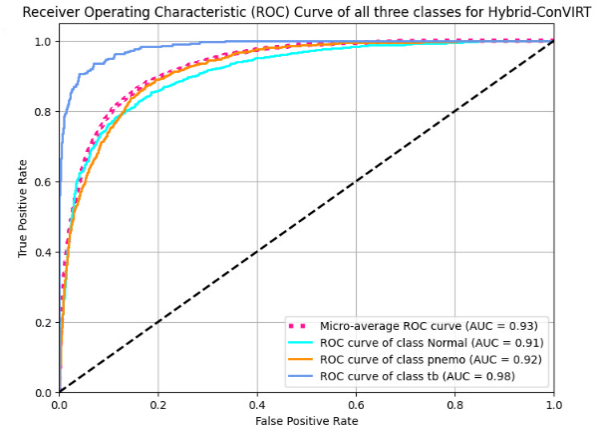
Model	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
CLIP	0.639	0.632	0.717	0.625	(1084.0, 723.0, 234.2) (195.0, 898.6, 61.4) (30.2, 17.2, 250.4)
BioVil	0.752	0.725	0.806	0.755	(1455.4, 463.0, 122.8) (244.0, 895.8, 15.2) (18.2, 2.4, 277.2)
Hybrid-ConVirt	**0.813**	**0.765**	**0.839**	**0.791**	(1598.8, 332.8, 109.6) (135.0, 977.0, 43.0) (24.6, 8.4, 264.8)
MedCLIP	0.797	0.733	0.818	0.761	(1554.4, 343.2, 143.6) (116.0, 976.4, 62.6) (34.4, 11.2, 252.2)



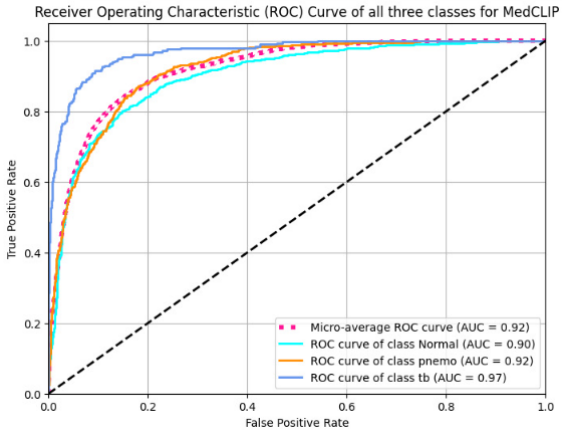
(a) ROC of all 3 classes and the Micro-AVG curve for BioVIL



(b) ROC of all 3 classes and the Micro-AVG curve for CLIP



(c) ROC of all 3 classes and the Micro-AVG curve for Hybrid-ConVIRT



(d) ROC of all 3 classes and the Micro-AVG curve for MedCLIP

Figure 4.7: ROC and Micro-Average curves of all classes for the Various Models

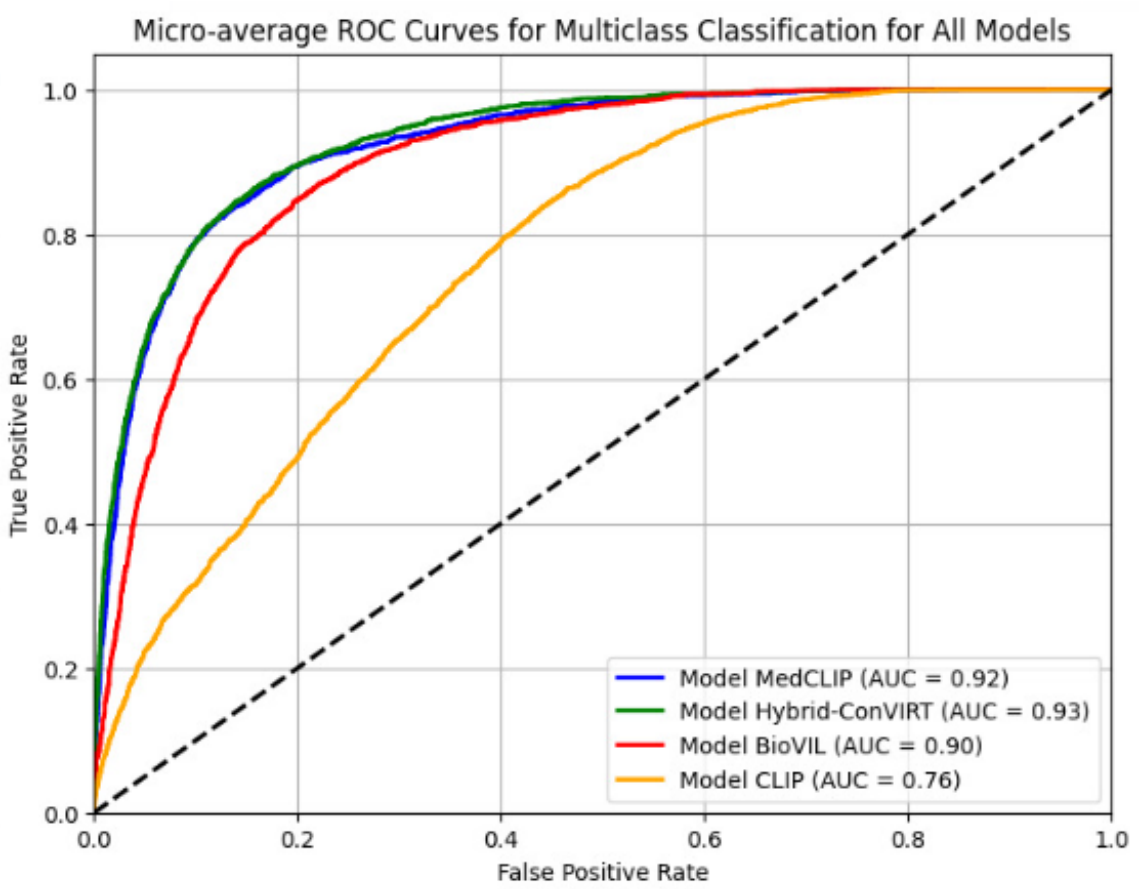


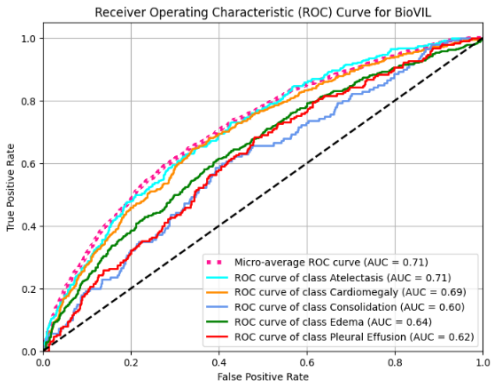
Figure 4.8: ROC Curve of All Models for Pneumonia

4.3.4 Assessing Linear Probe Performance on CheXpert

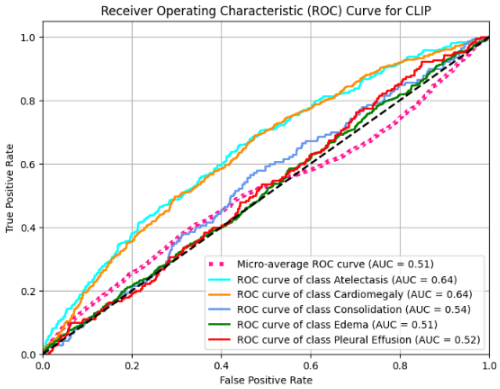
This dataset has 5 classes, so this classification becomes a multi-class classification again. The way to interpret the results and how the metrics have been calculated can be seen in Section 5.5

Table 4.5: **Average Metrics across Five Folds for Various Models on CheXpert**

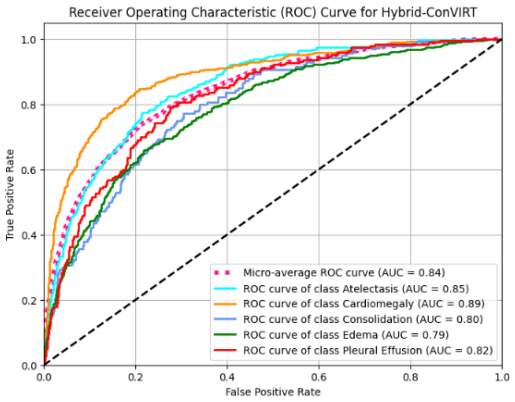
Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Confusion Matrix**
BioVIL	0.313	0.312	0.292	0.250	[87.8, 55.8, 50.4, 49.4, 65.2] [69.0, 250.2, 87.2, 91.6, 94.0] [22.4, 30.2, 32.6, 30.2, 24.6] [93.6, 163.4, 114.4, 165.6, 101.4] [35.6, 39.8, 32.4, 33.0, 49.6]
MedCLIP	0.537	**0.507**	0.514	0.481	[173.4, 16.6, 31.6, 27.0, 60.0] [41.8, 395.8, 43.6, 76.6, 34.2] [25.4, 12.4, 61.4, 19.4, 21.4] [103.8, 96.8, 84.2, 287.6, 66.0] [44.6, 15.6, 25.8, 18.6, 85.8]
CLIP	0.221	0.166	0.234	0.143	[201.6, 40.6, 8.2, 5.2, 53.0] [288.4, 159.0, 32.0, 9.8, 102.8] [72.6, 26.0, 7.2, 2.4, 31.8] [353.6, 127.2, 25.2, 10.4, 122.0] [105.0, 38.4, 10.4, 2.0, 34.6]
Hybrid	**0.557**	0.501	**0.525**	**0.502**	[164.0, 17.0, 26.8, 35.8, 65.0] [32.6, 395.4, 35.0, 91.8, 37.2] [16.6, 13.0, 60.6, 24.4, 25.4] [84.0, 89.6, 72.4, 331.4, 61.0] [34.0, 15.0, 28.2, 23.2, 90.0]



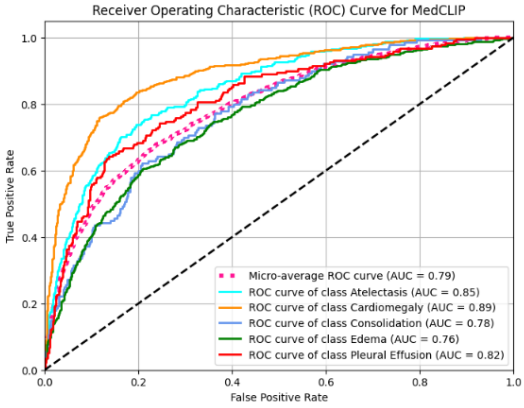
(a) ROC of all 5 classes and the Micro-AVG curve for BioVIL



