Zero-Shot Object Detection for Chest X-Rays

Ellie Talius
Stanford University
etalius@stanford.edu

Ruhi Sayana
Stanford University
rsayana@stanford.edu

Abstract

Despite the success of the application of deep learning in medicine in recent years, deep learning models still require large amounts of expensive training data. We propose a novel zero-shot object detection method for pathologies in chest x-rays, building off previous works in object detection, by using a similar method to slow R-CNN and self-supervised learning, by using the CLIP model. We continue training a CLIP model on chest x-rays and radiology reports to learn a representation then applied to object detection. We also perform data augmentations on MIMIC-CXR to improve performance. We find that a larger batch size, data augmentations, using selective search, and different textual prompts improve performance, while a ResNet backbone does not.

1. Introduction

The medical field, and specifically radiology, has long been a promising domain for the application of deep learning methods. One of the most widely seen applications of deep learning is performing object detection for different pathologies in chest x-rays. Automated detection can make diagnostic imaging more accessible, enabling the early diagnosis of severe disease and significantly influencing patient outcomes. However, most deep learning methods require large amounts of labelled data, which is both expensive and time consuming to obtain, since it must be annotated by an expert with the domain and technical knowledge to correctly label the training examples [15]. Furthermore, obtaining labelled training data for the task of object detection is even more difficult than typical medical image data, as it requires medical experts to draw and label bounding boxes for each pathology.

Several methods such as model pre-training and self-supervised learning have attempted to decrease the reliance of deep learning models on labeled data [3,9,14]. However, most of these methods still require a fine-tuning step, which uses labelled data to improve model performance [23]. As a result, these models are only able to detect pathologies that they have seen explicit labels for during the fine-tuning phase, making them unable to generalize to new classes and objects.

We propose the application of Open AI's CLIP model to the task of object detection for pathologies in chest x-rays without doing any fine-tuning or using any explicit labels. **To our knowledge, we are the first to perform zero-shot object detection for chest x-ray images.** CLIP is trained using contrastive learning between images and unstructured text, meaning it learns to predict which images are associated with which texts [14]. In our application, we leverage the fact that chest x-rays are naturally found paired with radiology reports, and continue training the CLIP model on these image-report pairs. Then at test time, we use the trained CLIP model to generate labelled bounding bounding boxes of different patholgies in a chest x-ray image.

Specifically, the input to the model during training time is a batch of chest x-ray images and their associated text reports. The model then learns to predict which image belongs to which text report and outputs the cosine similarities between all the image embeddings and text embeddings in the batch.

For inference, the input to the model is a single chest x-ray image and a set of all the labels of pathologies that could be present in the image. The model then outputs the coordinates of boxes containing a pathology in the image and the label for each box (see Figure 7). If there are no bounding boxes found, the model predicts the chest x-ray is "Normal".

2. Related Work

2.1. Object detection

Traditionally, approaches to object detection have been based off convolutional architectures such as Fast-R-CNN [6] and Faster-R-CNN [17]. The basic approach is to use a regional proposal network to propose potential areas where objects might be and then to use an image classifier to predict class labels. The original R-CNN [7] uses a fixed algorithm (selective search) to generate region proposals, while faster methods first run the CNN through the input image

and come up with region proposals from the feature map. This leads to significant speeding up in performance and the incorporation of region proposals into the training process, leading to better classification results. Follow up work in [16] improves on efficiency by jointly predicting bounding boxes and class probabilities, treating object detection as a regression instead of classification problem. YOLO conducts real-time detection, thus producing results much faster than Faster-R-CNN, but lacks robustness since spatial constraints of the algorithm make it difficult to identify smaller objects [16].

Recent work in [1] adapts the Transformer architecture from [21] to object detection by treating predictions as a bipartite graph matching problem and achieves competitive performance with convolutional networks. Since the introduction of the vision transformer in [4], work in [5, 13] have adapted transformer architectures to exceed the performance of convolutional architectures while not compromising on performance. While Faster-R-CNN is currently the state-of-the-art method for object detection, transformer architectures have shown comparable accuracy and performance with simpler architectures, making them a promising alternative [17]. Transformer-based methods also favorably eliminate the need to manually tune parameters for anchor generation and non-maximum suppression in region proposals [5, 13].

In medical imaging spaces, detection of pathologies in chest x-rays is still being done manually by trained radiologists. However, in recent years attempts in industry and academia are being made to integrate artificial intelligence object detection into clinical practice [22].

