# Multi-Label Generalized Zero Shot Learning for the Classification of Disease in Chest Radiographs - Reproduction

**Tyler Cheng[1], Kelvin Chong[2], Wen Tong[3]**

tcheng@illinois.com[1], kvchong2@illinois.edu[2], went2@illinois.edu[3]

## Abstract

This project successfully reproduces the findings of the paper, "Multi-Label Generalized Zero Shot Learning for the Classification of Disease in Chest Radiographs" by (Hayat, Nasir and Lashen, Hazem and Shamout, Farah E. 2021). The original work addresses a limitation in conventional supervised learning for medical imaging: the inability to predict unseen disease classes. The authors proposed a framework which trains on a set of seen disease classes and classifies images belonging to both seen and unseen classes during inference by leveraging semantic information.

Our reproduction effort utilized the NIH Chest X-ray Dataset and leveraged the authors' published GitHub repository to replicate the training and inference pipelines, including the use of BioBERT for extracting semantic features and their implementation of the ranking and alignment loss functions.

We were able to successfully reproduce the reported results, validating the efficacy and implementation of the CXR-ML-GZSL framework. Furthermore, we investigated a new ablation to study the effects of replacing the BioBERT embedding model with BioGPT. The results of this study are discussed alongside the core reproduction findings, providing additional insight into the model's strengths and limitations.

**Original Code** —
https://github.com/nyuad-cai/CXR-ML-GZSL

**Datasets** — https://nihcc.app.box.com/v/ChestXray-NIHCC

**Reproduction** —
https://github.com/kvchong2/CXR-ML-GZSL-reproduction

**PyHealth Pull Request** —
https://github.com/sunlabuiuc/PyHealth/pull/677

**Video** — https://mediaspace.illinois.edu/media/t/1_06is2aci

## Introduction

The development of high-performing DL models is highly dependent on large amounts of accurately annotated data, a process that is exceptionally tedious, time-consuming, and expensive. Furthermore, it is difficult to collect enough training examples for rare diseases or during novel outbreaks (such as COVID-19). A typical deep learning network cannot classify a disease not represented in its training set. However, the authors leverage an insight where while visual examples of a disease may be scarce, its characteristics and attributes are documented in medical literature. This knowledge can be used to describe unseen diseases, providing a semantic bridge to enable classification. The paper's goal was to operate under a Generalized Zero-Shot Learning (GZSL) paradigm, where the system is trained exclusively on a set of seen disease classes but must accurately predict instances from both seen and unseen classes during inference.

The core contribution of the framework is in its integration of visual and semantic domains. It learns a shared latent feature space by mapping image features (from the chest X-rays) and textual semantic features (derived from disease descriptions using BioBERT) to bridge the knowledge gap for unseen diseases. This approach is powered by a specialized compound loss function to help ranking of relevant labels, proper feature alignment across modalities, and preservation of inherent semantic relationships.

(Hayat, Nasir and Lashen, Hazem and Shamout, Farah E. 2021) showed that the model achieves improved classification performance accuracy across both the seen (training) and unseen (zero-shot) disease classes, outperforming existing baseline models in various metrics. Our reproduction effort validates the implementation and efficacy of this framework.

### Scope of Reproducibility

Our reproduction effort focused on validating the core CXR-ML-GZSL framework using the full dataset and leveraging the authors' publicly released code repository. We successfully reproduced all dataset preprocessing steps as outlined in the original paper's GitHub repository. We forked the authors' original code to combine the various Python files into a single Google Colab Jupyter notebook that we could run in a GPU. In doing so, we updated to newer Python versions and dependencies, which required small edits to the code.

A limitation of our effort was constrained GPU resource availability in Google Colab. To manage this, we implemented early-stopping during the model training phase where we stop training after the AUROC metric did not improve over 5 epochs. This may be the source of the slight discrepancies observed between our final quantitative results and those reported in the original paper. Furthermore, due to these resource and time limitations, we did not reproduce the baseline models (the LESA and MLZSL models) that the original study used for comparison. Reproducing these comparison models requires reproducing these sepa-

Table 1: Seen and Unseen Disease Labels

| Disease Label | Seen | Unseen |
|---|---|---|
| Atelectasis | X | |
| Cardiomegaly | X | |
| Effusion | X | |
| Infiltration | X | |
| Mass | X | |
| Nodule | X | |
| Pneumonia | | X |
| Pneumothorax | X | |
| Consolidation | X | |
| Edema | | X |
| Emphysema | | X |
| Fibrosis | | X |
| Pleural_Thickening | X | |
| Hernia | X | |

rate papers with their own loss functions and hyperparameters that were not specified in the original paper. Our focus remained strictly on replicating the performance of the proposed CXR-ML-GZSL model itself, along with the ablation study performed in the original paper.