(b) ROC of all 5 classes and the Micro-AVG curve for CLIP



(c) ROC of all 5 classes and the Micro-AVG curve for Hybrid-ConVIRT



(d) ROC of all 5 classes and the Micro-AVG curve for MedCLIP

Figure 4.9: ROC and Micro-Average curves of all classes for the Various Models

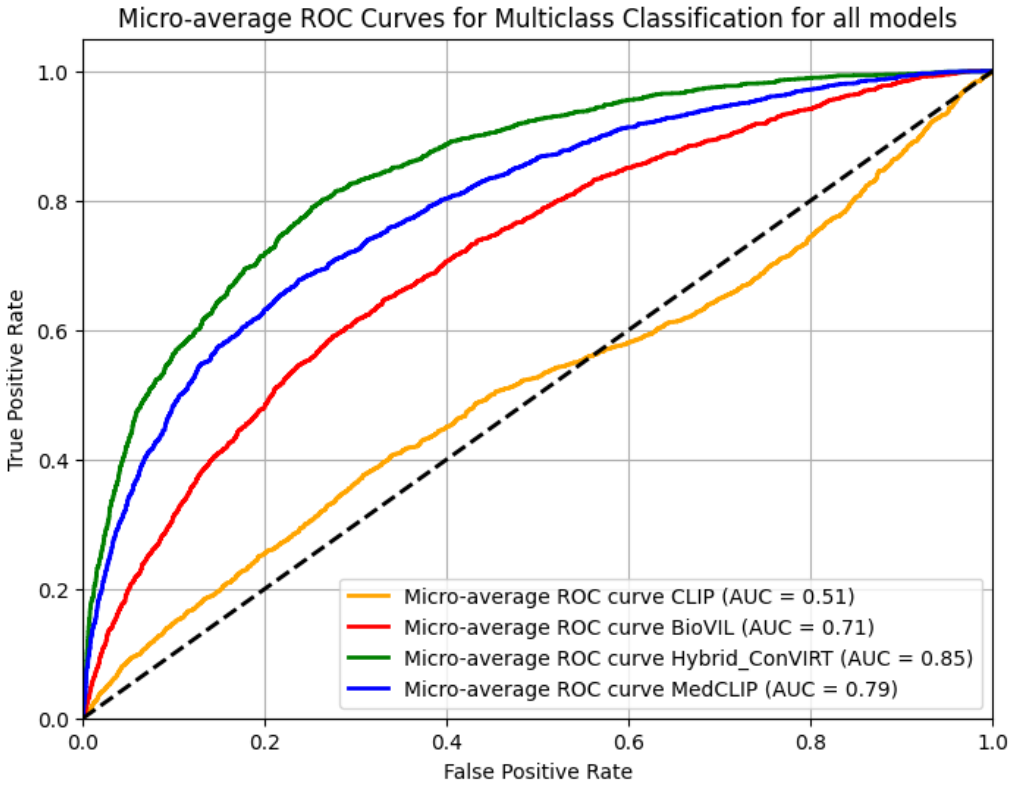


Figure 4.10: ROC Curve of All Models for CheXpert

4.3.5 Evaluating Tuberculosis on Zero-Shot Classification

Table 4.6: **Zero-Shot: Performance Metrics for Various Models on Tuberculosis**

Model	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
MECLIP	**0.717**	**0.660**	**0.778**	**0.654**	608	2403	1097	92
Hybrid	0.669	0.600	0.671	0.588	472	2337	1163	228
BioVIL	0.455	0.563	0.602	0.437	576	1335	2165	124
CLIP	0.425	0.488	0.479	0.390	392	1391	2109	308

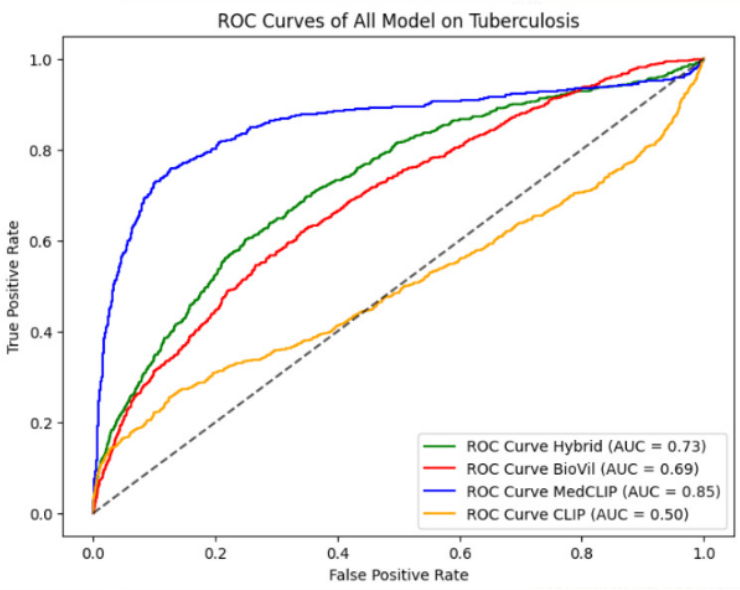


Figure 4.11: Zero-shot Classification: ROC Curve of All Models on Tuberculosis

4.3.6 Evaluating CoVID on Zero-Shot Classification

Table 4.7: **Zero-Shot: Performance Metrics for Various Models on COVID**

Model	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
MECLIP	0.778	0.734	0.784	0.745	2873	7873	2319	743
Hybrid-ConVIRT	**0.843**	**0.795**	**0.820**	**0.806**	2792	8853	1339	824
BioVIL	0.760	0.817	0.546	0.517	349	10143	49	3267
CLIP	0.738	0.369	0.500	0.425	0	10192	0	3616

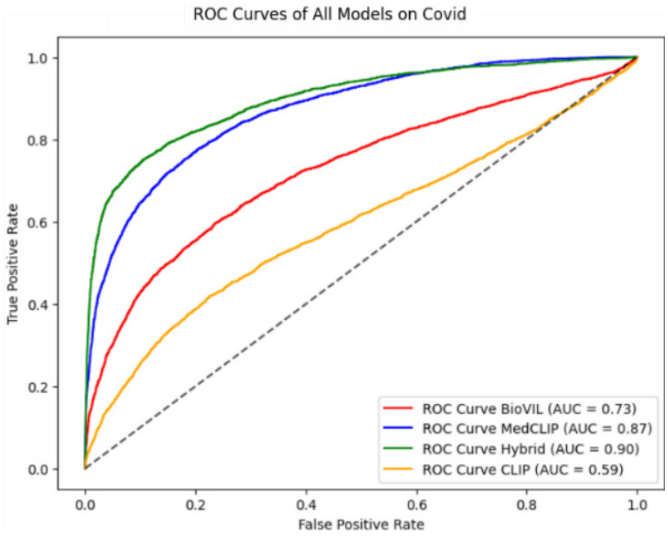


Figure 4.12: Zero-shot Classification: ROC Curve of All Models on CoVID

Chapter 5

Discussion

This study explored the feature extraction capabilities of four models—CLIP, BioViL, MedCLIP, and Hybrid-ConVIRT—across four medical imaging datasets: COVID, Tuberculosis, Pneumonia/TB Mix, and CheXpert. Each model's feature quality was assessed by attaching a linear classifier (probe) for evaluation, allowing a comparison of the discriminative ability of the features produced by each backbone model. Additionally, we conducted zero-shot classification on the COVID and Tuberculosis datasets to evaluate the models' generalizability to previously unseen data. Below, we discuss each dataset individually, focusing on both linear probe and zero-shot performance. We also discuss the the choice of evaluation metrics used and why certain thresholds value were chosen for the binary classification task and disscuss the choice of One vs Rest ROC and how that can be useful in a multi-class setting.

In our results, we initially obtained metric scores with six decimal places (all tables in the Appendix are in this form), providing a high level of precision in capturing model performance. However, for clarity and readability, we reported all results in three decimal places in the section above. This rounding simplifies comparisons without sacrificing significant interpretative value. Specifically, when analyzing performance on the tuberculosis dataset, the difference between Hybrid-ConVIRT and MedCLIP scores appeared

minimal across metrics. While MedCLIP consistently outperformed Hybrid-ConVIRT slightly, this performance difference became visible only from the third decimal point onward, with MedCLIP maintaining a marginal edge across all five cross-validation folds. This slight variance underscores MedCLIP's stable advantage, albeit negligible in practical terms.

5.1 Threshold Selection

When using a linear classifier, a threshold is a cutoff value applied to the classifier's output score (often a probability) to determine the predicted class. In a binary classification setting, a score above the threshold would classify the instance as positive, while a score below it would classify the instance as negative. Choosing an appropriate threshold is significant because it directly impacts the classifier's performance metrics, such as accuracy, precision, and recall, by balancing the trade-off between sensitivity and specificity.

In this analysis, thresholds play a crucial role in highlighting the relative strengths and weaknesses of each model (MedCLIP, Hybrid-ConVIRT, CLIP, and BioVIL). By examining the models' performance at multiple thresholds, I aimed to observe how changes in threshold values affect each model's ability to correctly classify instances. Lower thresholds tend to increase recall (i.e., capturing more positive instances), which may be valuable in scenarios where missing positive cases is costly. In contrast, higher thresholds prioritize precision (i.e., reducing false positives), which is preferable in cases where minimizing incorrect positive predictions is essential.

I experimented with a range of thresholds, specifically 0.4, 0.5, and 0.65, as these values provided distinct, observable variations in the performance metrics across all models. These particular thresholds were chosen based on observed results that indicated meaningful differences in accuracy, precision, and recall for each model. Lower values, like 0.4, increased recall across the models, capturing more positive samples but at the cost

of precision. Higher values, such as 0.65, prioritized precision, showing how each model balances these metrics differently when required to make more conservative positive predictions.

While the process of selecting thresholds can be somewhat arbitrary, I specifically chose these values because they effectively maximized the interpretability of the results. These thresholds revealed the most distinct performance differences between models, enabling a clearer comparative analysis. Other values, such as 0.3 or 0.6, were also tested but did not show as distinct variations in performance or were less effective in illustrating clear trends across models. Therefore, 0.4, 0.5, and 0.65 were chosen to optimize the visibility of performance variations, making it easier to understand how each model responds to varying thresholds.

We see the analysis of "threshold-tuning" and "threshold moving" in these studies, [15, 20], it is a common method, where different thresholds are tested to optimize certain evaluation metrics such as F1 score or AUC. This process is often empirical, involving trial and error to find the best trade-offs between false positives and false negatives, especially when there's no clear real-world cost function to dictate a specific threshold. Thus, selecting thresholds of 0.4, 0.5, and 0.65 could be justified by their practical effectiveness in revealing differences between models, rather than fixed rules. This aligns with the empirical approach often seen in the field, where thresholds are chosen based on how well they highlight model performance across desired metrics.

5.2 One-vs-Rest ROC Evaluation in a Multi-class Setting

The Receiver Operating Characteristic (ROC) curve is a well-established tool used to evaluate model performance in binary classification tasks. By plotting the true positive rate (sensitivity) against the false positive rate for various threshold levels, it reveals

how well a model distinguishes between positive and negative classes. The Area Under the ROC Curve (AUC-ROC) score provides a single value summarizing the curve, with higher values indicating stronger separation between classes [17]. However, in a multiclass setting, where the model produces probabilities for several classes instead of just two, applying the ROC approach requires extra considerations.

In a multiclass setting, models trained with categorical cross-entropy loss are designed to output a probability distribution across all classes, with the highest probability representing the model's predicted class. This contrasts with binary models that make direct positive-or-negative classifications. Because the multiclass model is trained to produce a single probability distribution over all classes, using a One-vs-Rest (OvR) ROC approach can seem somewhat mismatched, as it forces the multiclass predictions into separate binary evaluations for each class.

