Supplementary Material for ECCV Submission #5595: Joint Learning of Localized Representations from Medical Images and Reports

Philip Müller[1], Georgios Kaissis[1,2,3], Congyu Zou[4], and Daniel Rueckert[1,3]

[1] Institute of Artificial Intelligence in Medicine, Technical University of Munich, 81675 Munich, Germany
philip.j.mueller@tum.de
[2] Institute of Radiology, Technical University of Munich, 81675 Munich, Germany
[3] Department of Computing, Imperial College London, London SW7 2BX, UK
[4] Department for Internal Medicine I, Klinikum Rechts der Isar, Technical University of Munich, 81675 Munich, Germany

1 Analysis and Ablation Study

In this section we analyze the relevance of different components of LoVT (as proposed in the main paper) and study its learned representations. In order to save computational resources, all further analysis and the ablation study are conducted on 30% of the pre-training data.

Ablation Study We conduct an ablation study to analyze the relevance of components of LoVT and the effects of the changes we made compared to ConVIRT. Focus of our ablation study are (i) the local weighting, (ii) the global and local losses, (iii) and attention pooling. We compare different model variants and their results on the *RSNA YOLOv3 Frozen 10%* task in Tab. 1. Note that we focus our ablation study on this single task as this is the task used for tuning all models and baselines while the other tasks are only used for the final evaluation (see Sec. 5.3).

Starting from the unmodified LoVT model we first remove the local (region and sentence) weighting in the local losses, i.e. we use constant weights $w^{\mathcal{I}}_{i,k}$ and $w^{\mathcal{R}}_{i,m}$, and observe inferior results, showing the relevance of these weights. We then also remove attention pooling and replace it by average (avg) pooling for images and max pooling for reports. The performance further decreases highlighting the importance of attention pooling. Note that the local weights cannot be computed without attention pooling, making a model with local weighting but without attention pooling non-realizable. We further remove the global loss $\mathcal{L}_{\text{global}}$, i.e. set $\gamma = 0$ and only use the local losses without local weighting, and observe a large drop in downstream performance. We assume that this is caused by missing contrast between samples. Without the global loss, local weights can again not be computed, making a model with local weighting but without global loss non-realizable.

Table 1. Results of the ablation study evaluated on the *RSNA YOLOv3 Frozen 10%* task. Results are averaged over five evaluation runs and the 95%-confidence interval is shown. The best results are highlighted in bold.

Method	Global loss	Local losses	Local weighting	Pool	LR scheduler	RSNA YOLOv3 Frozen 10 %
LoVT (Ours)	✓	✓	✓	attention	cyclic-cosine	**17.9±0.4**
–	✓	✓	✗	attention	cyclic-cosine	16.9±1.3
–	✓	✓	✗[1]	avg/max	cyclic-cosine	15.7±0.4
–	✗	✓	✗[1]	–	cyclic-cosine	12.3±0.7
–	✓	non-smooth $\mathcal{L}_{\text{local-scan}}$	✓	attention	cyclic-cosine	15.6±1.4
–	✓	$\mathcal{L}_{\text{local-report}}$ only	✓	attention	cyclic-cosine	16.2±0.7
–	✓	$\mathcal{L}_{\text{local-scan}}$ only	✓	attention	cyclic-cosine	13.7±1.2
–	✓	✗	–	attention	cyclic-cosine	17.4±0.9
–	✓	✗	–	avg/max	cyclic-cosine	14.2±1.0
CLIP[2]	single sentence	✗	–	attention	cyclic-cosine	16.1±1.1
–	single sentence	✗	–	avg/max	cyclic-cosine	15.8±1.3
ConVIRT	single sentence	✗	–	avg/max	reduce-on-plateau	14.8±1.1

[1] Not realizable.
[2] Modified to use the same image and text encoders as ConVIRT and LoVT.

We also study the relevance of the local losses $\mathcal{L}_{\text{local-image}}$ and $\mathcal{L}_{\text{local-report}}$. Starting again from unmodified LoVT, we first adapt the local image loss $\mathcal{L}_{\text{local-image}}$ by redefining the positiveness score in a non-smooth way with $p^{\mathcal{I}}_{k,l} \propto \mathbb{1}_{[d_{\boldsymbol{x}}(k,l) \leq T]}$. The performance drops showing the relevance of the smoothness. When removing any of the local losses completely, i.e. either setting $\mu = 0$ or $\nu = 0$ and keeping only the global and one of the local losses ($\mathcal{L}_{\text{local-image}}$ or $\mathcal{L}_{\text{local-report}}$), the performance also drops compared to unmodified LoVT, showing that both local losses are required for optimal results. Note that removing $\mathcal{L}_{\text{local-report}}$ leads to a larger drop in downstream performance than removing only $\mathcal{L}_{\text{local-image}}$, indicating that $\mathcal{L}_{\text{local-report}}$ is more relevant for alignment. When both local losses are fully removed by setting $\mu = 0$ and $\nu = 0$, such that only the global loss remains, the performance slightly drops compared to unmodified LoVT showing that the local losses are relevant components of the model. However, the drop in performance is smaller than when removing only one of the local losses, which indicates that the symmetry of the local losses is essential for them to work. If we further replace attention pooling by avg/max pooling, a large drop in performance is observed, which again highlights the importance of attention pooling. Note that without avg/max pooling the local losses provide more improvements than when using attention pooling.