2.2. Unsupervised learning

Object detection models have been trained on large datasets, where the traditional benchmark has been the MS-COCO dataset [12]. However, collecting large amounts of data is an expensive process, so there has been recent interest in the community in developing self-supervised methods to learning representations. Contrastive techniques starting with SimCLR [3] and follow up works in [2, 11] try to project augmented views of images to be close in the embedding space. Finetuning these representations for object detection achieves competitive performance, with [2] currently being the state of the art for object detection on COCO.

Open AI's CLIP model [14] utilizes unsupervised techniques on multi-modal data. With text-image pairs, CLIP is trained by utilizing the pre-training task by associating captions with images. Compared to training on a fixed set of labels in [12], language is highly general and has the open-set capacity to represent a large label space. As a result, utilizing natural language supervision to develop visual models has demonstrated strong zero-shot capabilities. We utilize these methods to develop zero-shot object detectors on chest x-ray images.

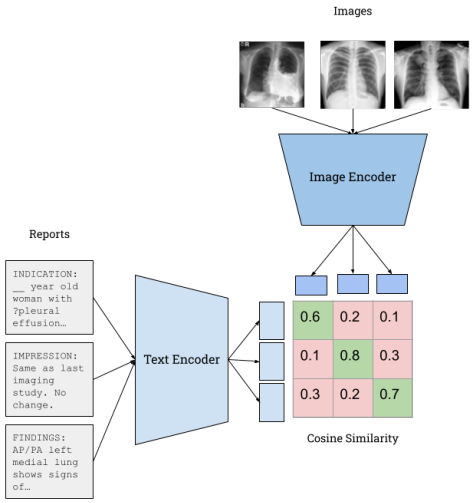

Figure 1. Visual representation of training procedure. The model learns to maximize the cosine similarity between correct image-text pairs and minimize the cosine similarity between incorrect pairs

3. Methods

3.1. Baseline Method

For our baseline result, we evaluated the detectron2 model pretrained on the ImageNet-5k dataset. The model contains a region proposal network, or deep fully convolutional network as its first module. The region proposal network takes in an image as an input and outputs rectangular object proposals, each of which have an objectness score, as the output. An $n \times n$ spatial window is slid over the feature map from the last convolutional layer and fed into fully connected networks for regression and classification. Multiple region proposals are then generated per sliding-window location, and the classification layer outputs scores corresponding to the probability of the object or not object in each proposal. The bounding boxes are then fed into CNNs and classified as different objects.

Unlike our model, this method is fully supervised and relies on explicit labels for training. This method achieves high performance on the Kaggle VinBigData Chest X-Ray Abnormalities Detection challenge, and represents the ultimate goal for zero-shot object detection in the future.

3.2. Model Training

To train a model to perform object detection, we use the pre-trained CLIP model released by OpenAI and continue

training it on the MIMIC-CXR dataset, which consists of chest x-ray image and radiology report pairs. The CLIP model consists of an image and text encoder, which are jointly trained using a constrastive loss. This loss learns to maximize the cosine similarity between correct image-text pairs and minimize the cosine similarity between incorrect pairs, as seen in Figure 1.

Specifically, given an $N \times N$ minibatch of pairs made of N images and N radiology reports, the model learns to maximize the cosine similarity of the N correct pairings and minimize the cosine similarity of the $N^2 - N$ incorrect pairings. To do so, a symmetric cross entropy loss from images to text and text to images is used. We define an image encoding as v and text encoding as u to define the loss as:

$$L = \frac{1}{n} \sum_{i=1}^{N} l_i^{v \to u} + l_i^{u \to v}$$

$$l_i^{v \to u} = -\log \frac{\exp(sim(u_i, v_i))}{\sum_{k=1}^{N} \exp(sim(u_i, v_k))}$$

$$l_i^{u \to v} = -\log \frac{\exp(sim(v_i, u_i))}{\sum_{k=1}^{N} \exp(sim(v_i, u_k))}$$

where $l_i^{v \to u}$ is the loss from image to text encoding and $l_i^{v \to u}$ is the loss from text to image encoding.

To perform this training, we build upon the CLIP codebase released by OpenAI. We use the model architecture defined in model.py as well as the load_clip function in the CLIP API for our training. Since OpenAI doesn't release code to train the CLIP model, we write scripts to perform the full training procedure, including loading the model, optimizer and data and creating a training loop to train the model. We then package this entire training pipeline into a command-line script that can be easily called with different settings (such as model backbone to use, batch size, etc).

3.3. Evaluation Set-Up

To evaluate the model, we take the learned joint representation of the model and apply it to the task of object detection. To do so, we follow the procedure in [18], which does not have an associated paper. The procedure of using CLIP for object detection is as follows: we first propose potential bounding boxes using two methods of region of interest proposal, then extract the crops of the images corresponding to the bounding boxes, and then classify each bounding box using the learned representation of the model, as seen in Figure 2.