Justification for Using One-vs-Rest ROC in Multiclass Evaluation The One-vs-Rest (OvR) ROC approach can still be a valuable evaluation method for multiclass classifiers. By examining each class individually, with one class considered "positive" and the rest "negative," the OvR approach allows us to evaluate how well the model distinguishes each class from the others. This can be particularly insightful when the goal is to compare the model's discriminative power across different classes, especially in cases where some classes may be harder to identify. Therefore, even if the model is not explicitly trained for binary classification, OvR ROC still provides a useful way to assess class-level separability [35].

Using One-vs-Rest ROC as a Post-Training Evaluation Rather Than Part of the Training Process An important distinction to make is that OvR ROC curves serve as an evaluation tool rather than a training objective. When using categorical cross-entropy loss, the model is optimized to maximize the probability for the correct class in a single distribution across all classes, without directly applying binary thresholds. Using

OvR ROC for post-training evaluation, then, does not alter the model's behavior but rather offers a way to analyze class separability. In this context, each class probability is treated independently for evaluation purposes, even though the model was trained to output probabilities for all classes at once [13].

Limitations of One-vs-Rest ROC for Overall Multiclass Performance While OvR ROC is useful for understanding how well the model separates each class individually, it may not be the best choice if the goal is to assess overall multiclass performance or capture how the model was intended to behave. The nature of ROC curves, being threshold-dependent, doesn't fully align with how multiclass models trained with cross-entropy loss operate, as these models select the class with the highest probability rather than applying thresholds to individual classes. If the objective is to understand how the model performs as a whole, it may be more meaningful to evaluate accuracy, precision, recall, and F1-score across all classes. These metrics provide a clearer picture of the model's overall behavior in a multiclass context.

While OvR ROC evaluation is not a direct reflection of a model's multiclass training objective, it still offers valuable insights into class-level discrimination. By analyzing each class separately, OvR ROC allows us to see how well the model identifies individual classes. However, for assessing overall performance and understanding the model's multiclass behavior, metrics like accuracy, precision, recall, and F1-score may provide a more straightforward evaluation. The combination of OvR ROC for class-level insights and standard performance metrics for overall behavior helps ensure a comprehensive assessment of multiclass classifiers.

5.3 Interpreting Results

CoVID For the COVID dataset, Hybrid-ConVIRT consistently demonstrated the highest performance across both the linear probe (Table 5.2) and zero-shot (Table 5.7) set-

tings. In the linear probe setup, where a lightweight linear classifier was trained on the dataset for ten epochs with a consistent learning rate across models, Hybrid achieved the best metrics across accuracy, precision, recall, and F1-score. Notably, Hybrid maintained this high performance across all three thresholds (0.40, 0.50, and 0.65), indicating its robustness in producing features that remain discriminative under various decision boundaries. Observing the ROC curves further supports this, with Hybrid achieving the highest AUC scores, suggesting that its feature representations are more separable for this dataset than those of the other models.

In the zero-shot setting, where no classifier training was conducted, Hybrid continued to outperform the other models, though with slightly lower overall metrics. This drop is expected, as the zero-shot approach relies on direct similarity comparisons between embeddings without any prior exposure to the dataset, unlike the linear probe, where the classifier was trained on the COVID data. Nonetheless, Hybrid's superior performance, followed by MedCLIP, BioViL, and CLIP, highlights its ability to produce transferable feature representations that generalize well to unseen data. The performance of CLIP, in particular, was comparatively lower, likely due to its lack of training on medical images, which limits its ability to capture the nuanced features present in medical imaging.

Tuebrculosis For the Tuberculosis dataset, MedCLIP achieved slightly better results than other models in the linear probe setting (Table 5.3). The classifier attached to MedCLIP converged to a lower loss faster than the others, suggesting a more efficient learning process for this model on this dataset. However, the metric scores for MedCLIP were only marginally higher than those of Hybrid-ConVIRT, which was the second-best performer. BioViL also performed reasonably well on this dataset, showing higher scores than on COVID, while CLIP displayed improved metrics compared to its performance on the COVID dataset. These higher scores across all models may be attributed to the smaller size of the Tuberculosis dataset, which may have facilitated quicker convergence

for the classifiers attached to each model, leading to comparable performance across models.

In the zero-shot evaluation (Table 5.6), the performance in all models was significantly lower than in the linear probe setting. This reduction is likely due to the lack of classifier training, as zero-shot testing merely computes similarity scores between embeddings, without adjusting for the dataset-specific patterns that the linear probe could learn. Here, CLIP performance dropped substantially, with a noticeably lower AUC compared to its linear probe counterpart, underscoring the importance of domain-specific training data for medical imaging tasks. Hybrid-ConVIRT and MedCLIP continued to show superior performance compared to the other models, confirming their ability to produce more discriminative and transferable feature representations even without direct training on the Tuberculosis dataset.

Overall, for both zero-shot and linear probe evaluations, MedCLIP achieves higher scores than Hybrid on the Tuberculosis dataset. In the linear probe setting, the difference is slight, likely due to the linear layer being trained on the Tuberculosis images, giving it exposure to the dataset. However, in the zero-shot setting, where no additional training on the Tuberculosis images occurs, MedCLIP's advantage is more pronounced. This raises the question of why MedCLIP performs better on this dataset. One explanation is that the Hybrid model was trained exclusively on 100,000 images from CheXpert, which contained no Tuberculosis cases, while MedCLIP was trained on both MIMIC-CXR and CheXpert, potentially including cases with "Tuberculosis" in the diagnosis descriptions from the MIMIC-CXR set [48]. Additionally, MedCLIP was trained on a dataset more than three times the size of Hybrid's, which may improve its ability to produce embeddings for tuberculosis images and corresponding text that are closer in the feature space.

Pneumonia Dataset The pneumonia / TB mix data set presented a more complex multi-class setting, where Hybrid-ConVIRT showed slightly higher overall performance. Hybrid-ConVIRT's advantage across accuracy, precision, recall, and F1-score highlights its ability to generate features that support a more balanced classification, capturing both minority and majority class patterns effectively. The consistency of Hybrid's performance, maintained across a 5-fold cross-validation (see Appendix) setup, further reinforces its robustness and suggests that it provides more reliable features for general use.

To gain deeper insights into individual class performance, we analyzed the OvR ROC curves for each class. For the Tuberculosis vs. Rest classification within this dataset, BioViL achieved a notably high AUC of 99%, indicating excellent class separability. This result may explain the slight edge in precision of BioViL for this dataset which could suggest that there was a Tuberculosis class present in the dataset BioVIL was trained on. The other models also consistently identify tuberculosis cases with high confidence (Figure 4.9a, 4.9b, 4.9c, 4.9d). This overall trend of high performance on tuberculosis is also connected to the lower number of tuberculosis images present in the testing set during the five fold cross validation. Tuberculosis had the lowest number of images compared to the other two classes, so when that get further divided into 5 parts the validation would compare the lowest number of images hence overall all models perform well on it.

For the other classes (normal and pneumonia), the hybrid demonstrated marginally better performance in the OvR ROC, demonstrating its ability to maintain superior class separability across different categories. This aligns with the recommendations discussed in Section 6.2, where, while we discuss class separability for every class based on the OvR ROC curves, the metrics such as accuracy, precision, recall, and F1 score are prioritized to evaluate the model's overall multi-class performance and capture the intended behavior of the feature extraction capabilities of each model.

CheXpert (5 Classes) The CheXpert dataset, a five-class setup, posed a further challenge for assessing multiclass feature extraction. Hybrid-ConVIRT again emerged as the best-performing model across all evaluation metrics, displaying superior class-level discrimination. Examining the OvR ROC curves for each individual class, we observe that Hybrid achieved consistently higher AUC scores, further underscoring its enhanced ability to distinguish between different classes within this more complex dataset. This consistent performance across all CheXpert classes reflects Hybrid's capacity to adapt to a more varied dataset and generate nuanced, domain-appropriate features.

One possible reason for Hybrid's strong performance on this dataset is that it was trained on the 100,000-images of the CheXpert training set. In contrast, other models, such as MedCLIP and BioViL, were trained on splits of CheXpert and other datasets, like MIMIC-CXR, making their learning somewhat sparse in comparison. Additionally, Hybrid's training incorporated a more robust learning framework, including optimization toward a stronger similarity target (FSM), the application of image transformations for each training sample, and the use of CXR-BERT-specialized as the text encoder during contrastive training. These factors likely contributed to Hybrid's overall advantage in handling the 5 classes within the CheXpert test set.

5.4 Energy Usage and Inference Time Discussion

Training deep learning models can be computationally intensive and consume significant energy, especially when using high-powered GPUs. The energy consumption (E) for a GPU can be calculated using the formula:

$$E = P \times T$$

where:

- E is the energy consumption in kilowatt-hours (kWh),

- P is the power usage of the GPU in kilowatts (kW), and

- T is the time in hours (h) the GPU is used for training.

For the Hybrid-MedCLIP model, training was conducted over a period of 14 hours on an NVIDIA RTX 2080 TI. This GPU has an average power consumption of approximately 250 watts, or 0.25 kW. Thus, the energy usage can be calculated as follows:

$$E = 0.25\,\text{kW} \times 14\,\text{hours} = 3.5\,\text{kWh}$$

Hence, the training of the Hybrid-MedCLIP model consumed approximately 3.5 kWh. Notably, the training used 900 MiB out of the total 1126 MiB of GPU memory, which represents around 80% of the GPU's memory capacity. This calculation provides an estimate, as precise energy usage may vary slightly depending on the workload and GPU utilization efficiency.

Challenges in Comparative Energy Analysis

It is important to note that other models, such as MedCLIP and ConVIRT, did not clearly disclose their training times in published works, making direct comparisons in energy consumption challenging. Without publicly available training data, it is difficult to assess and compare the energy efficiency of these models relative to Hybrid-MedCLIP. This lack of transparency underscores the need for consistent reporting of computational resources in research, which would enable more accurate comparisons across models and support efforts towards more energy-efficient deep learning practices.

5.4.1 Analysis of Inference Times

The inference times for processing 1000 images from the COVID dataset reveal distinct variations in performance across different model architectures. Below is an analysis of the observed results:

Inference Times

- CLIP: The original Vision Transformer (ViT) architecture used by CLIP recorded an inference time of 230.59 seconds. This architecture is known for its simplicity and efficiency in processing images, making it relatively faster in this setting.

- MedCLIP: With an inference time of 339.77 seconds, MedCLIP utilizes the SWIN version of the ViT architecture. The SWIN transformer, designed for hierarchical feature representation, involves more complex computations, leading to higher inference times compared to the original ViT.

- BioViL: The BioViL model, which leverages a custom combination of ResNet50 and ViT, was the fastest, taking 179.66 seconds. The integration of ResNet50 might facilitate efficient initial feature extraction, contributing to reduced inference times.

- Hybrid-ConVIRT: The Hybrid-ConVIRT model also employed a SWIN ViT setup, resulting in a longer inference time of 323.61 seconds. Similar to MedCLIP, the architecture's intricate design contributes to the increased processing duration.

Performance vs. Speed Trade-off

It is notable that even though MedCLIP and Hybrid-ConVIRT have longer inference times, they outperform CLIP and BioViL in terms of accuracy and robustness. This discrepancy emphasizes that performance is not directly tied to the speed of image processing. Instead, the underlying training methodology and dataset quality play pivotal roles in enhancing model performance. MedCLIP and Hybrid-ConVIRT likely benefit from specialized training regimens and the utilization of medical imaging-specific datasets, optimizing their ability to extract nuanced features critical for clinical tasks.