# Methodology

## Environment

The model implementation, training, and evaluation were done within a Google Colab Pro environment, with an A100 GPU for the runtime. The operating environment consisted of

- Ubuntu 22.04.4 LTS
- Python 3.12.12

Additional Python packages installed were

- torch - 2.9.0+cu126
- torchvision - 0.24.0+cu126
- numpy - 2.0.2
- pandas - 2.2.2
- Scikit-learn - 1.6.1
- matplotlib - 3.10.0 for charting
- tqdm - 4.67.1 for progress
- pillow - 11.3.0 for image processing

## Data

We obtained the dataset for the reproduction via the link provided by the Github repository: https://nihcc.app.box.com/v/ChestXray-NIHCC

The data repository included a script to download sets of the entire dataset, which we incorporated into a Jupyter notebook for use in the Google Colab environment. The dataset consisted of 43GB of compressed X-ray images as well as 14 classes of disease labels for each image. The authors selected diseases as being “seen” while training and “unseen” to be predicted after training (table 1).

Table 2: Summary of Dataset Splits

| Split | Images | Max Labels | Classes |
|---|---|---|---|
| training | 30,758 | 7 | 10 |
| validation | 4,474 | 6 | 10 |
| test | 10,510 | 7 | 14 |

There are 112,120 NIH chest X-ray image files in the dataset across 30,805 unique patients (table 2. The paper included the dataset splits that were used in their training. It was noted that images that included any unseen class labels were excluded from the training set.

## Model

Original Code - https://github.com/nyuad-cai/CXR-ML-GZSL

The paper describes a model that consists of a multi-modal encoder to process the visual and semantic data. The resulting feature vectors from the visual and text models are then projected into a shared latent space through a learned multi-layer perceptron. The model is then optimized with a three-component loss function to rank positively correlated disease labels, align the visual and semantic features, and to ensure consistency between the visual representation and the underlying semantic meaning of the disease.

**Multi-Modal Encoders** The visual encoder extracts the visual features from the input CSR image. The paper uses a pre-trained DenseNet-121 CNN from the torchvision PIP package as the feature extractor. The classification layers are removed by replacing them with an identity function to pass through the features. This model is “fine-tuned” on the CXR images through the training cycle. This encoder outputs the visual feature vector that is projected into the latent feature space (described below).

The semantic encoder processes the text descriptions of the disease labels associated with each CXR. The BioBERT model (a BERT model pre-trained on biomedical text) was used to generate the embeddings for the disease labels. This model is fixed in that the model weights are not modified during training. It is used to only to obtain the embeddings of the disease labels. In the original paper, these embeddings are pre-computed outside of the model and then passed into the model during training time for lookup. These embeddings are mapped into a semantic feature vector that is projected into the latent feature space.

**Latent Space Projection** The feature vectors of the visual and text models are projected to a joint latent space through two separate fully-connected multi-layer perceptrons (one for each model). Both of these projection models use the same architecture. The MLP is composed of three linear layers with ReLU activation between each layer, starting with a 512-dimension input vector, to a 256-dimension hidden layer, and finally to a 128-dimension output vector that represents the embedding in the latent space.

**Loss Functions** The novelty and success of the paper’s model relies on a comprehensive loss function designed to handle the dual challenges of Multi-Label classification and GZSL knowledge transfer. The overall loss ($\mathcal{L}$) is a combination of three components:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{rank} + \lambda_2 \mathcal{L}_{align} + \lambda_3 \mathcal{L}_{con}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters that control the weight of each loss term.