3.3.1 Region Proposal Methods

We compare two different methods of proposing regions: superpixel segmentation and selective search. Selective search is highly time-consuming, but groups similar regions

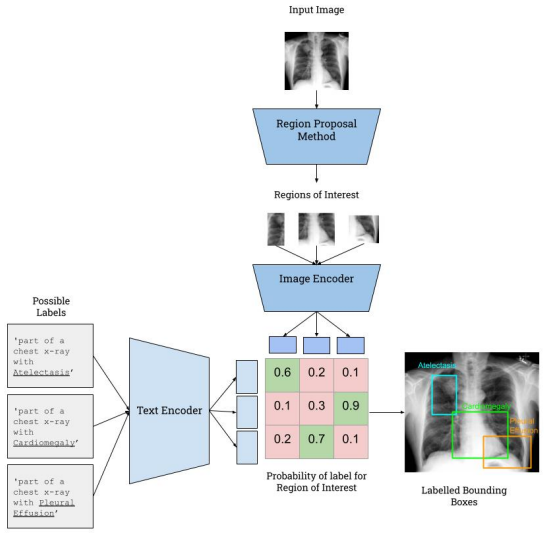

Figure 2. Visual representation of evaluation procedure. Given an image and set of labels, the model predicts bounding boxes labelled with the patholgy located in the bounding box.

together based on color, texture, size and shape compatibility as opposed to simple superpixel segmentation, which solely considers pixel value and location when determining regions to group together. Both methods propose bounding boxes directly from the test images.

In superpixel segmentation, we first blur and downsample the image using Mean Shift Filtering. We then partition each pixel into one of ten groups that they are the closest to (0, 28, 57, 85, 114, 142, 170, 199, 227, 255), creating superpixels. We then create a powerset with all possible combinations of superpixels in the image. Each combination in the power set is used to create an image mask, and possible bounding boxes are then generated from those masks.

For selective search [20], which was used to propose regions in the original R-CNN paper [7], similar regions are grouped together for many iterations. The similarity between regions is determined by measuring the similarity between color, texture, size, and fill. For two regions r_i and r_j, the following equations are used to compute similarity for each of these four features:

For color and texture, $s_{color}(r_i, r_j) = \sum_{k=1}^{n} \min c_i^k, c_j^k$ and $s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min t_i^k, t_j^k$ where c_i, c_j are one-dimensional color histograms and t_i, t_j are one-dimensional texture histograms. The size feature encourages smaller regions to merge together, such that $s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(r_j)}{\text{size}(im)}$ Lastly, the fill feature aims to merge regions that contain significant overlaps with each other. It defines BB_{ij}, the tight outer bounding box that contains both r_i and r_j, and defines $s_{fill}(r_i, r_j)$ to be the fraction of BB_{ij} not in r_i or r_j. Thus, $s_{fill}(r_i, r_j) =$

$1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_j)}{\text{size}(im)}$

The final similarity is denoted as

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

where $a_i \in \{0, 1\}$ determines if a specific component of the similarity is used or not.

3.3.2 Object Detection with CLIP

After generating bounding boxes using either superpixel segmentation or selective search, we then generate crops of the images corresponding to each bounding box and encode each crop using the trained CLIP image encoder. We then generate a text encoding for each pathology by filling in a text template such as "a chest x-ray with {}" with the name of each pathology and encoding it using the CLIP text encoder.

For each bounding box, we compute logits using the cosine similarity of the image encoding of the crop and text encodings of the filled in templates, as follows:

$$\begin{bmatrix} l_1 \\ \vdots \\ l_n \end{bmatrix} = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix} \begin{bmatrix} t_1 \\ \vdots \\ t_n \end{bmatrix},$$

where each row $c_{11} \cdots c_{1n}, c_{21} \cdots c_{2n}$, etc represent the image encodings for each bounding box image crop, $t_1 \cdots t_n$ are the values of the text encoding, and $l_1 \cdots l_n$ are the encoded logits.

Via this metric, images and text that project to similar places in the feature space will have large scalar values, while those that do not will have low scalar values. Thus, the cosine similarity tells us how likely the object in the bounding box matches the text query. Next, we compute the likelihood that our label is a false positive. First, we embed the entire image using the model image encoder. We then compute the dot product between this image embedding and the text embedding from the query, obtaining a scalar threshold. If this threshold is very low, it's likely that the text query did not appear in the image at all, thus indicating that the logits show a false positive. Thus, no boxes will be predicted for this image.