In summary, while inference speed is a valuable metric, especially in real-time applications, the ultimate performance of these models hinges on how well they are trained

and the data they are exposed to. Therefore, a model's architectural complexity and training paradigm should be carefully considered based on the desired balance between efficiency and efficacy.

Chapter 6

Conclusion

In this research, we aimed to enhance medical image-text representation learning by introducing Hybrid-ConVIRT, a model designed to improve upon the MedCLIP framework. Our approach involved training both the image encoder and text encoder towards a Fused Similarity Matrix (FSM), which combined the Predicted Similarity Matrix (PSM) from MedCLIP with the Contrastive Similarity Matrix (CSM) from ConVIRT. This fused target was intended to provide a more robust training objective than the SSM (semantic similarity matrix used in MedCLIP) alone, capturing a richer, more meaningful similarity structure between image and text pairs.

Additionally, we incorporated several key modifications to improve model robustness. Unlike MedCLIP, we applied various image transformations during training, enhancing the model's adaptability to diverse imaging conditions and features. We also replaced the general purpose BioClinicalBERT with a more specialized CXR-BERT-spcialized initialization, which is better aligned with the nuances of chest X ray related text. With these new approaches, and training on a 100,000-image subset from CheXpert, Hybrid-ConVIRT consistently showed promising results. Out of six evaluations, which included linear probe testing on four datasets and zero-shot classification on two datasets, Hybrid-ConVIRT performed best in four. Even though the performance gains were sometimes

slight, they demonstrate that the hybrid approach, even with a smaller dataset (100,000 images versus the 350,000+ images used in MedCLIP's training), can be effective in producing a model that generalizes well across multiple classification tasks.

These findings support the potential of combining two models, leveraging MedCLIP's pre-trained PSM and ConVIRT's CSM, to create a more robust target matrix. However, while this combined approach proved effective in these classification settings, a broader evaluation is necessary to fully determine Hybrid-ConVIRT's capabilities. Testing Hybrid-ConVIRT on a wider range of tasks, such as segmentation, object detection, image-to-text retrieval, and text-to-image retrieval, would provide a more comprehensive assessment of its utility across diverse medical imaging applications.

6.1 Technological Advancements and Limitations

Medical imaging applications are evolving rapidly, with new models and techniques emerging frequently. Since this project began, multiple other models have been introduced, potentially offering superior performance on specific datasets or tasks compared to Hybrid-ConVIRT. Evaluating Hybrid-ConVIRT alongside these cutting-edge models could offer valuable insights into the strengths and limitations of our approach and indicate where this type of training stands in comparison to current advancements.

Furthermore, it is important to recognize that while models like Hybrid-ConVIRT, MedCLIP, and ConVIRT show promise in creating generalizable, robust representations using a contrastive learning framework, they remain experimental. Their primary strength lies in their versatility, allowing them to generalize across a range of tasks, from classification to potential applications in segmentation or detection. However, these general purpose models currently face limitations when compared to models trained specifically for a single task. A model trained and optimized solely for, say, segmentation or classification, still tends to outperform general models on that task due to task specific

training advantages.

6.2 Future Directions

This research highlights the value of a hybrid approach to medical image-text representation learning but also underscores the importance of comprehensive testing and benchmarking. While Hybrid-ConVIRT has shown promising results across several classification tasks, there are multiple avenues for future work to enhance its performance and validate its generalizability across various applications.

Expanding Task Scope and Testing: The current evaluation of Hybrid-ConVIRT focuses primarily on classification tasks through a linear probe and zero-shot testing. Expanding this evaluation across a broader range of tasks would offer a clearer picture of the model's practical utility in real-world clinical settings. Specifically, testing Hybrid-ConVIRT on segmentation, object detection, and image-text retrieval tasks could provide insights into its versatility and robustness. By applying Hybrid-ConVIRT to these tasks, we can better assess whether the combined training target (Fused Similarity Matrix) is capable of supporting diverse applications, or if further task-specific adaptations are required. Comparisons with specialized models and other recent, high-performing general-purpose models will also be valuable in understanding where Hybrid-ConVIRT stands in the rapidly evolving landscape of medical imaging technology.

Optimizing the Fused Similarity Matrix with a Tunable Alpha: An important future direction is optimizing the Fused Similarity Matrix by adjusting the weighting of the two components: the Contrastive Similarity Matrix (CSM) and the Predicted Similarity Matrix (PSM). In this study, these matrices were combined with equal weights (1:1) to form the Fused Similarity Matrix, which served as the target for training the encoders. However, it is likely that this equal weighting does not yield the most effective

results across all tasks or datasets. To address this, we propose treating the weighting parameter, α, as a tunable hyper-parameter. By experimenting with combinations defined by α * CSM + (1 - α) * PSM, we can search for the ideal balance that best aligns with the model's objectives, resulting in a potentially more powerful training target.

This optimization process would involve fine-tuning α based on performance metrics from tasks such as classification on a CheXpert test set. Introducing α as a hyperparameter allows the model to adapt more flexibly to the Fused Similarity Matrix, enhancing the feature extraction process and potentially leading to further gains in performance. Optimizing α could be especially impactful when Hybrid-ConVIRT is applied to different datasets or tasks, as the optimal weighting of CSM and PSM may vary based on the specific nature of the data.

Increasing Dataset Size and Diversity: In this study, Hybrid-ConVIRT was trained on 100,000 images from the CheXpert dataset, a substantial dataset by medical imaging standards. Nevertheless, the size of the dataset likely limits the model's ability to generalize fully across diverse imaging conditions and clinical contexts. Increasing the dataset size and expanding it to include more varied examples could significantly enhance the model's robustness and accuracy. Training on a larger dataset, possibly covering over 350,000 images from datasets like CheXpert, MIMIC-CXR, and others, would provide more comprehensive exposure to medical imaging diversity, thereby strengthening the model's generalizability across tasks and improving performance in zero-shot classification.

Benchmarking Against Emerging Models: The field of medical imaging and machine learning continues to evolve rapidly, with new models frequently emerging that may outperform Hybrid-ConVIRT. During the project, several advanced models have been introduced, each with unique training technnique and architectural changes. Comparing Hybrid-ConVIRT with these new, cutting-edge models would help determine where this

type of hybrid training approach stands in relation to the latest developments. Additionally, benchmarking Hybrid-ConVIRT on emerging tasks within medical imaging, such as multi-label classification or 3D image interpretation, would provide further insights into its strengths and areas for improvement.

Exploring Task-Specific Models vs. Generalized Models: Hybrid-ConVIRT, MedCLIP, and ConVIRT represent a promising direction in using contrastive learning frameworks to develop versatile, general-purpose models for medical imaging. However, while these models excel in producing generalizable feature representations, they still face limitations when compared to models trained specifically for single tasks. For example, a model trained and optimized solely for segmentation or classification typically outperforms general models on that specific task due to task-specific fine-tuning. Future research could investigate methods for seamlessly integrating task-specific optimizations within a generalized model framework, allowing Hybrid-ConVIRT and similar models to achieve competitive performance across various tasks without sacrificing versatility.

Overall, Hybrid-ConVIRT has shown itself to be a strong model in its current form, successfully merging components from MedCLIP and ConVIRT to create a Fused Similarity Matrix for improved representation learning. However, to fully assess its potential and limitations, more extensive testing and optimization will be essential. By fine-tuning parameters like α, expanding training datasets, applying advanced augmentation techniques, and evaluating performance across emerging models and diverse tasks, Hybrid-ConVIRT can continue to evolve as a robust tool for medical image analysis.

As the field progresses, the adaptability of models like Hybrid-ConVIRT to different datasets, tasks, and imaging contexts will be critical in determining their real-world utility. Such improvements can help shape a future in which medical imaging analysis is

faster, more accurate, and more adaptable to the complex and evolving needs of clinical practice.

Bibliography

[1] Amro Abbas and Stéphane Deny. Progress and limitations of deep networks to recognize objects in unusual poses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 160–168, 2023.

[2] Kulsoom Abdullah. Chexpert multiclass multilabel repository. https://github.com/kulsoom-abdullah/chexpert_multiclass_multilabel, 2023. Accessed: 2024-10-19.

[3] Emily Alsentzer, John R Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew McDermott. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[4] Hangbo Bao, Wenhui Wang, Li Dong, Qiang Liu, Owais Khan Mohammed, Kriti Aggarwal, Subhojit Som, and Furu Wei. Vlmo: Unified vision-language pre-training with mixture-of-modality-experts. *arXiv preprint arXiv:2111.02358*, 2021.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] Benedikt Boecking, Naoto Usuyama, Shruthi Bannur, Daniel C Castro, Anton Schwaighofer, Stephanie Hyland, Maria Wetscherek, Tristan Naumann, Aditya Nori, Javier Alvarez-Valle, et al. Making the most of text semantics to improve biomedical vision–language processing. In *European conference on computer vision*, pages 1–21. Springer, 2022.

[7] Natalia Buda and Wojciech Kosiak. Is a linear probe helpful in diagnosing diseases of pulmonary interstitial spaces? *Journal of Ultrasonography*, 17(69):136–141, 2017.

[8] Axel Saul Bustos, Arianne Pertusa, Jose-Manuel Salinas, and Maria de la Iglesia-Vayá. Padchest: A large chest x-ray image dataset with multi-label annotated reports. https://bimcv.cipf.es/bimcv-projects/padchest/, 2020. Accessed: 2024-10-16.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[10] Md. Ejazul Haque Chowdhury, Tawsifur Rahman, Amith Khandakar, Rashid Mazhar, Muhammad Abu Bakr Siddique Kadir, Zannatul Faria Mahbub, Md. Toufikul Islam, Muhammad Zahid Khan, Ayesha Iqbal, Niousha Arabi Hadi Emadi, Mahmood Bin Ibne Reaz, et al. COVID-19 radiography database. Online resource: https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database, 2020. Accessed on [your access date].

[11] Joseph Paul Cohen. Covid-19 chest x-ray dataset. https://github.com/ieee8023/covid-chestxray-dataset, 2020. Accessed: 2024-10-16.

[12] Datagy. Python optuna: A guide to hyperparameter optimization. https://datagy.io/python-optuna/, 2023. 2023-03-21.

[13] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold,

Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[15] AI Druce. Understanding classification thresholds using isocurves, 2021. Accessed: 2024-10-29.

[16] European Society of Radiology (ESR). Eurorad radiology case database. https://eurorad.org. Accessed: 2024-10-16.

[17] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[19] Hashnode. Optuna vs hyperopt: Which hyperparameter optimization library should you choose? https://hashnode.com/post/optuna-vs-hyperopt-which-hyperparameter-optimization-library-should-you-choose, 2020. 2020-01-14.

[20] Haibo He and Yunqian Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. Wiley-IEEE Press, 2013. Threshold tuning in imbalanced classification.

[21] Shih-Cheng Huang, Liyue Shen, Matthew P Lungren, and Serena Yeung. Gloria: A multimodal global-local representation learning framework for label-efficient medical image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3942–3951, 2021.

[22] J. D. Hunter. Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9(3):90–95, 2007. Accessed: 2024-10-16.

[23] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, Jacob Seekins, David A. Mong, Safwan S. Halabi, Jason K. Sandberg, Roger Jones, David B. Larson, Curtis P. Langlotz, Bhavik N. Patel, Matthew P. Lungren, and Andrew Y. Ng. CheXpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

[24] Italian Society of Medical and Interventional Radiology (SIRM). Covid-19 database. https://sirm.org/category/senza-categoria/covid-19/, 2020. Accessed: 2024-10-16.