**Ranking Loss ($\mathcal{L}_{rank}$)** This loss ensures that the similarity score between the projected visual features ($z_v$) and the true (positive) semantic disease label features ($z_s^+ \in Y_P$) are significantly higher than the similarity score for any false (negative) label features ($z_s^- \in Y_N$). This weighting is defined by margin $\gamma_1$. This is the core component that enforces multi-label accuracy.

$$\mathcal{L}_{rank} = \sum_{c \in Y_P} \sum_{c' \in Y_N} \max\left(0, \gamma_1 - \left(\langle z_v, z_s^c \rangle - \langle z_v, z_s^{c'} \rangle\right)\right)$$

- $\langle \cdot, \cdot \rangle$ defines the cosine similarity function between the two projected vectors.
- $z_s^c$ is the projected semantic vector for a positive class $c$.
- $z_s^{c'}$ is the projected semantic vector for a negative class $c'$.
- $\gamma_1$ is the positive margin hyperparameter.

**Alignment Loss ($\mathcal{L}_{align}$)** The alignment loss forces the projected visual feature vector ($z_v$) to be close to the corresponding semantic text feature vector ($z_s^+$) for the positive labels in the latent space. This loss serves to "align" the visual and text features. This ensures that the model associates the visual appearance of the disease with its text description.

$$\mathcal{L}_{align} = \frac{1}{N}\left(1 - \sum_{\forall x \in X_s} \langle \psi(\rho(x)), \phi(w_x) \rangle\right)$$

- $\langle \cdot, \cdot \rangle$ defines the cosine similarity function between the two projected vectors.
- $N$ is the number of samples in the training set $X_s$.
- $\rho(x)$ is the visual feature extracted by the visual encoder (DenseNet-121).
- $\psi(\cdot)$ and $\phi(\cdot)$ are the visual and semantic projection functions, respectively.
- $w_x$ is the aggregated semantic embedding corresponding to the input image $x$.
- In case multiple labels are present in the image, their individual semantic embeddings are averaged to form $w_x$. This averaging allows the visual representation to align with the combined semantics in the multi-label setting.

**Consistency Loss ($\mathcal{L}_{con}$)** The semantic features are learned from medical text literature using BioBERT while the visual features are learned from chest X-ray images using DenseNet-121. To bridge this gap, both are mapped to a common latent domain with MLP projection functions, where the semantic representations ($\mathbf{w}$) remain fixed while the visual representations are fine-tuned.

The process of projecting the fixed semantic features ($\mathbf{w}$) to the latent space via $\phi(\cdot)$ may lead to a loss of the inherent inter-class relevance. To prevent this, the authors implemented this loss function to ensure that the conceptual similarity remains consistent throughout the projection process by leveraging the semantic inter-class relationship.

This loss considers the difference between the cosine similarity of two classes in the original space and their similarity in the projected latent space:

$$\mathcal{L}_{con} = \sum_{w_i \in \mathbf{W}} \sum_{w_j \in \mathbf{W}, j \neq i} |\langle w_i, w_j \rangle - \langle \phi(w_i), \phi(w_j) \rangle|$$

- $\mathbf{W}$ is the set of all original semantic representations (BioBERT vectors) for all classes.
- $w_i$ and $w_j$ are the original semantic representations for any two distinct classes.
- $\phi(\cdot)$ computes the projected representations in the latent space.
- $\langle \cdot, \cdot \rangle$ denotes the cosine similarity function.

Ideally, the cosine similarity between any two classes in the original semantic space ($\langle w_i, w_j \rangle$) should equal their cosine similarity in the projected latent space ($\langle \phi(w_i), \phi(w_j) \rangle$). So, minimizing this $L_1$ loss forces the projection function $\phi$ to preserve the structural relationships between the semantic disease concepts, which should help in the knowledge transfer to unseen classes.

**Inference** During inference, a test image $x_{test}$ is classified by projecting its visual features $v_{test}$ to $z_v$. The predicted diseases are determined by calculating the cosine similarity between $z_v$ and the projected semantic vectors ($z_s$) of all possible classes (both seen and unseen). The model outputs a score for every class, allowing for multi-label prediction based on a defined threshold.

## Training

**Hyperparameters** The original paper uses the Adam optimizer in the model that was trained for a total of 100 epochs, reducing the learning rate by a factor of 0.01 once the validation loss stagnates for 10 epochs. In the reference code, the learning rate starts at 0.0001 and then decays at intervals of 20 epochs by a learning rate of 1e-5. In addition, we specified the epsilon for Adam to be 1e-8.