We also study the differences to ConVIRT[91] and (modified) CLIP[63]. Starting from ConVIRT, replacing their learning rate schedule (reducing on plateaus) by our cyclic cosine schedule (see Sec. 5.1) improves the results. Further replacing their avg/max pooling (to compute global representations) by attention pooling improves the results even further. This settings corresponds to (modified) CLIP. In ConVIRT (and CLIP), only a single sentence per report is sampled when computing report representations. Replacing this sampling method by ours, where all sentences of a report are used to compute its representation, the results are improved if attention pooling is used. If no attention

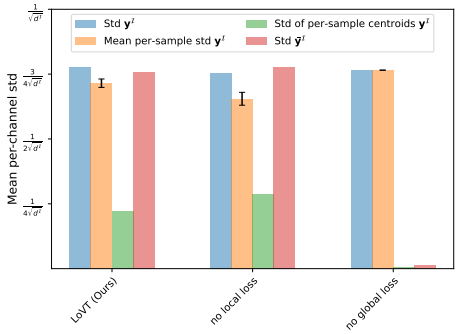

Fig. 1. Standard deviation (std) of image ($\bar{\boldsymbol{y}}_i^{\mathcal{I}}$) and image region ($\boldsymbol{y}_{i,k}^{\mathcal{I}}$) representations. **Left**: LoVT (Ours) **Middle**: No local losses. **Right**: No global loss. For $\boldsymbol{y}_{i,k}^{\mathcal{I}}$ we additionally show the mean std per-sample, i.e. how different representations are within a sample, and the std of the per-sample centroids. The models were trained on 30% of frontal MIMIC-CXR and then evaluated on the whole test set.

pooling is used, the performance degrades when using all sentences instead of a single randomly sampled one.

In our ablation study we highlighted the importance of all components of LoVT. We also showed that some of our proposed changes can also be used to improve the ConVIRT or CLIP models.

Representation Distribution and Alignment Analysis We analyze how representations are distributed and how well they are aligned. In Fig. 1 we show the standard deviation (std) of image $\bar{\boldsymbol{y}}_i^{\mathcal{I}}$ and image region $\boldsymbol{y}_{i,k}^{\mathcal{I}}$ representations of LoVT and variants of it without local losses or global loss. It can be observed that the (total) std of image region representations $\boldsymbol{y}_{i,k}^{\mathcal{I}}$ is similar in all three studied settings, indicating that the local and global losses have little influence on the overall variance of local representations. We additionally analyze the mean per-sample std and std of per-sample centroids of $\boldsymbol{y}_{i,k}^{\mathcal{I}}$ to study how representations are distributed within a sample and between different samples, respectively. The per-sample std of $\boldsymbol{y}_{i,k}^{\mathcal{I}}$ is smallest when only using the global loss (no local losses) and largest when only using the local losses (no global loss). Vice versa, the centroids std is largest when only using the global loss (no local losses) and smallest when only using the local losses (no global loss). Therefore, the local losses encourage the representations to differ within each sample, i.e. ensure spatial sensitivity, while the global loss encourages them to differ between samples, i.e. prevents the collapse of per-image representations to a constant vector. The std of global image representations $\bar{\boldsymbol{y}}_i^{\mathcal{I}}$ behaves similarly to the centroids std of $\boldsymbol{y}_{i,k}^{\mathcal{I}}$, except that the local losses have only little influence on it. Note that the centroids std and std of global image representations almost completely vanish without the global loss, while there is still notable per-sample std present without the local losses. This highlights the importance of the global loss for preventing the collapse of representations.

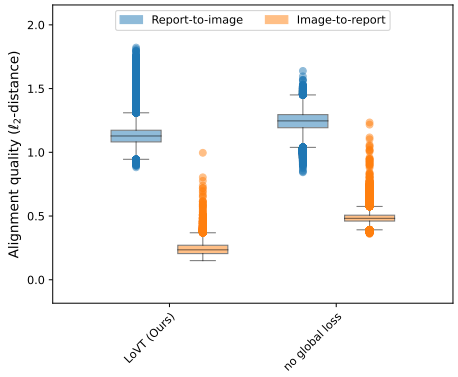

Fig. 2. Alignment quality of local representations. **Left**: LoVT (Ours) **Right**: No global loss. Measured by the ℓ_2-distance of uni-modal with their related cross-modal representations. **Blue**: Report-to-image ($\boldsymbol{z}_{i,k}^{\mathcal{R}\to\mathcal{I}}$) with image region ($\boldsymbol{z}_{i,k}^{\mathcal{I}}$) representations. **Orange**: Image-to-report ($\boldsymbol{z}_{i,m}^{\mathcal{I}\to\mathcal{R}}$) with report sentence ($\boldsymbol{z}_{i,m}^{\mathcal{R}}$) representations. All representations are ℓ_2-normalized before the distance is computed. The models were trained on 30% of frontal MIMIC-CXR and then evaluated on the whole test set.