[25] Woojeong Jin, Yu Cheng, Yelong Shen, Weizhu Chen, and Xiang Ren. A good prompt is worth millions of parameters: Low-resource prompt-based learning for vision-language models. *arXiv preprint arXiv:2110.08484*, 2021.

[26] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18:1–5, 2017. Accessed: 2024-10-16.

[27] Zachary C. Lipton, Charles Elkan, and Balakrishnan Naryanaswamy. Optimal thresholding of classifiers to maximize f1 measure. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 225–239, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[28] Microsoft. Biomedvlp-cxr-bert-specialized. https://huggingface.co/microsoft/BiomedVLP-CXR-BERT-specialized, 2024. Accessed: 2024-09-30.

[29] Microsoft Research. Biomedvlp-biovil-t: A vision-language pretraining model for biomedical tasks. https://huggingface.co/microsoft/BiomedVLP-BioViL-T, 2022. Accessed: 2024-10-16.

[30] ML Workgroup. Covid-19 image repository. https://github.com/ml-workgroup/covid-19-image-repository/tree/master/png, 2020. Accessed: 2024-10-16.

[31] National Institute of Allergy and Infectious Diseases (NIAID). Niaid tuberculosis chest x-ray dataset. https://www.niaid.nih.gov. Accessed: 2024-10-16.

[32] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. https://pytorch.org, 2019. Accessed: 2024-10-16.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Accessed: 2024-10-16.

[35] Ploomber. Micro vs. macro vs. weighted averages in classification, 2023. Accessed: 2024-10-24.

[36] Yurui Qian, Yu Wang, and Jingyang Lin. Enhancing the linear probing performance of masked auto-encoders. In *International Conference on Pattern Recognition*, pages 289–301. Springer, 2022.

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. https://github.com/openai/CLIP, 2021. Accessed: 2024-10-16.

[39] Radiological Society of North America (RSNA). RSNA pneumonia detection challenge dataset. Online resource: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data, 2018. Accessed on [your access date].

[40] Radiological Society of North America (RSNA). Rsna pneumonia detection challenge dataset. https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data, 2018. Accessed: 2024-10-16.

[41] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. *Advances in neural information processing systems*, 32, 2019.

[42] Tawsifur Rahman, Amith Khandakar, Muhammad A. Kadir, Khandaker R. Islam, Khandaker F. Islam, Zaid B. Mahbub, Mohamed Arselene Ayari, and Muhammad E. H. Chowdhury. Reliable tuberculosis detection using chest x-ray with deep learning, segmentation and visualization. *IEEE Access*, 8:191586–191601, 2020.

[43] Tawsifur Rahman, Amith Khandakar, Muhammad A. Kadir, Khandaker R. Islam, Khandaker F. Islam, Zaid B. Mahbub, Mohamed Arselene Ayari, and Muhammad

E. H. Chowdhury. Reliable tuberculosis detection using chest x-ray with deep learning, segmentation and visualization. *IEEE Access*, 2020. Accepted.

[44] Mert Bulent Sariyildiz, Julien Perez, and Diane Larlus. Learning visual representations with caption annotations. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 153–170. Springer, 2020.

[45] Preet Viradiya. Covid-19 radiography dataset. https://www.kaggle.com/datasets/preetviradiya/covid19-radiography-dataset, 2020. Accessed: 2024-10-16.

[46] Xiaosong Wang, Yong Peng, Le Lu, Zhiyong Lu, Mojdeh Bagheri, and Ronald M Summers. Chestx-ray14: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *arXiv preprint arXiv:1705.02315*, 2017.

[47] Zifan Wang and collaborators. Medclip: Contrastive learning from medical images and reports. https://github.com/RyanWangZf/MedCLIP, 2022. Accessed: 2024-10-16.

[48] Zifeng Wang, Zhenbang Wu, Dinesh Agarwal, and Jimeng Sun. Medclip: Contrastive learning from unpaired medical images and text. *arXiv preprint arXiv:2210.10163*, 2022.

[49] Yuan Yao, Ao Zhang, Zhengyan Zhang, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Cpt: Colorful prompt tuning for pre-trained vision-language models. *AI Open*, 5:30–38, 2024.

[50] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D Manning, and Curtis P Langlotz. Contrastive learning of medical visual representations from paired images and text. In *Machine Learning for Healthcare Conference*, pages 2–25. PMLR, 2022.

A Supplementary Results

Testing on the CoVID Dataset

Table 1: **Metrics for 5 Folds on the CoVID Dataset for BioVIL**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.817	0.782	0.852	0.794	670	1586	452	54
	0.50	0.877	0.834	0.874	0.850	629	1792	246	95
	0.65	0.892	0.880	0.832	0.852	511	1954	84	213
2	0.40	0.818	0.782	0.851	0.794	665	1593	446	58
	0.50	0.873	0.830	0.869	0.845	622	1789	250	101
	0.65	0.898	0.890	0.838	0.859	514	1966	73	209
3	0.40	0.838	0.797	0.860	0.813	656	1658	381	67
	0.50	0.885	0.846	0.869	0.856	605	1839	200	118
	0.65	0.893	0.889	0.826	0.850	495	1972	67	228
4	0.40	0.828	0.792	0.861	0.806	672	1615	423	51
	0.50	0.878	0.836	0.871	0.851	619	1806	232	104
	0.65	0.891	0.888	0.822	0.847	488	1973	65	235
5	0.40	0.836	0.798	0.867	0.813	675	1632	406	48
	0.50	0.887	0.848	0.876	0.860	617	1833	205	106
	0.65	0.899	0.900	0.832	0.858	499	1984	54	224

Table 2: **Linear Probe on BioVIL Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	11046	2762	16308	1	0.653
				2	0.579
				3	0.528
				4	0.490
				5	0.460
				6	0.436
				7	0.416
				8	0.399
				9	0.384
				10	0.371
2	11046	2762	16306	1	0.653
				2	0.583
				3	0.533
				4	0.496
				5	0.465
				6	0.441
				7	0.420
				8	0.402
				9	0.387
				10	0.373
3	11046	2762	16306	1	0.650
				2	0.581
				3	0.531
				4	0.493
				5	0.463
				6	0.438
				7	0.417
				8	0.400
				9	0.385
				10	0.371
4	11047	2761	16308	1	0.651
				2	0.582
				3	0.532
				4	0.494
				5	0.465
				6	0.441
				7	0.421
				8	0.403
				9	0.388
				10	0.375
5	11047	2761	16308	1	0.652
				2	0.580
				3	0.530
				4	0.493
				5	0.463
				6	0.439
				7	0.419
				8	0.401
				9	0.386
				10	0.373

Table 3: **Metrics for 5 Folds on the CoVID Dataset for MedCLIP**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.875	0.833	0.879	0.850	642	1776	262	82
	0.50	0.894	0.857	0.878	0.867	612	1857	181	112
	0.65	0.907	0.889	0.864	0.875	560	1944	94	164
2	0.40	0.874	0.831	0.875	0.848	635	1778	261	88
	0.50	0.900	0.865	0.885	0.874	617	1868	171	106
	0.65	0.911	0.893	0.873	0.882	572	1945	94	151
3	0.40	0.856	0.814	0.876	0.832	664	1699	340	59
	0.50	0.888	0.847	0.891	0.864	649	1803	236	74
	0.65	0.913	0.887	0.891	0.889	610	1913	126	113
4	0.40	0.872	0.829	0.879	0.847	646	1761	277	77
	0.50	0.894	0.856	0.883	0.868	623	1845	193	100
	0.65	0.917	0.899	0.882	0.890	585	1946	92	138
5	0.40	0.883	0.842	0.885	0.859	642	1797	241	81
	0.50	0.900	0.866	0.885	0.875	617	1869	169	106
	0.65	0.916	0.898	0.880	0.888	582	1946	92	141

Table 4: **Linear Probe on MedCLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	11046	2762	16308	1	0.427
				2	0.356
				3	0.335
				4	0.320
				5	0.306
				6	0.297
				7	0.291
				8	0.284
				9	0.278
				10	0.274
2	11046	2762	16306	1	0.430
				2	0.361
				3	0.340
				4	0.324
				5	0.312
				6	0.304
				7	0.294
				8	0.288
				9	0.283
				10	0.279
3	11046	2762	16306	1	0.405
				2	0.355
				3	0.334
				4	0.316
				5	0.305
				6	0.297
				7	0.291
				8	0.284
				9	0.280
				10	0.276
4	11047	2761	16308	1	0.402
				2	0.355
				3	0.333
				4	0.318
				5	0.307
				6	0.297
				7	0.289
				8	0.283
				9	0.278
				10	0.273
5	11047	2761	16308	1	0.408
				2	0.357
				3	0.336
				4	0.320
				5	0.308
				6	0.300
				7	0.293
				8	0.287
				9	0.282
				10	0.278

Table 5: **Metrics for 5 Folds on the CoVID Dataset for Hybrid-ConVIRT**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.934	0.924	0.904	0.913	609	1972	66	115
	0.50	0.934	0.933	0.894	0.911	586	1994	44	138
	0.65	0.929	0.938	0.877	0.902	556	2011	27	168
2	0.40	0.853	0.816	0.889	0.833	696	1661	378	27
	0.50	0.881	0.840	0.903	0.860	686	1747	292	37
	0.65	0.912	0.876	0.912	0.891	660	1858	181	63
3	0.40	0.928	0.899	0.923	0.910	659	1905	134	64
	0.50	0.937	0.916	0.922	0.919	644	1943	96	79
	0.65	0.941	0.934	0.912	0.922	614	1986	53	109
4	0.40	0.920	0.887	0.918	0.901	660	1880	158	63
	0.50	0.930	0.906	0.917	0.911	642	1927	111	81
	0.65	0.941	0.934	0.910	0.921	611	1986	52	112
5	0.40	0.936	0.918	0.917	0.917	633	1952	86	90
	0.50	0.943	0.935	0.915	0.924	620	1983	55	103
	0.65	0.941	0.944	0.900	0.919	590	2007	31	133

Table 6: **Linear Probe on Hybrid-ConVIRT Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	2688	672	4480	1	0.357
				2	0.228
				3	0.202
				4	0.185
				5	0.174
				6	0.162
				7	0.154
				8	0.142
				9	0.138
				10	0.134
2	2688	672	4480	1	0.331
				2	0.231
				3	0.203
				4	0.183
				5	0.168
				6	0.170
				7	0.150
				8	0.141
				9	0.135
				10	0.126
3	2688	672	4480	1	0.283
				2	0.212
				3	0.188
				4	0.176
				5	0.159
				6	0.154
				7	0.140
				8	0.135
				9	0.127
				10	0.120
4	2688	672	4480	1	0.363
				2	0.239
				3	0.210
				4	0.192
				5	0.175
				6	0.164
				7	0.158
				8	0.147
				9	0.140
				10	0.133
5	2688	672	4480	1	0.278
				2	0.215
				3	0.186
				4	0.174
				5	0.156
				6	0.144
				7	0.139
				8	0.129
				9	0.122
				10	0.119