The paper mentions that they ran multiple experiments to select $\lambda_2$ and $\lambda_3$ where selection was made in 0.1, 0.01, 0.05 to choose the best performing model based on harmonic mean AUROC. Their code ultimately selected 0.01 for $\lambda_2$ and $\lambda_3$. The ranking margin $\gamma_1$ was 0.5 but their code ultimately selected 0.2 as the penalty weight (table 3). $\lambda_1$ was not specified in the paper, but was included for our ablation reproduction as a hyperparameter to include or exclude the ranking loss during ablation studies.

A batch size was not specified in the paper. The authors expected the results to improve as they tune this as one of the parameters. However, the reference code used a batch size of 16.

Table 3: Loss Function Hyperparameters

| Loss Term | Hyperparameter | Value |
|---|---|---|
| Rank Loss | $\lambda_1$ (for $\mathcal{L}_{rank}$) | 1.0 |
| Alignment Loss | $\lambda_2$ (for $\mathcal{L}_{align}$) | 0.01 |
| Consistency Loss | $\lambda_3$ (for $\mathcal{L}_{con}$) | 0.01 |
| Ranking Margin | $\gamma_1$ (for $\mathcal{L}_{rank}$) | 0.2 |

Table 4: Dataset Splits

| Split | Percentage | Images | Seen | Unseen |
|---|---|---|---|---|
| Training | 70% | 36,231 | 36,231 | 0 |
| Validation | 10% | 5,175 | 5,176 | 0 |
| Test | 20% | 10,353 | 2,860 | 7,492 |

The visual and semantic feature mapping modules are each 3-layer MLPs starting with a 512 → 256 → 128 dimensions with ReLU activation and no dropout.

The original paper does not specify the transformations that were performed on the chest X-Ray images. The reference code performs a normalization on the channels, a randomized crop and resize of 224x224 pixels (corresponding to ImageNet image size), a randomized horizontal image flip on the training set. The normalization specifications with means [0.485, 0.456, 0.406] and standard deviations [0.229, 0.224, 0.225] for three color channels.

**Computational Requirements for Dataset Splits** To generate the train/test/validation data splits, we followed the paper's splits of 70% training, 10% validation, and 20% test. We generated CSV files containing the splits on a local laptop with the following specifications:

- Operating system: macOS 26.1
- CPU: M1 8-core CPU 16 GB RAM

The format of the CSV files include a path to the image in column 1, followed by one-hot encodings of "0" or "1" for the subsequent columns representing each of the 14 disease labels: Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural_Thickening, and Hernia (table 1). The labels for the images were read from the included Data_Entry_2017_v2020.csv file, of which the first two columns were used: "Image Index" and "Finding Labels". "Finding Labels" is a "—" separated list of disease label names.

The original paper does not mention images where there were no disease labels. Our pre-processing filtered 51,759 images with at least one disease label. Of these, 44,267 images were selected as candidates for training (containing images without "unseen" labels). From this set, our data splits are shown in table 4.

**Requirements for Pre-generated Embeddings** The original paper specifies that they used BioBERT to create embeddings of the disease labels. There were several variations of this model, including "cased" versions. Our reproduction selected the standard model at https://huggingface.co/dmis-lab/biobert-v1.1. The embeddings were created by using the HuggingFace transformers module for each disease label. The output is a serialized numpy embedding array of the disease classes.

- Operating system: macOS 26.1
- CPU: M1 8-core CPU 16 GB RAM
- Embedding Model: dmis-lab/biobert-v1.1

**Requirements for Training** The authors report that it took about 8 hours to train a single model on an NVIDIA Quadro RTX 6000 GPU. Our reproduction initially used Google Colab on a T4 GPU. However, we encountered RAM issues on the GPU. We migrated to an A100 GPU and trained against the entire CXR dataset, which took about 4.5 hours per model training. Approximately 7.52 compute units per hour.

- Operating system: Ubuntu 22.04.4 LTS
- CPU: 167GB RAM (high RAM selected) (24GB used)
- GPU NVIDIA A100 80GB RAM (27GB used)
- Disk: 236GB (110GB Used)

**Requirements for Extension Study**

- Service: AWS Sagemaker
- Instance: ml.g6.4xlarge
- GPU: NVIDIA L4 24GB RAM

**Training Details** The loss function is the main innovation of the original paper. As described in the model, it is a combination of three loss functions for ranking, alignment, and consistency.

$$\mathcal{L} = \lambda_1\mathcal{L}_{rank} + \lambda_2\mathcal{L}_{align} + \lambda_3\mathcal{L}_{con}$$

where $\lambda_1, \lambda_2, \lambda_3$ are hyperparameters that control the weight of each loss term.