In Fig. 2 we plot the alignment quality of local representations, i.e. the ℓ_2-distance of report-to-image ($\boldsymbol{z}_{i,k}^{\mathcal{R}\to\mathcal{I}}$) with their related image region representations ($\boldsymbol{z}_{i,k}^{\mathcal{I}}$) and of image-to-report ($\boldsymbol{z}_{i,m}^{\mathcal{I}\to\mathcal{R}}$) with report sentence representations ($\boldsymbol{z}_{i,m}^{\mathcal{R}}$). We compare the representations learned by LoVT with (default) and without global loss. It can be observed that in both cases the image-to-report representations are better aligned than the report-to-image representations. This can be expected, as most of the information contained in the report is based on the image, making it easy to compute sentence representations from image region representations, while images contain additional details not described by the reports, making it harder to compute report-to-image representations. Both, report-to-image and image-to-report representations, are slightly better aligned when the global loss is used additionally to the local losses during training (as in the unmodified LoVT model). One can therefore deduce that the global loss supports the learning of local representations.

In Fig. 3 we plot a t-SNE[51] projection of local representations learned by LoVT. Sentence representations are similarly distributed to image-to-report representations confirming, as already observed in the alignment analysis, that the model is able to align both distributions. Only one cluster of sentence representations is separated from image-to-report representations. We assume that these are sentences that do not describe features present in the image, e.g. describing features from lateral views or differences to previous studies of the patient. Image region representations and report-to-image representations are distributed differently, which again confirms that these could not be fully aligned. In the t-SNE[51] projection the image region representations are split into many clusters. We assume that this is a result of the negatives in the local image loss

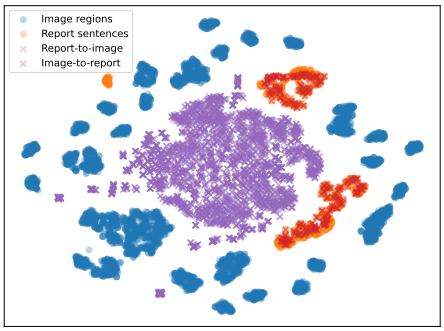

Fig. 3. t-SNE[51] plot of projected local uni-modal representations (points) and the aligned cross-modal representations (crosses). **Blue**: Image regions ($\boldsymbol{z}^{\mathcal{I}}_{i,k}$). **Orange**: Report sentences ($\boldsymbol{z}^{\mathcal{R}}_{i,m}$). **Purple**: Report-to-image ($\boldsymbol{z}^{\mathcal{R}\to\mathcal{I}}_{i,k}$). **Red**: Image-to-report ($\boldsymbol{z}^{\mathcal{I}\to\mathcal{R}}_{i,m}$). We trained our model on 30% of frontal MIMIC-CXR and computed the representations on the first 100 samples of the test set.

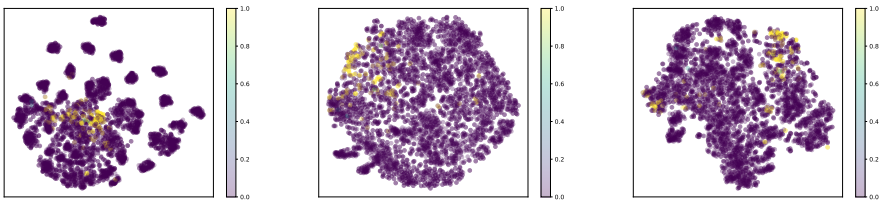

Fig. 4. t-SNE[51] plots of image region representations from samples of the RSNA pneumonia detection dataset. The color of each point indicates the overlap of the related region with a pneumonia opacity region. **Left**: LoVT (Ours). **Middle**: No local losses. **Right**: No local losses and no attention pooling. We trained all models on 30% of frontal MIMIC-CXR and computed the representations on the first 100 samples of the RSNA test set.

encouraging contrast between (spatially distant) region representations of each sample, such that our model behaves similarly to a clustering algorithm.

To further study the effect of the local losses, we plot t-SNE[51] projections of image region representations from samples of the RSNA pneunomia detection dataset in Fig. 4. We compare the representations learned using our unmodified LoVT model, our model without local losses, and our model without local losses and attention pooling. For the unmodified LoVT model, image region clusters can again be observed while such clusters cannot be observed without the local losses. This confirms our assumption that these clusters are a result of the local losses. It can also be observed that in the unmodified LoVT model the representations of pneumonia positive regions are distributed in a very confined area of space and are therefore probably easily separable from non-pneumonia regions.

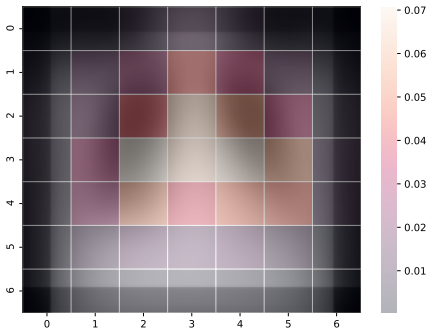

Fig. 5. Local image region weights $w^{\mathcal{I}}_{i,k}$ of different regions averaged over all samples and plotted on top of the mean image. The model was trained on 30% of frontal MIMIC-CXR. Weights and mean image were computed on the whole test set.