Table 7: **Metrics for 5 Folds on the CoVID Dataset for CLIP**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.781	0.717	0.716	0.716	420	1737	301	304
	0.50	0.797	0.744	0.688	0.706	333	1867	171	391
	0.65	0.795	0.787	0.638	0.657	222	1974	64	502
2	0.40	0.807	0.758	0.707	0.725	359	1870	169	364
	0.50	0.814	0.802	0.679	0.705	286	1961	78	437
	0.65	0.795	0.834	0.620	0.635	183	2014	25	540
3	0.40	0.774	0.723	0.763	0.734	534	1604	435	189
	0.50	0.806	0.749	0.744	0.747	444	1783	256	279
	0.65	0.811	0.782	0.687	0.711	308	1931	108	415
4	0.40	0.805	0.750	0.722	0.733	395	1828	210	328
	0.50	0.805	0.773	0.678	0.701	297	1926	112	426
	0.65	0.792	0.812	0.618	0.632	184	2002	36	539
5	0.40	0.562	0.666	0.688	0.561	689	864	1174	34
	0.50	0.643	0.674	0.722	0.631	641	1135	903	82
	0.65	0.746	0.703	0.750	0.711	549	1511	527	174

Table 8: **Linear Probe on CLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	2688	672	4480	1	0.457
				2	0.439
				3	0.420
				4	0.384
				5	0.367
				6	0.350
				7	0.341
				8	0.318
				9	0.305
				10	0.311
2	2688	672	4480	1	0.457
				2	0.412
				3	0.386
				4	0.365
				5	0.344
				6	0.326
				7	0.318
				8	0.318
				9	0.309
				10	0.293
3	2688	672	4480	1	0.485
				2	0.427
				3	0.403
				4	0.375
				5	0.359
				6	0.341
				7	0.331
				8	0.324
				9	0.319
				10	0.309
4	2688	672	4480	1	0.547
				2	0.432
				3	0.413
				4	0.389
				5	0.370
				6	0.346
				7	0.364
				8	0.325
				9	0.314
				10	0.313
5	2688	672	4480	1	0.466
				2	0.414
				3	0.389
				4	0.381
				5	0.347
				6	0.332
				7	0.320
				8	0.326
				9	0.296
				10	0.289

Testing on the Tuberculosis Dataset

Table 9: **Metrics for 5 Folds on the Tuberculosis Dataset for BioVIL**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.936	0.932	0.829	0.870	75	554	6	37
	0.55	0.932	0.955	0.798	0.852	67	559	1	45
	0.65	0.926	0.959	0.777	0.835	62	560	0	50
2	0.40	0.936	0.941	0.822	0.868	73	556	4	39
	0.55	0.932	0.962	0.795	0.851	66	560	0	46
	0.65	0.914	0.953	0.741	0.801	54	560	0	58
3	0.40	0.935	0.935	0.821	0.865	73	555	5	39
	0.55	0.917	0.955	0.750	0.810	56	560	0	56
	0.65	0.909	0.951	0.728	0.787	51	560	0	61
4	0.40	0.935	0.918	0.836	0.870	77	551	9	35
	0.55	0.933	0.950	0.806	0.858	69	558	2	43
	0.65	0.933	0.963	0.799	0.855	67	560	0	45
5	0.40	0.942	0.951	0.837	0.881	76	557	3	36
	0.55	0.924	0.958	0.772	0.831	61	560	0	51
	0.65	0.915	0.954	0.746	0.805	55	560	0	57

Table 10: **Linear Probe on BioVIL Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	2688	672	4480	1	0.421
				2	0.344
				3	0.297
				4	0.268
				5	0.243
				6	0.228
				7	0.217
				8	0.206
				9	0.196
				10	0.189
2	2688	672	4480	1	0.438
				2	0.359
				3	0.309
				4	0.275
				5	0.250
				6	0.233
				7	0.219
				8	0.209
				9	0.199
				10	0.191
3	2688	672	4480	1	0.433
				2	0.352
				3	0.302
				4	0.269
				5	0.249
				6	0.231
				7	0.218
				8	0.207
				9	0.197
				10	0.191
4	2688	672	4480	1	0.411
				2	0.336
				3	0.291
				4	0.261
				5	0.240
				6	0.224
				7	0.211
				8	0.202
				9	0.195
				10	0.186
5	2688	672	4480	1	0.420
				2	0.343
				3	0.297
				4	0.267
				5	0.244
				6	0.226
				7	0.215
				8	0.204
				9	0.195
				10	0.187

Table 11: **Metrics for 5 Folds on the Tuberculosis Dataset for MedCLIP**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.960	0.955	0.897	0.923	90	555	5	22
	0.55	0.951	0.957	0.863	0.902	82	557	3	30
	0.65	0.948	0.965	0.847	0.893	78	559	1	34
2	0.40	0.969	0.948	0.938	0.943	100	551	9	12
	0.55	0.969	0.965	0.921	0.941	95	556	4	17
	0.65	0.960	0.959	0.894	0.922	89	556	4	23
3	0.40	0.954	0.959	0.872	0.909	84	557	3	28
	0.55	0.952	0.963	0.864	0.904	82	558	2	30
	0.65	0.949	0.966	0.852	0.897	79	559	1	33
4	0.40	0.964	0.966	0.904	0.931	91	557	3	21
	0.55	0.954	0.969	0.865	0.907	82	559	1	30
	0.65	0.948	0.971	0.844	0.892	77	560	0	35
5	0.40	0.967	0.953	0.927	0.939	97	553	7	15
	0.55	0.960	0.959	0.894	0.922	89	556	4	23
	0.65	0.955	0.960	0.877	0.912	85	557	3	27

Table 12: **Linear Probe on MedCLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	2688	672	4480	1	0.275
				2	0.204
				3	0.176
				4	0.159
				5	0.151
				6	0.135
				7	0.128
				8	0.129
				9	0.113
				10	0.107
2	2688	672	4480	1	0.286
				2	0.211
				3	0.181
				4	0.161
				5	0.153
				6	0.140
				7	0.129
				8	0.125
				9	0.116
				10	0.120
3	2688	672	4480	1	0.268
				2	0.199
				3	0.180
				4	0.165
				5	0.146
				6	0.143
				7	0.127
				8	0.135
				9	0.124
				10	0.110
4	2688	672	4480	1	0.267
				2	0.209
				3	0.180
				4	0.158
				5	0.145
				6	0.138
				7	0.130
				8	0.121
				9	0.117
				10	0.107
5	2688	672	4480	1	0.292
				2	0.209
				3	0.180
				4	0.164
				5	0.164
				6	0.139
				7	0.127
				8	0.129
				9	0.113
				10	0.108

Table 13: **Metrics for 5 Folds on the Tuberculosis Dataset for Hybrid-ConVIRT**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.954	0.923	0.908	0.915	94	547	13	18
	0.55	0.951	0.940	0.878	0.905	86	553	7	26
	0.65	0.948	0.955	0.855	0.895	80	557	3	32
2	0.40	0.960	0.944	0.908	0.925	93	552	8	19
	0.55	0.958	0.954	0.893	0.920	89	555	5	23
	0.65	0.954	0.954	0.876	0.909	85	556	4	27
3	0.40	0.952	0.958	0.868	0.905	83	557	3	29
	0.55	0.945	0.958	0.842	0.887	77	558	2	35
	0.65	0.944	0.962	0.834	0.883	75	559	1	37
4	0.40	0.972	0.963	0.933	0.947	98	555	5	14
	0.55	0.966	0.971	0.905	0.934	91	558	2	21
	0.65	0.955	0.965	0.873	0.911	84	558	2	28
5	0.40	0.964	0.954	0.914	0.933	94	554	6	18
	0.55	0.958	0.958	0.889	0.919	88	556	4	24
	0.65	0.952	0.958	0.868	0.905	83	557	3	29

Table 14: **Linear Probe on Hybrid-ConVIRT Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	2688	672	4480	1	0.357
				2	0.228
				3	0.202
				4	0.185
				5	0.174
				6	0.162
				7	0.154
				8	0.142
				9	0.138
				10	0.134
2	2688	672	4480	1	0.331
				2	0.231
				3	0.203
				4	0.183
				5	0.168
				6	0.170
				7	0.150
				8	0.141
				9	0.135
				10	0.126
3	2688	672	4480	1	0.283
				2	0.212
				3	0.188
				4	0.176
				5	0.159
				6	0.154
				7	0.140
				8	0.135
				9	0.127
				10	0.120
4	2688	672	4480	1	0.363
				2	0.239
				3	0.210
				4	0.192
				5	0.175
				6	0.164
				7	0.158
				8	0.147
				9	0.140
				10	0.133
5	2688	672	4480	1	0.278
				2	0.215
				3	0.186
				4	0.174
				5	0.156
				6	0.144
				7	0.139
				8	0.129
				9	0.122
				10	0.119

Table 15: **Metrics for 5 Folds on the Tuberculosis Dataset for CLIP**

Fold	Threshold	Accuracy	Precision	Recall	F1 Score	TP	TN	FP	FN
1	0.40	0.899	0.862	0.743	0.784	57	547	13	55
	0.55	0.880	0.937	0.638	0.683	31	560	0	81
	0.65	0.857	0.927	0.571	0.586	16	560	0	96
2	0.40	0.903	0.864	0.760	0.798	61	546	14	51
	0.55	0.880	0.937	0.638	0.683	31	560	0	81
	0.65	0.854	0.926	0.563	0.571	14	560	0	98
3	0.40	0.900	0.898	0.722	0.773	51	554	6	61
	0.55	0.872	0.933	0.616	0.653	26	560	0	86
	0.65	0.845	0.922	0.536	0.524	8	560	0	104
4	0.40	0.887	0.797	0.793	0.795	73	523	37	39
	0.55	0.884	0.894	0.666	0.714	38	556	4	74
	0.65	0.857	0.927	0.571	0.586	16	560	0	96
5	0.40	0.908	0.851	0.798	0.821	71	539	21	41
	0.55	0.884	0.926	0.655	0.704	35	559	1	77
	0.65	0.866	0.931	0.598	0.627	22	560	0	90

Table 16: **Linear Probe on CLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	2688	672	4480	1	0.457
				2	0.439
				3	0.420
				4	0.384
				5	0.367
				6	0.350
				7	0.341
				8	0.318
				9	0.305
				10	0.311
2	2688	672	4480	1	0.457
				2	0.412
				3	0.386
				4	0.365
				5	0.344
				6	0.326
				7	0.318
				8	0.318
				9	0.309
				10	0.293
3	2688	672	4480	1	0.485
				2	0.427
				3	0.403
				4	0.375
				5	0.359
				6	0.341
				7	0.331
				8	0.324
				9	0.319
				10	0.309
4	2688	672	4480	1	0.547
				2	0.432
				3	0.413
				4	0.389
				5	0.370
				6	0.346
				7	0.364
				8	0.325
				9	0.314
				10	0.313
5	2688	672	4480	1	0.466
				2	0.414
				3	0.389
				4	0.381
				5	0.347
				6	0.332
				7	0.320
				8	0.326
				9	0.296
				10	0.289

Pneumonia/TB Mix Dataset

Table 17: **Metrics for 5 Folds on Pneumonia dataset for BioVIL**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.755	0.733	0.812	0.762	(1445, 482, 115) (229, 916, 10) (14, 5, 278)
2	0.753	0.720	0.808	0.753	(1479, 427, 135) (269, 870, 16) (15, 1, 282)
3	0.751	0.721	0.801	0.751	(1450, 468, 123) (236, 901, 18) (23, 3, 272)
4	0.755	0.727	0.804	0.756	(1478, 447, 116) (252, 885, 18) (21, 2, 275)
5	0.747	0.722	0.807	0.752	(1425, 491, 125) (234, 907, 14) (18, 1, 279)