We then normalize the logits to get probabilities using a softmax function and find all bounding boxes with a probability that exceeds the skip box threshold (skip_box_thr=0.5). From these boxes, we then use non-maximum suppression to further remove boxes that have high overlap with other object-containing boxes with an IoU threshold of 0.01. These bounding boxes are then labelled as the class label and returned.

To establish our baseline with CLIP not trained on MIMIC-CXR, we find the maximum probability score across all boxes and return only that bounding box. This avoids the need to tune hyperparameters such as the threshold probability to skip a box, but is not ideal since each test image may have more than one pathology. For the evaluation, we call the code in [18], in functions we write to handle the evaluation for all the images in the test set. Our prediction code handles all the formatting and resizing of the test data, as well as how to handle the case in which the model predicts there are no bounding boxes for an image and thus it is labelled as "Normal". As in training, we also package all these functions into a command line script to easily run the evaluation procedure.

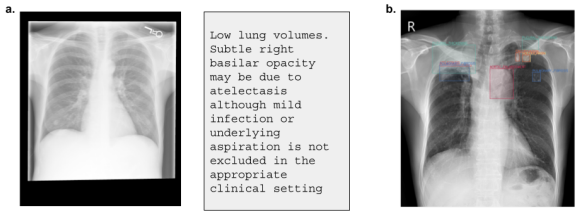

Figure 3. a) Example of a chest x-ray image and radiology report impression section pair from the MIMIC-CXR dataset. b) Example chest x-ray image with labelled bounding boxes plotted in it from the VinBigData Chest X-ray Abnormalities Detection Challenge Dataset.

4. Dataset

4.1. MIMIC-CXR

We train the model using the MIMIC-CXR training set [10], which consists of 377,110 images corresponding to 227,835 radiographic studies. Each of the studies also contains a text radiology report containing the Examination, Indication, Impression, Findings, Technique, Comparison done by the radiologist. An example chest x-ray image radiology report pair is given in Figure 3.

We resized the data to 224×244 (as is standard in medical applications and to speed up training). We then further preprocessed the data by normalizing using the sample mean and standard deviation of the dataset. For training, we place the data in an h5 file to efficient access. We preprocess the text data by extracting the impressions section and truncating the text to 77 tokens, as this is the context length of the pre-trained CLIP model. The text data is then tokenized via byte-level Byte-Pair-Encoding, which uses bytes as the base vocabulary and counts the frequencies of each byte within the report.

4.2. VinBigData Chest X-ray Abnormalities Detection

We evaluate using the VinBigData Chest X-ray Abnormalities Detection dataset which is hosted as a Kaggle competition. The data set contains 15,000 training images and 3,000 test images, of which we only use the test set. The each image in the test set either has no diseases, one disease or multiple diseases that can be identified with a bounding box placed around them. A visualization of an example image with its labelled bounding boxes plotted on it is presented in Figure 3.

The labels for the test set are not public, and as such performance on the test set can only be evaluated through a kaggle submission. We download and preprocess the test set by converting the dicom files to png's, scaling pixel values to be between 0 to 255 (greyscale), flattening them to one channel, and then normalizing them using the mean and standard deviation of the data. We also resize test images to 224×224 prior to evaluation.

4.3. Data Augmentation Methods

To determine the impact of data augmentation methods on the performance of the model on the task of object detection for pathologies in chest x-ray images, we test three different data augmentation formulations.

Firstly, we experiment with a horizontal flip of the image with probability $p = 0.5$ as done in [15]. We then also experiment with a random crop of the image to downsize the image to 224×224 as used previously without any augmentations as done in [14].

Lastly, following the transforms used in [19], we apply the following transformations to the training data: random perspective transformation with $p = 0.5$ and distortion scale $= 0.1$, a random affine transformation with max rotation $= 10$ degrees, max translation $= 0.0625$, and scaling factor $= (0.1, 0.3)$, a random crop to the input resolution, a random horizontal flip with $p = 0.5$ and random color jitter with brightness $= (0.8, 1.2)$ and contrast $= (0.8, 1.2)$. We refer to this set of transformations as the "full transformation pipeline".

5. Experiments, Results and Discussion

5.1. Evaluation Metric

The metric used for evaluation by the Kaggle competition is the standard PASCAL VOC 2010 mean Average Precision (mAP) at IoU > 0.4. Intersection over Union (IoU) measures the area of overlap between the actual and predicted bounding box where IoU $= \frac{\text{Area of Intersection of Boxes}}{\text{Area of Union of Boxes}}$ and mAP is defined as the mean of the average precision scores for each class, given by the following equation:

$$\text{mAP} = \frac{\sum_{c=1}^{C} \text{AP}(c)}{N}$$

where C is the number of classes and $\text{AP}(c)$ is the average precision of class c.