Without local losses the positive region representations are more spread over the space making them harder to separate. If attention pooling is not used as well, the positive region representations are distributed around multiple areas in space which may also hurt separability. Therefore, using the local losses and attention pooling improves separability of downstream representations which is confirmed by the results shown in Tab. 1.

Local Weights In order to understand how the weighting of image regions works, we study how the region weights $w^{\mathcal{I}}_{i,k}$ are distributed. In Fig. 5 we therefore plot the mean region weights on the mean image of the pre-training test set. The weights are distributed horizontally symmetrically around the center of the images and most focus is on the lungs and around the spine. This indicates that the weighting works as expected, as most pathologies in a frontal chest X-ray are typically observed at lungs or heart.

2 Discussion of the Limitations of LoVT

Weak Supervision and Sensitivity to Hyperparameters As no supervision for the alignment of image regions and report sentences is available, we implicitly learn an alignment model in the latent representation space. We jointly learn this alignment model and the latent representations of image regions and report sentences, having only the global alignment information of image-report pairs as supervision. Therefore, we suspect that the model tends to converge to local optima, explaining its sensitivity to hyperparameters, especially to the learning rate. While using the cyclic-cosine learning rate schedule helps, our method is still hard to train and tune. We leave studying more explicit supervision, e.g. by including generative losses, to future work.

Limited Negatives for Local Alignment We only use local negatives from the same sample. By design, the number of local negatives is therefore very limited and many of these negatives may be very easy. This may limit the model performance on downstream tasks[7,28]. In preliminary experiments we also included negatives from other samples but could not observe a benefit. We leave the study of losses with more negatives (e.g. based on the MoCo[28] approach) or without explicit negatives (e.g. based on the BYOL[27] approach) to future work.

Limited Alignment Model We decided to use single-head scaled dot product attention with linear projections as our alignment model. While this keeps the alignment model simple and computationally cheap, it also limits its capabilities. We leave studying more complex alignment models, like multi-head attention or (one or more) transformer[77] layers, to future work.

Non-Adaptive Regions In LoVT the image region representations are computed for fixed regions, i.e. patches. Their boundaries are arbitrary and relevant features may therefore be spread across regions or multiple neighboring features may be in the same region, making it hard to learn region representations. We leave studying other techniques for finding content-based regions and computing their representations to future work.

3 Discussion of Negative Societal Impact

In this section we discuss the possible negative societal impact of our work. We identified three primary aspects: i) usage of our method in medical applications, ii) data privacy issues, and iii) energy consumption.

Usage in Medical Applications As our method is targeted towards medical applications, potential issues in our method may lead to harm through misdiagnosis. Most of the potential issues, including interpretability and reliability issues, are not specific to our method but apply to most deep learning methods in medicine and we therefore do not discuss them here. Still, we identified another potential issue: data bias learned during pre-training. While bias from data may be learned by most machine learning methods, in our case the bias might be learned from both, pre-training and fine-tuning data. During fine-tuning the pre-training dataset might not be available making it hard to identify such a data bias. As possible mitigation strategies the pre-training dataset (if available) should also be analyzed for data bias or the fine-tuned model should be tested for learned bias before using it in medical applications. Note that this issue applies to most transfer learning approaches including other pre-training methods.

Data Privacy Issues Information learned from the pre-training dataset is contained in the pre-trained model weights, which may include information about individuals in the dataset. When distributing such models to make them available for others to fine-tune, this information is distributed as well. If the pre-training

dataset is non-public but the pre-trained model is made publicly available, this may lead to data leakage and therefore a privacy breach. This is especially problematic if individuals can be re-identified. Therefore, pre-trained models should be distributed only under the same conditions as the pre-training dataset or other precaution measures, like privacy-preserving machine learning, should be taken to prevent data leakage. We thus decided not to release our pre-trained model weights publicly. Note that this issue applies to most transfer learning approaches including other pre-training methods.

Energy Consumption and Environmental Impact Training of deep learning models consumes substantial amounts of energy and may therefore have environmental impacts. In our experiments, we observe that pre-training (including LoVT and the baselines) typically takes 0.5-2 days while downstream tasks typically only train for minutes to a few hours per run. While we did not study the exact energy consumption, we use the training times as an estimate and conclude that the energy consumption, and therefore the environmental impact, during pre-training is substantially higher than during finetuning. We observe that in our setting most studied pre-training methods, including LoVT, have similar runtimes (1-2 days, depending on the exact hyperparameters, for training on the full dataset) and only CheXpert pre-training is significantly faster (typically taking 0.5-1 days for pre-training). Besides training, also hyperparameter tuning needs to be considered, which may be required when applying LoVT to another pre-training dataset.