Table 18: **Linear Probe on BioVIL Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	13976	3494	24492	1	0.944
				2	0.779
				3	0.691
				4	0.634
				5	0.594
				6	0.564
				7	0.541
				8	0.522
				9	0.507
				10	0.495
2	13976	3494	24495	1	0.961
				2	0.789
				3	0.697
				4	0.637
				5	0.596
				6	0.565
				7	0.541
				8	0.523
				9	0.508
				10	0.495
3	13976	3494	24495	1	0.952
				2	0.783
				3	0.692
				4	0.634
				5	0.592
				6	0.562
				7	0.538
				8	0.519
				9	0.504
				10	0.491
4	13976	3494	24495	1	0.945
				2	0.775
				3	0.686
				4	0.628
				5	0.588
				6	0.558
				7	0.535
				8	0.517
				9	0.502
				10	0.490
5	13976	3494	24495	1	0.947
				2	0.776
				3	0.686
				4	0.628
				5	0.588
				6	0.557
				7	0.534
				8	0.516
				9	0.501
				10	0.488

Table 19: **Metrics for 5 Folds on Pneumonia dataset for MedCLIP**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.795	0.732	0.808	0.757	(1526, 381, 135) (87, 1014, 54) (44, 16, 237)
2	0.796	0.734	0.814	0.761	(1541, 365, 135) (107, 992, 56) (35, 16, 247)
3	0.799	0.728	0.821	0.758	(1598, 277, 166) (141, 934, 80) (33, 5, 260)
4	0.800	0.737	0.827	0.767	(1567, 329, 145) (126, 968, 61) (30, 7, 261)
5	0.793	0.732	0.819	0.761	(1540, 364, 137) (119, 974, 62) (30, 12, 256)

Table 20: **Linear Probe on MedCLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	13976	3494	24492	1	0.705
				2	0.565
				3	0.531
				4	0.508
				5	0.490
				6	0.477
				7	0.465
				8	0.455
				9	0.447
				10	0.439
2	13976	3494	24495	1	0.696
				2	0.571
				3	0.537
				4	0.515
				5	0.498
				6	0.485
				7	0.473
				8	0.464
				9	0.455
				10	0.448
3	13976	3494	24495	1	0.705
				2	0.574
				3	0.539
				4	0.515
				5	0.498
				6	0.483
				7	0.471
				8	0.460
				9	0.452
				10	0.444
4	13976	3494	24495	1	0.730
				2	0.573
				3	0.538
				4	0.515
				5	0.497
				6	0.483
				7	0.472
				8	0.462
				9	0.454
				10	0.446
5	13976	3494	24495	1	0.704
				2	0.564
				3	0.530
				4	0.507
				5	0.490
				6	0.476
				7	0.464
				8	0.455
				9	0.446
				10	0.438

Table 21: **Metrics for 5 Folds on Pneumonia dataset for Hybrid-ConVIRT**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.821	0.777	0.850	0.805	(1603, 341, 98) (123, 1000, 32) (18, 12, 267)
2	0.824	0.794	0.826	0.808	(1672, 311, 58) (168, 961, 26) (38, 14, 246)
3	0.821	0.769	0.830	0.794	(1688, 264, 89) (177, 925, 53) (35, 6, 257)
4	0.806	0.755	0.852	0.786	(1521, 385, 135) (96, 1017, 42) (17, 4, 277)
5	0.793	0.732	0.840	0.764	(1510, 363, 168) (111, 982, 62) (15, 6, 277)

Table 22: **Linear Probe on Hybrid-ConVIRT Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	13976	3494	24492	1	0.575
				2	0.478
				3	0.440
				4	0.421
				5	0.403
				6	0.393
				7	0.385
				8	0.378
				9	0.369
				10	0.367
2	13976	3494	24495	1	0.582
				2	0.485
				3	0.449
				4	0.429
				5	0.415
				6	0.402
				7	0.393
				8	0.388
				9	0.380
				10	0.374
3	13976	3494	24495	1	0.593
				2	0.485
				3	0.447
				4	0.422
				5	0.410
				6	0.398
				7	0.386
				8	0.379
				9	0.374
				10	0.368
4	13976	3494	24495	1	0.579
				2	0.480
				3	0.445
				4	0.424
				5	0.409
				6	0.398
				7	0.386
				8	0.382
				9	0.373
				10	0.368
5	13976	3494	24495	1	0.580
				2	0.475
				3	0.439
				4	0.418
				5	0.401
				6	0.391
				7	0.382
				8	0.376
				9	0.368

Table 23: **Metrics for 5 Folds on the Pneumonia Dataset for CLIP**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.689	0.631	0.743	0.659	(1394, 388, 260) (330, 746, 79) (26, 4, 267)
2	0.714	0.667	0.731	0.691	(1506, 399, 136) (355, 750, 50) (52, 6, 240)
3	0.591	0.595	0.724	0.576	(821, 822, 398) (95, 967, 93) (10, 10, 278)
4	0.660	0.657	0.703	0.656	(1116, 798, 127) (153, 978, 24) (52, 33, 213)
5	0.541	0.608	0.683	0.545	(583, 1208, 250) (42, 1052, 61) (11, 33, 254)

Table 24: **Linear Probe on CLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	13976	3494	24492	1	0.889
				2	0.770
				3	0.743
				4	0.703
				5	0.687
				6	0.679
				7	0.663
				8	0.654
				9	0.645
				10	0.640
2	13976	3494	24495	1	0.887
				2	0.775
				3	0.729
				4	0.704
				5	0.685
				6	0.678
				7	0.667
				8	0.648
				9	0.651
				10	0.634
3	13976	3494	24495	1	0.886
				2	0.768
				3	0.725
				4	0.698
				5	0.675
				6	0.664
				7	0.649
				8	0.646
				9	0.657
				10	0.638
4	13976	3494	24495	1	0.888
				2	0.761
				3	0.727
				4	0.698
				5	0.685
				6	0.669
				7	0.657
				8	0.649
				9	0.639
				10	0.635
5	13976	3494	24495	1	0.881
				2	0.756
				3	0.724
				4	0.712
				5	0.690
				6	0.660
				7	0.662
				8	0.645
				9	0.650
				10	0.638

CheXpert (5 Classes)

Table 25: **Metrics for 5 Folds on the CheXpert Dataset for BioVIL**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.275	0.309	0.297	0.258	(85, 35, 37, 15, 136) (62, 213, 49, 41, 227) (16, 20, 28, 22, 54) (94, 137, 84, 93, 231) (34, 25, 22, 14, 96)
2	0.318	0.308	0.312	0.282	(181, 33, 20, 28, 46) (175, 191, 58, 78, 90) (62, 16, 22, 20, 20) (204, 116, 75, 152, 92) (70, 27, 21, 24, 49)
3	0.201	0.298	0.270	0.192	(20, 16, 183, 31, 59) (21, 131, 306, 66, 68) (2, 12, 101, 13, 12) (22, 54, 382, 82, 98) (6, 12, 111, 20, 41)
4	0.380	0.308	0.291	0.249	(139, 135, 0, 24, 11) (81, 463, 7, 35, 6) (31, 77, 3, 21, 8) (143, 373, 12, 96, 14) (63, 98, 1, 18, 10)
5	0.392	0.336	0.289	0.269	(14, 60, 12, 149, 74) (6, 253, 16, 238, 79) (1, 26, 9, 75, 29) (5, 137, 19, 405, 72) (5, 37, 7, 89, 52)

Table 26: **Linear Probe on BioVIL Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	7477	1870	12765	1	1.674
				2	1.589
				3	1.559
				4	1.601
				5	1.519
				6	1.542
				7	1.495
				8	1.459
				9	1.493
				10	1.473
				11	1.502
				12	1.449
				13	1.453
				14	1.461
				15	1.468
				16	1.454
				17	1.437
				18	1.450
				19	1.452
				20	1.405
2	7477	1870	12765	1	1.695
				2	1.594
				3	1.578
				4	1.551
				5	1.570
				6	1.510
				7	1.481
				8	1.499
				9	1.433
				10	1.481
				11	1.499
				12	1.429
				13	1.479
				14	1.443
				15	1.451
				16	1.469
				17	1.418
				18	1.437
				19	1.433
				20	1.416
3	7478	1869	12770	1	1.672
				2	1.609
				3	1.534
				4	1.572
				5	1.528
				6	1.547
				7	1.496
				8	1.470
				9	1.478
				10	1.458
				11	1.508
				12	1.438
				13	1.476
				14	1.440
				15	1.460
				16	1.455
				17	1.448
				18	1.424
				19	1.459
				20	1.429
4	7478	1869	12770	1	1.723
				2	1.602
				3	1.562
				4	1.514
				5	1.523
				6	1.538
				7	1.486
				8	1.500
				9	1.492
				10	1.492
				11	1.457
				12	1.466
				13	1.442
				14	1.485
				15	1.465
				16	1.456
				17	1.501
				18	1.434
				19	1.443
				20	1.441
5	7478	1869	12770	1	1.697
				2	1.608
				3	1.555
				4	1.557
				5	1.564
				6	1.522
				7	1.495
				8	1.566
				9	1.477
				10	1.474
				11	1.499
				12	1.504
				13	1.489
				14	1.459
				15	1.428
				16	1.462
				17	1.475
				18	1.481
				19	1.483
				20	1.462

Table 27: **Metrics for 5 Folds on the CheXpert Dataset for MedCLIP**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.566	0.516	0.538	0.515	[152, 20, 16, 31, 89] [25, 411, 30, 71, 55] [10, 17, 62, 22, 29] [69, 98, 58, 330, 84] [31, 16, 16, 24, 104]
2	0.541	0.504	0.505	0.472	[210, 25, 40, 23, 10] [46, 412, 52, 73, 9] [33, 6, 72, 23, 6] [144, 108, 90, 281, 16] [72, 18, 39, 25, 37]
3	0.516	0.488	0.498	0.457	[92, 21, 52, 40, 104] [12, 451, 40, 52, 37] [12, 24, 68, 11, 25] [34, 153, 97, 247, 107] [10, 32, 31, 11, 106]
4	0.551	0.519	0.492	0.478	[240, 10, 3, 22, 34] [87, 364, 10, 105, 26] [58, 11, 24, 30, 17] [183, 68, 18, 328, 41] [85, 8, 4, 20, 73]
5	0.512	0.507	0.537	0.485	[173, 7, 47, 19, 63] [39, 341, 86, 82, 44] [14, 4, 81, 11, 30] [89, 57, 158, 252, 82] [25, 4, 39, 13, 109]