To calculate the average precision for each class, each bounding box predicted by the model is classified as true positive (IoU > 0.4 and correct class is predicted), false positive (IoU < 0.4 or the box is a duplicate of a true positive box), or false negative (no box is predicted or IoU > 0.4 and incorrect class is predicted).

A precision-recall curve for each class is then computed using the confidence scores for each box output by the model labelled as the given class and the true positive, true negatives and false negatives for the given class. The AP for the class c is then defined as $\text{AP}(c) = \int_0^1 p_c(r)dr$ where p_c is the precision-recall curve for the class c.

We report results using the private mAP scores calculated by the Kaggle competition for each submission, which are calculated using 90% of the test data set.

5.2. Training and Hyperparameters

When training the model, we followed [14] using an Adam optimizer with the same hyperparameters except for the mini-batch size and number of epochs. We started with a learning rate of 5×10^{-4} as done in [14], and we inspected the loss curves (see Figure 4) to determine that this learning rate was appropriate. We chose to use an Adam optimizer because as reported in [14], it gave the best performance.

For the mini-batch size, run an experiment using batch sizes of $16, 32, 64$ and 128 and see that as expected, the batch size of 128 performs the best (see 5.3.2). [14] uses a mini-batch size of 32768, but with our computing resources, the largest batch size we are able to use without running out of memory is 128.

We train for 2 epochs as opposed to 32 in [14] because our dataset is much smaller than the training dataset used for training the pre-trained CLIP model. Furthermore, the loss plateaus after the second epoch (step 650,000) in Figure 4, indicating that further training would cause overfitting.

The pre-trained CLIP model we use is trained on 400 million image-text pairs, and as such, the authors of [14] are not concerned about the model overfitting the training set since the dataset is so large. Our training dataset is much smaller, so we explore data augmentations to mitigate overfitting as discussed in 4.3, 5.3.4. We see that the model trained using the full transformation pipeline as described 4.3 performs better on the object detection task, indicating that the model trained without any data augmentations may have been overfitting the training set.

We also determined hyperparameters for region proposal methods by qualitatively sampling images. For superpixel

Figure 4. Loss curves of the models with a ViT-B/32 backbone for the image encoder and varying batch sizes. We see that the learning rate is appropriate and the loss plateaus after 2 epochs (step 650,000), indicating that the model should no longer be trained.

segmentation, we set the maximum box dimensions to 0.9 of the image dimensions and the minimum box dimensions to 0.1 of the image dimensions to ensure that both small and large lesions could be detected. We set the threshold for skipping boxes to be over 0.5 so that only bounding boxes thought to have an object in them by the CLIP text and image encoders would be considered, and the false positive threshold to be -2 to prevent nonsensical images from returning any bounding boxes.

For selective search, through qualitative sampling, we determined that the best parameters were $\sigma = 1$ (controls Gaussian blur), $k = 500$ (size at which to merge boxes), and distortion = 1.25 (threshold to preserve aspect ratio). We set the smallest and largest box dimensions to be 10 and 224, to ensure that pathologies of all sizes could be detected.

5.3. Experiments and Results

5.3.1 Comparison to Baseline

For all experiments, we compare the result of the experiment against our baseline model of CLIP model with a ViT-B/32 transformer backbone for the image encoder, batch size of 128, superpixel segmentation for region proposal method, prompt "a chest x-ray with {}" and no data augmentations. In each experiment, only the component explicitly being tested is modified.

Our best model is this baseline with the full image transformation pipeline used, and achieves a mAP score of 0.045 on the test set. This model outperforms the pre-trained CLIP model without any further fine-tuning that predicts only one bounding box per image (mAP = 0.021) and the pre-trained CLIP model without further training that predicts multiple boxes (mAP = 0.017).

Our best model underperforms when compared to the dectectron2 fully supervised baseline, which achieves a mAP score of 0.235. However, this is to be expected, as our model uses no explicit labels to perform object detection, which limits the complexity of the region proposal method we can use when compared to the integrated and trained region proposal network used in detectron2.

Supervision	Model	mAP Score
Fully Supervised	Detectron 2	**0.235**
Zero-Shot	CLIP-Single-No Train	0.022
	CLIP-Multiple-No Train	0.017
	Best CLIP	**0.045**

Table 1. Comparison of mAP scores on the test set between the fully supervised baseline and different CLIP models. CLIP-Single-No Train and CLIP-Multiple-No Train refer to the pre-trained CLIP model without further training on MIMIC-CXR predicting a single and multiple bounding boxes respectively. Best CLIP refers to the best performing trained CLIP model.