The ranking loss $\mathcal{L}_{rank}$ specifies 0.2 for the penalty hyperparameter $\gamma_1$:

$$\mathcal{L}_{rank} = \sum_{c\in Y_P}\sum_{c'\in Y_N} \max\left(0, \gamma_1 - \left(\langle z_v, z_s^c\rangle - \langle z_v, z_s^{c'}\rangle\right)\right)$$

The alignment loss $\mathcal{L}_{align}$:

$$\mathcal{L}_{align} = \frac{1}{N}\left(1 - \sum_{\forall x\in X_s} \langle \psi(\rho(x)), \phi(w_x)\rangle\right)$$

And the consistency loss $\mathcal{L}_{con}$

$$\mathcal{L}_{con} = \sum_{w_i\in\mathbf{W}}\sum_{w_j\in\mathbf{W}, j\neq i} |\langle w_i, w_j\rangle - \langle \phi(w_i), \phi(w_j)\rangle|$$

Table 5: Timings

| Study | Runtime / Epoch | Epochs | GPU hrs |
|---|---|---|---|
| All loss functions | 4.6 min | 34 | 2h 36m |
| Rank+consistency | 4.7 min | 27 | 2h 7m |
| Rank+alignment | 4.6 min | 31 | 2h 22m |
| Rank only | 4.7 min | 26 | 2h 2m |

Table 6: AUROC Comparison Against Original Paper

| Method | Seen | Unseen | Harmonic Mean |
|---|---|---|---|
| Original Paper | 0.79 | 0.66 | 0.72 |
| Reproduction | 0.79 | 0.65 | 0.72 |

**Evaluation** The primary metric used in the original paper is the average area under the receiving operating characteristic curve (AUROC) for seen and unseen classes. This is computed using the roc_auc_score function in the sklearn.metrics Python module between the ground truth labels and the predicted scores. A harmonic mean is also computed against the AUROC of the two class types where:

$$H = \frac{2}{\left(\frac{1}{A} + \frac{1}{B}\right)}$$

The original paper also reported precision, recall, and f1 scores against the MLZSL (Chung-Wei Lee and Wei Fang and Chih-Kuan Yeh and Yu-Chiang Frank Wang 2018) and LESA (Dat Huynh and Ehsan Elhamifar 2020) baseline models. However, the paper focused the results and comparisons on the AUROC scores.

## Results

The run times for the model (with the three loss functions) as well as for the reproduced ablation studies are in table 5.

### Reproduced Results

The reproduced results for the seen and unseen classes against the NIH chest X-ray dataset are in tables 6 and 7.

### Reproduced Ablation Study

The reproduced results for the ablation study reproduction are in tables 8 and 9.

### Comparison with Original Paper

Despite using the same dataset splits, random seed values, and training on the full dataset, the AUROC values we obtained, although consistent with the original paper's results, differed slightly. Our results reveal the same values in the test dataset for "seen" diseases. For "unseen" diseases, our value was slightly lower than the reported value in the paper. A comparison of the individual disease classes also shows slight discrepancies, although the values are generally in line with the original paper's results.

In our reproduction of the original paper's ablation study, where the loss functions are progressively added, our results

Table 7: Class-wise AUROC Comparison Against Original Paper

| Class | Original Paper | Reproduction |
|---|---|---|
| **Seen Mean** | 0.79 | 0.79 |
| **Unseen Mean** | 0.66 | 0.65 |
| **Atelectasis** | 0.76 | 0.77 |
| **Cardiomegaly** | 0.90 | 0.90 |
| **Effusion** | 0.83 | 0.83 |
| **Infiltration** | 0.70 | 0.70 |
| **Mass** | 0.80 | 0.80 |
| **Nodule** | 0.75 | 0.77 |
| **Pneumothorax** | 0.83 | 0.83 |
| **Consolidation** | 0.69 | 0.69 |
| **thickening** | 0.72 | 0.70 |
| **Hernia** | 0.90 | 0.91 |
| **Pneumonia** | 0.62 | 0.63 |
| **Edema** | 0.67 | 0.64 |
| **Emphysema** | 0.74 | 0.76 |
| **Fibrosis** | 0.60 | 0.58 |