While, as we observed, the high energy consumption of pre-training is an effect that is general to many methods, it should still be considered when deciding whether and how to use LoVT. An approach to reduce the energy consumption is to limit the hyperparameter tuning of LoVT on the pre-training dataset (e.g. only tune the learning rate) and use the defaults from our paper for most hyperparameters, although tuning other hyperparameters may improve downstream results. Instead, hyperparameter tuning could be more focused on the finetuning of the model. Additionally, pre-trained models should be made publicly available where this does not lead to privacy issues.

4 Detailed Discussion of Related Work

4.1 Self-supervised Representation Learning

State-of-the-art methods for pre-training image models using self-supervised representation learning can be categorized into *generative* and *discriminative* approaches. Generative models learn a distribution over the training images and a latent representation space. Typically, these approaches are autoencoding models[78,42,56,64], which learn to reconstruct the input image (or parts of it), adversarial models[25,15,19,4,16], where data and representation are modeled jointly, autoregressive models[58,57], where image regions are conditioned on previous image regions, or flow-based models[12,13,40], which estimate high-dimensional densities from data. Generative models can recover the original data

distribution without the need for assumptions on downstream tasks and are therefore well-suited for a wide range of applications, most notably for generative tasks[46]. However, they have some inherent problems, most notably, they model the distribution in the data space (e.g. in pixel-space) and therefore focus too much on low-level details (like pixels) instead of encouraging high-level abstractions that are typically required for discriminative downstream tasks like classification[46].

Discriminative approaches are better suited for such tasks as they define discriminative objectives based on pretext tasks created from the unlabeled data. Early discriminative approaches relied on heuristics when defining the pretext tasks[18,14,90,55,22], limiting the generality of the resulting representations. In recent years, contrastive approaches[85,59,32,29,53,43,7,28,27,8,89,3,20], have become the state-of-the-art discriminative approaches for self-supervised representation learning. Contrastive methods act in the representation space and try to align representations of similar images (e.g. different views from the same image) while spreading representations of different images. Clustering approaches like DeepCluster[5] also belong to the contrastive approaches. Discriminative approaches are in general very lightweight and contrastive methods are currently state-of-the art for discriminative downstream tasks. However, they are not suited for generative tasks and many aspects, like the need for negative sampling, are not well-understood yet although being tackled by approaches like BYOL[27], SimSiam[8], BarlowTwins[89], and VICReg[3]. Contrastive methods have also been successfully applied to medical imaging including image classification on chest X-rays[21,71,72].

Most contrastive learning approaches use instance-level contrast, i.e. represent each view of the image by a single vector. While the resulting representations are well-suited for global downstream tasks like image classification they lack properties like spatial sensitivity or smoothness required for more localized downstream tasks (like segmentation or detection)[87]. Therefore, there is a number of recent approaches that use region-level contrast[87,86,83,6,61,52], i.e. they act on representations of image regions. These approaches are more suited for localized tasks and therefore typically outperform instance-level methods on such tasks.

4.2 Multimodal Representation Learning

While self-supervised representation learning on a single modality (e.g. images) already achieves great results, in some settings more modalities are available. Utilizing such additional modalities can improve the downstream results as additional information is available that can be utilized during representation learning. One form of such additional modalities is text, that often accompanies images in the form of captions or linked reports. Early works on combining image and text modalities did not focus on pre-training for downstream tasks but instead on learning aligned representations for cross-modal retrieval[80,24], on encoder-decoder tasks like image captioning[79,39,17], and on joint prediction tasks like visual question answering[26] and visual grounding[9,30]. In recent

years, several works utilized transformer[77] models to compute joint representations of image content and text[74,50,73,75,92,31]. They pre-trained their models using self-supervised tasks like multi-modal alignment prediction on paired image-text datasets and then finetuned them on multi-modal downstream tasks like image retrieval or visual question answering. While these methods can effectively pre-train joint image-text models, these models cannot be used for image-only downstream tasks. Recently, there is much focus on self-supervised representations learning methods that pre-train image models for downstream tasks by taking advantage of the companion text[63,35,91,10,68]. VirTex[10] and ICMLM[68] use image captioning tasks (generative tasks), ConVIRT[91], CLIP[63] and ALIGN[35] use multiview contrastive learning[2] (contrastive tasks). In[82] generative and contrastive losses are combined to train on mixed chest X-ray data, i.e. where only for some images paired reports is available. LocTex[47] does localized pre-training on natural images with companion text using a dot product based model to predict alignment of text and image regions. Unlike our method it uses supervision generated by mouse gazes instead of learning the alignment implicitly using a local contrastive loss. Most related to our work is the recently published local-mi[44] that does contrastive learning on report sentences and image regions but aligns each sentence with its most related region instead of using an alignment model like our method. Also, it targets classification instead of localized tasks and does therefore neither encourage contrast between regions nor spatial smoothness.

5 Experiment Details

In all our experiments we use PyTorch[60] Version 1.10 (BSD-style license[5]) and train on a single NVIDIA Quadro RTX 8000.

5.1 Pre-Training

Pre-training Data and Pre-Processing Our method can be used with any dataset containing pairs of medical images and reports supposing that the reports contain multiple sentences and the sentences in the reports provide a semantically useful description of the contents in the image. We use version 2 of MIMIC-CXR[38,37,36,23] as, to our best knowledge, it is the largest and most commonly used dataset of this kind conatining more than $200,000$ imaging studies, each with one or more frontal or lateral chest X-rays and one semi-structured free-text radiology report, written by a practicing radiologist during routine clinical care, describing radiological findings of the images.