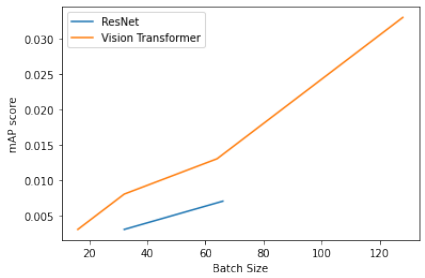

Figure 5. Effect of batch size on mAP scores of CLIP models trained on the ResNet backbone versus the vision transformer backbone. As expected, performance improves with increases in batch size.

5.3.2 Impact of Batch Size

Prior work in using self-supervised models with a contrastive loss has highlighted that larger mini-batch sizes lead to better results because a larger mini-batch gives the model more negative examples to learn from [3]. As such, we perform an experiment to determine the extent to which increasing the batch size used to train the model improves performance.

We continue training the standard CLIP model setup described in 5.3.1 using batch sizes of 16, 32, 64 and 128. We are not able to use a batch size larger than 128 due to memory constraints.

As shown in Figure 5, we see that increasing the batch size substantially improves the performance of the model. Specifically, the mAP on the test set increases from 0.003 to 0.008 to 0.0013 to 0.033 as the batch size the model was trained with increases from 16 to 32 to 64 to 128.

This demonstrates a linear relationship between the batch size and model performance, meaning that increasing the batch size used to train the model or switching to a queue based method such as [9] that allows for more

	Batch Size			
	16	32	64	128
Vision Transformer	0.003	0.008	0.013	0.033
ResNet		0.003	0.007	

Table 2. Comparison of mAP scores on the VinBigData Chest X-ray Abnormalities Detection test set for different CLIP models using the ResNet backbone and vision transformer backbone at different batch sizes.

Data Augmentation	**mAP Score**
Random Crop	0.035
Random Flip	0.025
All Transforms	0.045

Table 3. Comparison of mAP scores on the VinBigData Chest X-ray Abnormalities Detection test set after training with different data augmentations. All CLIP models compared here used the vision transformer backbone with a batch size of 128.

negative examples to be seen for each example without increasing the batch size can improve the performance of the model.

5.3.3 Choice of Model Backbone

OpenAI released pretrained CLIP models with both transformer and ResNet backbones for the image encoder [14]. As ResNet models have been applied to chest x-ray classification [24] and used for object detection [8] with success, we experiment with the choice of model backbone.

In the results of [14], they find that the vision transformer is more compute efficient and thus gives better performance. We also we find the ResNet backbone is less compute efficent. As shown in Figure 8, the largest batch size we can use with a ResNet is 64, and the model takes 3x as long to train. As batch size is critical to model performance, this greatly impacts the results. When comparing results of the ResNet and ViT-B/32 models with the same batch size, we see the ResNet performs worse with a batch size of 32 (mAP = 0.003 vs mAP = 0.008) and 64 (mAP = 0.007 vs mAP = 0.013) 8.

5.3.4 Impact of Data Augmentation

To decrease the potential of the model to overfit the training data, we explore 3 different forms of data augmentation: a random horizontal flip with $p = 0.5$ as done in [15], a random crop of the image to 224×224, as done in [14] and a full transformation pipeline, as done in [19]. We describe these augmentations further in section 4.3.

We find that using the random flip transformation decreases the performance of the model (mAP = 0.025) as compared the baseline (mAP = 0.033). However, the random crop data augmentation improves performance (mAP = 0.035) and the full transformation pipeline gives the best results (mAP = 0.045), as seen in Table . The model trained using the full transformation pipeline achieves the best mAP score of any models we trained, indicating that the other models were likely overfitting to the training set. Since we evaluate on a completely different data set than we train on, the improved mAP score of the model trained using the full transformation pipeline indicates that our model is able to generalize to new tasks and data more successfully when data augmentations are used.

We hypothesize that the model trained using only random horizontal flips performs worse because unlike [15], we evaluate the model on the task of object detection instead of image classification. Thus, since our model cannot see the full chest x-ray during evaluation (as we feed the proposed crop through the model), it may be that a random horizontal flip is no longer an effective data transformation.

5.3.5 Impact of Text Prompt

The text encoding generated from the query paired with test images plays a significant role in determining which bounding boxes remain and what classification an object receives [14]. To better examine possible queries that could be used in practice with detecting pathologies in chest x-rays and understand what phrases are represented in the radiology reports, we explore the effects of using different text queries on the performance of the model. We evaluate the baseline model with five different prompts as shown in Table 4.