Table 28: **Linear Probe on MedCLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	7477	1870	12765	1	1.266
				2	1.214
				3	1.215
				4	1.193
				5	1.182
				6	1.175
				7	1.169
				8	1.160
				9	1.170
				10	1.160
				11	1.166
				12	1.144
				13	1.141
				14	1.144
				15	1.145
				16	1.133
				17	1.131
				18	1.137
				19	1.141
				20	1.129
2	7477	1870	12765	1	1.245
				2	1.203
				3	1.185
				4	1.176
				5	1.172
				6	1.156
				7	1.157
				8	1.146
				9	1.151
				10	1.141
				11	1.144
				12	1.134
				13	1.137
				14	1.132
				15	1.120
				16	1.125
				17	1.119
				18	1.130
				19	1.126
				20	1.123
3	7478	1869	12770	1	1.253
				2	1.210
				3	1.192
				4	1.182
				5	1.186
				6	1.164
				7	1.167
				8	1.166
				9	1.152
				10	1.141
				11	1.142
				12	1.137
				13	1.126
				14	1.141
				15	1.129
				16	1.121
				17	1.130
				18	1.121
				19	1.123
				20	1.124
4	7478	1869	12770	1	1.247
				2	1.208
				3	1.191
				4	1.177
				5	1.178
				6	1.165
				7	1.181
				8	1.165
				9	1.157
				10	1.161
				11	1.142
				12	1.156
				13	1.146
				14	1.145
				15	1.135
				16	1.128
				17	1.131
				18	1.138
				19	1.129
				20	1.126
5	7478	1869	12770	1	1.251
				2	1.212
				3	1.198
				4	1.190
				5	1.166
				6	1.176
				7	1.165
				8	1.163
				9	1.156
				10	1.145
				11	1.142
				12	1.148
				13	1.143
				14	1.139
				15	1.140
				16	1.130
				17	1.124
				18	1.133
				19	1.116
				20	1.123

Table 29: **Metrics for 5 Folds on the CheXpert Dataset for Hybrid-ConVIRT**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.562	0.509	0.531	0.511	[177, 15, 14, 29, 73] [38, 374, 36, 94, 50] [16, 14, 60, 24, 26] [100, 71, 48, 350, 70] [46, 12, 16, 27, 90]
2	0.548	0.484	0.503	0.487	[145, 27, 30, 44, 62] [28, 395, 32, 108, 29] [13, 9, 63, 37, 18] [77, 92, 79, 349, 42] [36, 17, 38, 27, 73]
3	0.540	0.492	0.519	0.490	[152, 12, 47, 38, 60] [30, 410, 45, 83, 24] [18, 15, 72, 15, 20] [78, 111, 104, 293, 52] [19, 22, 46, 20, 83]
4	0.561	0.511	0.539	0.512	[156, 15, 24, 39, 75] [30, 374, 32, 107, 49] [12, 14, 56, 25, 33] [73, 75, 69, 344, 77] [22, 9, 19, 22, 118]
5	0.574	0.507	0.532	0.511	[190, 16, 19, 29, 55] [37, 424, 30, 67, 34] [24, 13, 52, 21, 30] [92, 99, 62, 321, 64] [47, 15, 22, 20, 86]

Table 30: **Linear Probe on Hybrid-ConVIRT Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	7477	1870	12765	1	1.461
				2	1.283
				3	1.226
				4	1.202
				5	1.189
				6	1.182
				7	1.178
				8	1.175
				9	1.172
				10	1.170
				11	1.168
				12	1.166
				13	1.164
				14	1.163
				15	1.162
				16	1.160
				17	1.159
				18	1.159
				19	1.157
				20	1.155
2	7477	1870	12765	1	1.458
				2	1.269
				3	1.214
				4	1.191
				5	1.180
				6	1.173
				7	1.169
				8	1.165
				9	1.162
				10	1.159
				11	1.158
				12	1.156
				13	1.154
				14	1.153
				15	1.150
				16	1.150
				17	1.148
				18	1.146
				19	1.147
				20	1.144
3	7478	1869	12770	1	1.423
				2	1.267
				3	1.215
				4	1.195
				5	1.185
				6	1.173
				7	1.172
				8	1.165
				9	1.162
				10	1.159
				11	1.157
				12	1.156
				13	1.155
				14	1.152
				15	1.155
				16	1.149
				17	1.148
				18	1.147
				19	1.145
				20	1.143
4	7478	1869	12770	1	1.474
				2	1.286
				3	1.226
				4	1.199
				5	1.191
				6	1.181
				7	1.174
				8	1.171
				9	1.171
				10	1.166
				11	1.164
				12	1.164
				13	1.159
				14	1.159
				15	1.156
				16	1.156
				17	1.157
				18	1.156
				19	1.153
				20	1.154
5	7478	1869	12770	1	1.446
				2	1.275
				3	1.221
				4	1.199
				5	1.187
				6	1.176
				7	1.173
				8	1.170
				9	1.164
				10	1.164
				11	1.161
				12	1.160
				13	1.157
				14	1.156
				15	1.155
				16	1.156
				17	1.152
				18	1.152
				19	1.149
				20	1.150

Table 31: **Metrics for 5 Folds on the CheXpert Dataset for CLIP**

Fold	Accuracy	Precision	Recall	F1 Score	Confusion Matrix
1	0.316	0.218	0.257	0.198	[173, 95, 0, 26, 14] [172, 358, 0, 49, 13] [46, 75, 0, 12, 7] [236, 333, 0, 52, 18] [81, 93, 0, 10, 7]
2	0.201	0.181	0.247	0.170	[190, 16, 16, 0, 86] [196, 107, 69, 0, 220] [52, 16, 16, 0, 56] [289, 57, 55, 0, 238] [84, 23, 22, 0, 62]
3	0.185	0.159	0.216	0.096	[298, 7, 4, 0, 0] [529, 42, 21, 0, 0] [132, 2, 6, 0, 0] [600, 22, 16, 0, 0] [173, 9, 8, 0, 0]
4	0.163	0.103	0.207	0.078	[286, 1, 21, 0, 1] [515, 4, 70, 0, 3] [123, 1, 14, 0, 2] [576, 5, 55, 0, 2] [164, 4, 22, 0, 0]
5	0.240	0.168	0.245	0.173	[61, 84, 0, 0, 164] [30, 284, 0, 0, 278] [10, 36, 0, 0, 94] [67, 219, 0, 0, 352] [23, 63, 0, 0, 104]

Table 32: **Linear Probe on CLIP Loss per Epoch for Each Fold**

Fold	Train Size (Before SMOTE)	Validation Size	Train Size (After SMOTE)	Epoch	Loss
1	7477	1870	12765	1	1.694
				2	1.698
				3	1.677
				4	1.676
				5	1.673
				6	1.662
				7	1.661
				8	1.665
				9	1.660
				10	1.668
				11	1.665
				12	1.659
				13	1.654
				14	1.648
				15	1.643
				16	1.638
				17	1.633
				18	1.628
				19	1.590
				20	1.548
2	7477	1870	12765	1	1.695
				2	1.684
				3	1.688
				4	1.679
				5	1.672
				6	1.666
				7	1.671
				8	1.651
				9	1.655
				10	1.649
				11	1.645
				12	1.640
				13	1.635
				14	1.631
				15	1.627
				16	1.621
				17	1.618
				18	1.614
				19	1.590
				20	1.578
3	7478	1869	12770	1	1.704
				2	1.673
				3	1.679
				4	1.668
				5	1.675
				6	1.667
				7	1.678
				8	1.666
				9	1.656
				10	1.654
				11	1.650
				12	1.645
				13	1.640
				14	1.635
				15	1.630
				16	1.625
				17	1.621
				18	1.599
				19	1.592
				20	1.552
4	7478	1869	12770	1	1.694
				2	1.696
				3	1.668
				4	1.676
				5	1.658
				6	1.684
				7	1.659
				8	1.652
				9	1.675
				10	1.671
				11	1.668
				12	1.662
				13	1.656
				14	1.652
				15	1.646
				16	1.642
				17	1.598
				18	1.574
				19	1.589
				20	1.548
5	7478	1869	12770	1	1.699
				2	1.683
				3	1.675
				4	1.670
				5	1.664
				6	1.657
				7	1.663
				8	1.673
				9	1.661
				10	1.647
				11	1.643
				12	1.637
				13	1.633
				14	1.629
				15	1.625
				16	1.621
				17	1.617
				18	1.599
				19	1.553
				20	1.521

B Hyper-parameter Tuning Logs

Table 33: Optuna 40 trials - Hyper-parameter tuning of ConVIRT

Trial	Loss	Learning rate	Batch Size	Number of Epoch
0	0.796	1.081e-05	16	26
1	2.866	7.151e-03	32	13
2	1.479	6.832e-04	8	21
3	2.173	9.852e-05	16	11
4	2.866	1.382e-03	32	13
5	2.866	2.507e-03	32	21
6	1.749	3.569e-05	32	24
7	2.173	6.767e-03	16	19
8	2.866	9.563e-05	32	30
9	1.479	4.724e-03	8	11
10	0.755	2.161e-05	16	30
11	0.728	1.130e-05	16	30
12	0.687	1.509e-05	16	30
13	2.173	1.111e-04	16	28
14	0.759	1.088e-05	16	26
15	1.718	3.624e-05	16	17
16	2.173	1.999e-04	16	28
17	0.704	2.924e-05	8	30
18	1.479	3.817e-05	8	26
19	1.479	3.612e-04	8	17
20	1.479	5.784e-05	8	28
21	0.330	1.865e-05	8	30
22	0.355	1.848e-05	8	30
23	0.355	1.858e-05	8	28
24	0.457	2.029e-05	8	24
25	1.479	6.434e-05	8	28
26	1.479	2.523e-04	8	26
27	0.474	2.190e-05	8	24
28	1.479	4.601e-05	8	28
29	0.404	1.631e-05	8	26
30	1.479	1.444e-04	8	30
31	0.364	1.657e-05	16	26
32	0.289	1.064e-05	16	24
33	0.283	1.072e-05	16	28
34	0.283	1.034e-05	16	28
35	0.350	1.213e-05	8	21
36	0.377	1.019e-05	8	28
37	1.338	2.797e-05	32	28
38	1.479	6.179e-04	8	26
39	1.404	6.819e-05	8	19

Table 34: Optuna 40 trials - Hyper-parameter tuning of Hybrid ConVIRT

Trial	Loss	Learning Rate (lr)	Batch Size	Temperature
0	2.733	0.002	31	0.077
1	2.389	0.000	22	0.083
2	2.281	0.001	16	0.070
3	2.478	0.000	24	0.056
4	2.667	0.000	29	0.013
5	2.191	0.000	18	0.082
6	2.765	0.002	32	0.084
7	2.478	0.002	24	0.012
8	0.423	0.000	16	0.010
9	2.519	0.000	25	0.038
10	0.543	0.000	21	0.051
11	0.348	0.000	16	0.028
12	0.349	0.000	16	0.030
13	0.565	0.000	16	0.030
14	0.593	0.000	19	0.034
15	2.691	0.010	19	0.026
16	2.243	0.000	19	0.048
17	1.624	0.000	16	0.025
18	2.436	0.007	21	0.063
19	2.342	0.001	21	0.065
20	2.595	0.006	27	0.096
21	0.422	0.000	18	0.023
22	1.730	0.000	18	0.038
23	0.570	0.000	22	0.044
24	0.455	0.000	18	0.023
25	0.457	0.000	17	0.023
26	0.849	0.000	20	0.019
27	0.493	0.000	17	0.031
28	0.239	0.000	16	0.058
29	2.073	0.000	16	0.061
30	1.062	0.000	20	0.069
31	2.243	0.000	19	0.042
32	0.671	0.000	26	0.074
33	2.133	0.001	17	0.057
34	0.382	0.000	17	0.017
35	0.499	0.000	17	0.016
36	2.073	0.000	16	0.028
37	0.537	0.000	22	0.035
38	0.658	0.000	17	0.052
39	2.296	0.000	20	0.017