We download the already pre-processed images from its JPG-version[6] and remove all except the frontal views, i.e. we only keep the *antero-posterior (AP)*

[5] `https://github.com/pytorch/pytorch/blob/master/LICENSE`

[6] `https://physionet.org/content/mimic-cxr-jpg/2.0.0/` (PhysioNet Credentialed Health Data License)

and *postero-anterior (PA)* views. We download the reports from MIMIC-CXR[7] and extract the text from the *Findings* and *Impression* sections. Reports containing none of these sections are removed. For each report we concatenate the extracted text from both sections and remove reports where the extracted text contains less than three tokens (based on tokenizing it using Stanza[62]). We split the extracted text into sentences using Stanza[62] again. Finally, we remove all samples that contain no images or no report (after the previous steps) and then apply the training/validation/test splits provided by MIMIC-CXR-JPG[36] such that we have 210228/1712/2867 training/validation/test samples (i.e. studies with one report and one or more images each), respectively.

Encoders and Model Details In the image encoder we use the ResNet50 implementation from Torchvision[8] and initialize it with ImageNet[67] weights[9]. In the report encoder we use the BERT_base PyTorch implementation from Huggingface Transformers[84][10] and initialize it with weights from ClinicalBERT[1][11] that was trained on clinical notes.

We model the nonlinear transformations $f^{\mathcal{I}}$, $f^{\mathcal{R}}$, $\bar{f}^{\mathcal{I}}$, and $\bar{f}^{\mathcal{R}}$ as shallow neural networks without shared parameters, consisting of a (element-wise) linear layer with output size 2048, batch norm, and ReLU followed by another linear layer with output size 512. This follows previous works[7,27,91] except for the batch norm which we found beneficial.

Data Augmentation For image augmentations we first randomly sample one of the frontal chest X-rays of the sample (i.e. study) and then follow the augmentation scheme of ConVIRT[91], i.e. random cropping (resized to 224×224), horizontal flipping, affine transformations, contrast and brightness jittering and Gaussian blur. We also tried removing geometric augmentations but found this setting to perform worse. For text augmentations we concatenate all sentences of the Findings and Assessment sections of the report in the sample but randomly change the order by swapping pairs of sentences with a probability of 0.6. We also tried randomly removing or duplicating sentences but did not find it to be beneficial.

Training Details and Cyclic Cosine Learning Rate Schedule For pre-training, we experimented with different learning rate schedules and found that a cyclic cosine learning rate schedule[48] where the restarts also follow the cosine function and with a cycle length of two epochs (i.e. one decreasing and one increasing epoch) is beneficial. As both modalities have different properties (e.g. type of contained information) and the encoders have different architectures, they may converge at different speeds making it hard for both to adapt to each other. Therefore, we

[7] `https://physionet.org/content/mimic-cxr/2.0.0/` (PhysioNet Credentialed Health Data License)
[8] `https://github.com/pytorch/vision` (BSD 3-Clause License)
[9] `https://pytorch.org/hub/pytorch_vision_resnet/` (BSD 3-Clause License)
[10] `https://github.com/huggingface/transformers` (Apache-2.0 License)
[11] `https://huggingface.co/emilyalsentzer/Bio_ClinicalBERT` (MIT License)

assume that the decreasing phase of the schedule allows the encoders to catch up and adapt to each other while the increasing phase allows them to learn faster and escape local optima. We use the AdamW[49] optimizer with a batch size of 32, with 16 gradient accumulation steps, an initial learning rate of 1×10^{-4}, and weight decay 1×10^{-6} and train until the validation loss does not decrease for 10 consecutive epochs.

Hyperparameter Tuning We tune the hyperparameters of LoVT using only the *RSNA YOLOv3 Frozen 10%* task (see Sec. 5.3). For the hyperparameters of the global loss, i.e. τ and λ, and for the global representation dimension $\bar{d}^{\mathcal{Z}}$ we use the default values from ConVIRT[91]. In preliminary experiments, we tried different values for the local representation dimension $d^{\mathcal{Z}}$ but did not find that small changes to it have significant influence on the results, and therefore set $d^{\mathcal{Z}} = \bar{d}^{\mathcal{Z}}$ (i.e. 512). We determined the hyperparameters of the local losses, i.e. τ', β, and T, in preliminary experiments including grid searches and manual tuning. The loss weights γ, μ, and ν were determined by running small grid searches in the following way: We first set $\gamma = \mu = \nu = 1$ and run a grid search to balance the local loss weights μ and ν while keeping γ fixed, i.e. trying $(\mu = 0.5, \nu = 1.5)$, $(\mu = 1.0, \nu = 1.0)$, and $(\mu = 1.5, \nu = 0.5)$. After we found that $(\mu = 1.0, \nu = 1.0)$ performs best, we ran a grid search to balance local and global losses while keeping $\mu = \nu$, i.e. trying $(\gamma = 0.75, \mu = 1.0, \nu = 1.0)$, $(\gamma = 1.0, \mu = 1.0, \nu = 1.0)$, $(\gamma = 1.0, \mu = 0.75, \nu = 0.75)$, and $(\gamma = 1.0, \mu = 0.25, \nu = 0.25)$. We found that $(\gamma = 1.0, \mu = 0.75, \nu = 0.75)$ performs best.