We achieved best performance with prompt 1 (mAP = 0.036). Prompts 3 and 4 also performed similarly (mAP=0.035, 0.031), while Prompts 2 and 5 performed the worst. This indicates that radiology reports paired with chest x-rays may not explicitly label pathologies with words that express sections, such as ”part” or ”crop”. Using queries that are more general, containing keywords like ”chest x-ray” and the name of the specific lesion lead to better performance.

5.3.6 Comparison Using Selective Search

We lastly compared the effects of the region proposal method on the performance of our model. We compared two different methods: superpixel segmentation and selective search, which are both described in detail in 3.3. We found that using selective search (mAP = 0.042) improves the performance of the model when compared to superpixel segmentation (mAP = 0.033) as shown in Table 5. Selective search optimizes for similarity among four different features and combines similar regions together, while su-

Prompts	mAP Score
1) a chest x-ray with {}	**0.036**
2) part of a chest x-ray with {}	0.025
3) {}	**0.035**
4) {} present in a chest x-ray	**0.031**
5) crop of a chest x-ray showing {}	0.027

Table 4. Comparison of mAP scores after encoding different prompts using the CLIP text encoder.

Region Proposal Method	mAP Score
Superpixel Segmentation	0.033
Selective Search	0.042

Table 5. Comparison of mAP scores after using superpixel segmentation and selective search to generate bounding boxes.

perpixel segmentation only groups pixels together based on pixel value, losing expressivity from other features. However, using selective search instead of superpixel segmentation leads to significant increases in evaluation time (6 hours with selective search, 1.5 hours with superpixel segmentation).

5.4. Qualitative Analysis

In order to better understand the model predictions, we visualize the bounding boxes and pixel masks outputted for the model for various pathologies.

As seen in Figure 6, the model typically generates masks of the edges of pathologies, rather than the full region containing the pathology if the pathology appears as a lighter than normal region. Specifically, in part a, the model highlights the edges of the lung containing lung opacity, but misses the lighter region in the center of the lung. Similarly, in part b that mask picks up the edges of the enlarged heart correctly, but not the entire heart. In part c too many of the edges in the image surrounding pleural effusion are also masked, and the resulting bounding box is too large. Lastly, in part f, the model picks up the brighest edges in the image as a nodule, missing the actual nodule.

However, in parts d and e, the model does correctly mask most of the lungs containing pneumothorax and pulmonary fibrosis respectively. These appear as darker than normal sections of the image, suggesting the model is better at masking entire dark regions as compared to lighter ones.

As such, this visualization demonstrates that the trained model using superpixel segmentation is more successful at masking the entire regions of pathologies that appear as darker regions of a chest x-ray than lighter regions. When needing to detect pathologies in lighter regions, the model commonly detects only the brightest edges rather than the full region. Therefore, performance could be improved by seeing more examples of chest x-rays with pathologies that appear as brighter than normal regions, and switching the region proposal method to one that better masks out entire objects instead of focusing on edges.

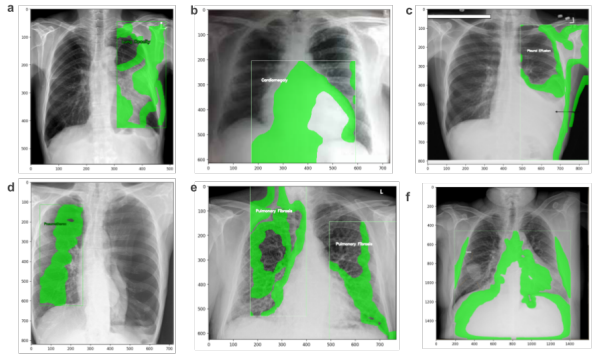

Figure 6. Visualization of bounding boxes and attention masks for 6 different pathologies. a) Lung Opacity. b) Cardiomegaly c) Pleural Effusion d) Pneuothorax e) Pulmonary Fibrosis f) Nodule

6. Conclusion and Future Work

Our work addresses the challenging problem of zero-shot object detection to detect pathologies in chest x-rays, which to our knowledge has not been attempted before. We found that using a vision transformer backbone and a batch size of 128 while training led to the best performance from CLIP. Increasing our training dataset with data augmentations such as perspective and affine transformations, distortions, rotations, and translations also improves the performance of the model. Using general queries that emphasize class names and general terms around chest x-rays also improve performance. Lastly, using selective search for region proposals instead of superpixel segmentation improves mean average precision.