Table 8: AUROC Results of Original Ablation Study

| Method | Seen | Unseen | Harmonic Mean |
|---|---|---|---|
| Ranking | 0.790 | 0.574 | 0.665 |
| Rank+Consistency | N/A | N/A | N/A |
| Rank+Alignment | **0.791** | 0.601 | 0.683 |
| All | 0.783 | **0.663** | **0.718** |

Table 9: AUROC Reproduction Results of Ablation Study

| Method | Seen | Unseen | Harmonic Mean |
|---|---|---|---|
| Ranking | 0.787 | **0.674** | **0.726** |
| Rank+Consistency | 0.787 | 0.653 | 0.714 |
| Rank+Alignment | 0.789 | 0.664 | 0.721 |
| All | **0.790** | 0.653 | 0.715 |

show that the best performing model is one where we include only the ranking loss (omitting the alignment and consistency loss functions). This is very curious as it would result in a different conclusion from the original paper in that all three loss functions would result in the best model for predicting "unseen" disease classes.

There are several possibilities that might have influenced the results.

- Hyperparameters - There were differences in some hyperparameters in the reference code compared to the paper. For example, the ranking loss penalty parameter was specified as 0.5 in the paper but was 0.2 in the reference code that we used.
- Early stopping - We introduced early stopping to optimize our GPU resource allocation in Google Colab. The reference code trains for 40 epochs each training run. We noticed convergence at around 30 epochs against a sample of the data, so we implemented early stopping where if the AUROC metric does not improve after 5 epochs, we stop the training run. This would account for some discrepancies if there are metric improvements after 5 epochs.
- Package environments - We decided to use the latest Python packages available in Google Colab. Random number generation, optimizer implementations, default behaviors, and other various operations might have changed since the original paper was published in 2021 (4 year difference in package implementations).
- Runtime environment - The authors trained on an NVIDIA Quadro RTX 6000 GPU while we trained using an NVIDIA A100 80GB which differs in their architecture (Turing versus Ampere) and their available memory (24GB vs 80GB). Small differences between the GPUs could introduce timing differences that might affect non-deterministic random values, for example.

## Ablation/Extension

Given our implementation, we asked the LLM to assist us in brainstorming some ablations, and it thought of switching the semantic encoder from BioBERT to BioGPT. From our own research, we realized PubMedBERT would be a better semantic encoder for our task, as it was trained similarly to BioBERT but was more nuanced which would in theory be an improvement for Zero Shot Learning. To train this portion, an ml.g6.4xlarge instance was used via AWS Sagemaker. The instance uses an NVIDIA L4 GPU with 24 GB VRAM.

To test this ablation, we generated a new semantic encoding and replaced BioBert with PubMedBert. However from testing, only using the new semantic encoder for new embeddings did not have a reproducible improvement compared to the BioBERT encoding, with our AUROC mean using the entire dataset being around 0.75 compared to 0.79 for the original implementation.

Due to this, we decided to take it a step further and utilized the LLM to brainstorm and introduce a new hybrid loss function that utilized the embedding with visual semantic alignment loss as well as semantic preservation loss to try to improve the model results.

After running the model with the additional Hybrid loss function, the score on the seen data improved over the original ablation with a mean score of 0.77 (table 10), however it appeared to regress in its ability to accurately define unseen diseases, with a 10% reduction in accuracy from 0.62 to 0.51 (table 11). This unfortunate result could potentially be due to the alignment and preservation loss functions confusing the model, leading to relatively poor results.

### Mathematical Formulation of Hybrid loss function

$$\mathcal{L}_{hybrid} = \alpha \cdot \mathcal{L}_{align} + (1-\alpha) \cdot \mathcal{L}_{preserve} \tag{1}$$

Where:

- $\mathcal{L}_{align} = 1 - \cos(v_i, t_i)$ for image-disease pair $(i)$
- $\mathcal{L}_{preserve} = D_{KL}(softmax(S_{mapped}) \| softmax(S_{original}))$
- $S$ = pairwise similarity matrices between diseases