All hyperparameters except the learning rate are tuned using 30% of the pre-training dataset and we tune the learning rate individually on 30% and 100% of the pre-training data. Note that we also slightly tune the learning rate when tuning other hyperparameters and in our ablation study.

5.2 Baselines

Random and ImageNet Init. For random initialization we do not pre-train the ResNet50 backbone but instead initialize it randomly following its default initialization. For the ImageNet initialization we use the weights[9] provided by Torchvision.

CheXpert We train the ResNet50 backbone using multi-label binary classification on MIMI-CXR. We use five CheXpert[33] labels (Cardiomegaly, Edema, Consolidation, Atelectasis, and Pleural Effusion), which are included in the MIMIC-CXR-JPG[36] dataset, and convert them to binary labels following the *U-Ones* mapping[33] (i.e. mapping all uncertain labels to positive labels). During CheXpert pre-training we use the full ResNet50 model including the average pooling and the fully connected (FC) layer but throw away the latter two for downstream tasks. All layers except the FC layer are initialized from ImageNet weights[9] and we randomly initialize the FC layer such that it has an output dimension of five (matching the number of classes). We use the sigmoid activation on the outputs, multi-label binary cross-entropy loss and the Adam[41] optimizer

and train with batch size 64 and weight decay 1×10^{-6} until the validation *Area Under Receiver Operating Characteristic (AUROC)* does not increase for 10 consecutive epochs after which we select the checkpoint with the best validation AUROC. We tuned the initial learning rate and set it to 3×10^{-4} (1×10^{-4} when trained on 30% of the data). If the validation AUROC does not increase for three consecutive epochs we multiply the current learning rate by 0.5.

SimCLR[7] We use the PyTorch implementation available at `https://github.com/spijkervet/SimCLR` (MIT License) with the default image augmentations from the paper (with images resized to 224×224) except for color jittering where we do not adjust saturation and hue due to the monochrome nature of chest X-rays. Following [91] we set the output dimension to 128 and hidden size to 4096, use batch size 128, and weight decay 1×10^{-4} For training, we use the Adam[41] optimizer and the cosine decay learning rate schedule[48], without restarts, over 100 epochs, with a single warm-up epoch. We tuned the initial learning rate and set it to 3×10^{-4}

BYOL[27] We use the PyTorch implementation available at `https://github.com/lucidrains/byol-pytorch` (MIT License) with the default image augmentations from the paper (with images resized to 224×224) except for color jittering where we do not adjust saturation and hue due to the monochrome nature of chest X-rays. We set the output dimension to 128 and hidden size to 4096, use decay rate 0.99, batch size 64, and weight decay 1×10^{-4} For training, we use the Adam[41] optimizer and the cosine decay learning rate schedule[48], without restarts, over 100 epochs, with a single warm-up epoch. We tuned the initial learning rate and set it to 1×10^{-4} (3×10^{-5} when trained on 30% of the data).

PixelPro[87] We use the PyTorch implementation available at `https://github.com/lucidrains/pixel-level-contrastive-learning` (MIT License) with the default image augmentations from the paper (with images resized to 224×224) except for color jittering where we do not adjust saturation and hue due to the monochrome nature of chest X-rays.

We set the output dimension to 512 and hidden size to 2048, use batch size 64, and weight decay 1×10^{-5} For training, we use the Adam[41] optimizer and the cosine decay learning rate schedule[48], without restarts, over 100 epochs, with a single warm-up epoch. We tuned the initial learning rate and set it to 1×10^{-3}

ConVIRT[91] We use our own implementation of ConVIRT (as the general framework of ConVIRT is similar to LoVT) and train until the validation loss does not decrease for 15 consecutive epochs after which we use the checkpoint with the lowest validation loss. We tuned the learning rate and set it to 1×10^{-4} (1×10^{-5} when trained on 30% of the data). If the validation loss does not decrease for 12 consecutive epochs we multiply the current learning rate by 0.5. We use the default values from the paper for all other hyperparameters.

CLIP[63] We use our own implementation of CLIP (as the general framework of CLIP is similar to LoVT). For better comparability with LoVT and the other baselines, we use ResNet50 and BERT_base as encoders. Following the framework of CLIP we only encode single sentences and therefore randomly sample a sentence from the report (as in ConVIRT). We use the AdamW[49] optimizer with a batch size of 32 (the same as used in ConVIRT and LoVT), with 16 gradient accumulation steps, and the cyclic cosine learning rate scheduler (as in LoVT) with an initial learning rate of 1×10^{-4} , and weight decay 1×10^{-6} and train until the validation loss does not decrease for 10 consecutive epochs.