Some challenges of this task include the integration of the region proposal with classification, since the model itself is not being used to propose bounding boxes. We hypothesize that one major reason our method does not perform as well as the gold-standard baseline, which employs supervised object detection, is due to the fact that bounding box proposals are not integrated with the CLIP image encoding process. In Fast and Faster-R-CNN, region proposals are selected after the image is run through the ConvNet, leading to better performance than the original R-CNN, where regions are proposed on the image prior to applying the ConvNet. Developing and improving methods to generate bounding box proposals on the image encodings produced by CLIP may be the next step for improving zero-shot object detection. Additionally, since increasing the batch size improves model performance, future work can be done to use a queue based method such as in [9] to remove this reliance on batch size.

7. Appendix

We include Figure 7 and Figure 8 as part of the Appendix for the reader.

8. Contributions, Acknowledgements

8.1. Contributions

Ellie: Wrote the training and evaluation pipeline code, trained all the models and ran evaluation for all results. Wrote sections of the paper.

Ruhi: Understood different object detection methods detect bounding boxes to fit with our evaluation pipeline. Wrote sections of paper.

We made use of the GitHub repositories released by OpenAI (https://github.com/openai/CLIP), CLIP for Object Detection (https://github.com/shonenkov/CLIP-ODS), and Selective Search for Object Recognition (https://github.com/sirius-mhlee/selective-search-for-object-recognition) as detailed in sections 3.2 and 3.3.

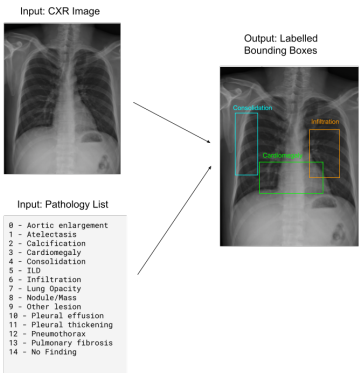

Figure 7. Visual representation of object detection task

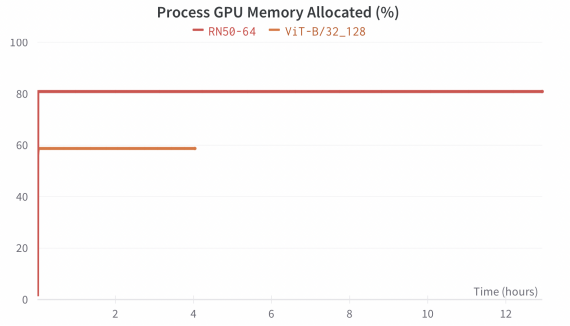

Figure 8. Comparison of % of process GPU memory allocated when training a CLIP model with a ResNet backbone with batch size 64 and vision transformer backbone with a batch size of 128. We note that the ResNet backbone model both requires more memory and takes much longer to train than the vision transformer backbone model.

8.2. Acknowledgements

We would like to thank the entire teaching team for their assistance with this project. Additionally, we used the Stanford AI Lab (SAIL) deep computing cluster for training some of our models.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 2

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021. 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 1, 2, 6

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[5] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021. 2

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1, 3

[8] Foysal Haque, Hye-Youn Lim, and Dae-Seong Kang. Object detection based on vgg with resnet network. pages 1–3, 01 2019. 7

[9] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2019. 1, 6, 8

[10] Alistair EW Johnson, Tom J Pollard, Nathaniel R Greenbaum, Matthew P Lungren, Chih-ying Deng, Yifan Peng, Zhiyong Lu, Roger G Mark, Seth J Berkowitz, and Steven Horng. Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. *arXiv preprint arXiv:1901.07042*, 2019. 4

[11] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning, 2021. 2

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. 2

[13] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2

[14] 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, 2021. 1, 2, 5, 7

[15] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning, 2017. 1, 5, 7

[16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. 2

[17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1, 2

[18] Alex] [Shonenkov. [clip object detection segmentation], 2021. 3, 4

[19] Ilyas Sirazitdinov, Maksim Kholiavchenko, Ramil Kuleev, and Bulat Ibragimov. Data augmentation for chest pathologies classification. pages 1216–1219, 04 2019. 5, 7

[20] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 3

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[22] Ruixin Yang and Yingyan Yu. Artificial convolutional neural network in object detection and semantic segmentation for medical imaging analysis. *Frontiers in Oncology*, 11:573, 2021. 2

[23] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. Contrastive learning of medical visual representations from paired images and text, 2020. 1

[24] Changjian Zhou, Jia Song, Sihan Zhou, Zhiyao Zhang, and Jinge Xing. Covid-19 detection based on image regrouping and resnet-svm using chest x-ray images. *IEEE Access*, 9:81902–81912, 2021. 7