Table 10: Test Seen Results (mean)

| Disease | PubMedBERT | PMB + Hybrid Loss |
|---|---|---|
| AUROC Mean | 0.7568 | 0.7693 |
| Atelectasis | 0.7421 | 0.7337 |
| Effusion | 0.7976 | 0.8008 |
| Infiltration | 0.6743 | 0.6857 |
| Mass | 0.7688 | 0.7873 |
| Nodule | 0.7510 | 0.7514 |
| Pneumothorax | 0.8182 | 0.8064 |
| Consolidation | 0.6669 | 0.6517 |
| Cardiomegaly | 0.8906 | 0.8916 |
| Pleural_Thickening | 0.6906 | 0.7138 |
| Hernia | 0.7678 | 0.8707 |

Table 11: Test Unseen Results (mean)

| Disease | PubMedBERT | PMB + Hybrid Loss |
|---|---|---|
| AUROC Mean | 0.6245 | 0.5138 |
| Edema | 0.6566 | 0.5899 |
| Pneumonia | 0.4883 | 0.5438 |
| Emphysema | 0.6450 | 0.4291 |
| Fibrosis | 0.7079 | 0.4923 |

Table 12: Results Summary

| Method | Harmonic Mean |
|---|---|
| Paper | 0.718 |
| PubMedBERT | 0.6843 |
| PubMedBERT + Hybrid Loss | 0.6161 |

## Discussion

### Reproducibility

The model and the results were reproducible with the help of the reference code provided by the paper's authors. The paper was well written and the reference code provided most of the specific hyperparameters used in the model training. In addition, although data processing scripts were not included, specific training, validation, and test splits were provided, along with pre-computed semantic embeddings of the disease class labels.

### What was Straightforward

The reproducibility of the model and the results in the original paper was significantly facilitated by the authors' release of their source code on GitHub. This provided the necessary implementation details, including the specific architecture of the visual (DenseNet-121) and semantic (BioBERT) components, the structure of the simple 3-layer MLP projection, the precise loss function definitions, and the overall training pipeline. This existing code base greatly mitigated major technical roadblocks. Our ability to utilize this code as a reference, convert the scripts to a Google Colab Jupyter notebook, and update Python dependencies allowed us to quickly establish the core training environment and validate the proposed model's performance.

### What was Challenging

However, relying on the original published paper alone would have made exact reproduction challenging due to several underspecified details and inconsistencies between the paper and the code. For instance, the methodology lacked details regarding the creation of the semantic embeddings, only mentioning BioBERT without detailing the exact model variant (e.g. cased or uncased) or the crucial tokenization method used (e.g., CLS token extraction vs. mean pooling).

The original paper also does not specify the transformations that were performed on the chest X-ray images. The reference code performs a specific normalization on the channels, a randomized crop and resize of 224x224 pixels, and a randomized horizontal image flip on the training set.

We also observed discrepancies between the hyperparameters stated in the paper (e.g., ranking loss margin of 0.5 and Adam optimizer decay at 10 epochs) and those found in the released GitHub code (ranking loss margin of 0.2 and decay at 20 epochs).

Finally, the committed code contained an error where the multiplication factor for the alignment loss ($\lambda_2$) was set to zero, effectively disabling a key loss component. In cases of discrepancy, we proceeded by assuming the GitHub repository contained the accurate and runnable configuration used to generate the final reported results, highlighting a gap between the formal publication and the working implementation.

### Recommendations for Improving Reproducibility

To improve the reproducibility, we recommend the following:

- Removing inconsistencies between the paper's stated hyperparameters and what was actually implemented in the reference code
- Specifying exact semantic embedding model used and tokenization method. Provide reference data generation scripts to reproduce the data and the semantic embeddings.
- Specifying transformations performed on the images for training

## Author Contributions

- Tyler Cheng (tylerc10) - Extensions, Ablations, and PyHealth task/model
- Kelvin Chong (kvchong2) - Model reproduction, results evaluation, reporting
- Wen Tong (went2) - Model reproduction, reporting

## References

Chung-Wei Lee and Wei Fang and Chih-Kuan Yeh and Yu-Chiang Frank Wang. 2018. Multi-label Zero-shot Learning with Structured Knowledge Graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1576–1585.

Dat Huynh and Ehsan Elhamifar. 2020. A Shared Multi-Attention Framework for Multi-Label Zero-Shot Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hayat, Nasir and Lashen, Hazem and Shamout, Farah E. 2021. Multi-Label Generalized Zero Shot Learning for the Classification of Disease in Chest Radiographs. In *Proceedings of the 6th Machine Learning for Healthcare Conference*, 461–477.