Batch Sizes of the Baselines Most contrastive learning methods are very sensitive to the used batch size, therefore the batch size is an important hyperparameter when comparing such methods. However, increasing the batch size also increases the GPU memory consumption and different methods have different memory requirements, such that using the same batch size for all methods does not allow for a fair comparison as in practice available GPU memory is typically limited. We therefore decided to use three different batch sizes: The smallest batch size (32[12]) is used for all text-supervised methods (i.e. ConVIRT, CLIP and our LoVT) as they require much memory due to their language model and they are also less sensitive to the batch size[91]. For image-only methods with a momentum encoder (i.e. BYOL and PixelPro) we use a larger batch size (64) and for SimCLR we further increase the batch size (128) as it does not have a momentum encoder and is very sensitive to the used batch size.

5.3 Downstream Evaluation

Datasets

- **RSNA Pneumonia Detection**[81,69] (Licensed following the competition rules[13]): We download the dataset from its Kaggle page[14] but use only their training set which we randomly split into our training, validation, and test set resulting in 16010/5337/5337 training/validation/test samples, respectively. For each sample we compute a segmentation mask (used in the *Linear* evaluation) from all the ground truth detection boxes of that sample.
- **COVID Rural**[76,11] (TCIA Data Usage Policy and CC BY 4.0 License): We download the dataset from its Github repository[15] and randomly split it into training, validation, and test set of sizes 133/44/44, respectively.
- **SIIM-ACR Pneumothorax Segmentation**[70] (Licensed following the competition rules[16]): We download the dataset from its Kaggle re-upload[17],

[12] We use this batch size as it was used in ConVIRT and as memory requirements are then kept below 24GB allowing training on widely used GPUs.

[13] `https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/rules`

[14] `https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data`

[15] `https://github.com/haimingt/opacity_segmentation_covid_chest_X_ray/tree/master/covid_rural_annot`

[16] `https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/rules`

[17] `https://www.kaggle.com/seesee/siim-train-test/`

which is officially recommended on the original challenge website[18], but use only their training set which we randomly split into our training, validation, and test set resulting in 7229/2409/2409 training/validation/test samples, respectively.
- **Object CXR**[34] (CC BY-NC 4.0 License): We download the dataset from a re-upload[19] as it is no longer available at its original source[20]. We randomly split their training set into our training and validation set and use their development set as our test set such that we have 6400/1600/1000 training/validation/test samples, respectively. For each sample we compute a segmentation mask (used in the *Linear* evaluation) from all the ground truth detection boxes of that sample.
- **NIH CXR**[81] (Licensed for public use with attribution[21]): We download the ChestX-ray8 dataset provided by the NIH Clinical Center from its official website[22] but use only the samples where bounding boxes are provided as ground truth. We randomly split these samples into our training, validation, and test set such that we have 588/196/196 training/validation/test samples, respectively.

5.4 Evaluation Protocols

In this section we describe the details of the evaluation protocols used in the evaluation framework[54], including downstream model architectures and training details. Note that we do not use image augmentations in any of the evaluation protocols but resize and pad the input images to size 224×224.

U-Net Finetune We do not use the original UNet[66] architecture but instead build a UNet-like model[23] based on the pre-trained ResNet50 model. Therefore, we use the ResNet50 (except its avg pooling and FC layer) as the contracting path (left side) of our UNet-like model. The last feature map of ResNet50 has a size of 7×7 and dimension 2048. Here we add two more convolutional blocks (each with a 3×3 convolution followed by batchnorm and ReLU) with output dimension 2048 to the contracting path. For the expansive path (right side) we closely follow the architecture of the original UNet but use five instead of four upsampling blocks (each with 2×2 transposed convolution, concatenation, and two 3×3 convolutions each followed by batchnorm and ReLU) which have output dimensions 1024, 512, 256, 128, and 64, respectively. For concatenation

[18] https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/overview/siim-cloud-healthcare-api-tutorial
[19] https://academictorrents.com/details/fdc91f11d7010f7259a05403fc9d00079a09f5d5
[20] https://jfhealthcare.github.io/object-CXR/
[21] https://nihcc.app.box.com/v/ChestXray-NIHCC/file/249502714403
[22] https://nihcc.app.box.com/v/ChestXray-NIHCC/
[23] Our implementation is based on https://github.com/kevinlu1211/pytorch-unet-resnet-50-encoder/blob/master/u_net_resnet_50_encoder.py (MIT License)

the ResNet50 blocks conv4, conv3, conv2, conv1, and the input image are used. We add a single 1×1 convolution that predicts the positive class scores and use the binary Dice loss from the Segmentation Models Pytorch library[88].

For training, we use the Adam[41] optimizer with weight decay 1×10^{-6} The learning rate is tuned individually for each model and task based on the best validation Dice[24]. The learning rate is multiplied by 0.5 if the validation Dice does not decrease for three consecutive epochs. We use a warmup period in which we do not train the ResNet50 backbone but only the other, randomly initialized, layers with a learning rate of 1×10^{-3} after which we train the whole model (including the ResNet50). On the COVID Rural dataset we use batch size eight, a warmup period of 20 iterations and do early stopping (based on validation Dice) after 20 epochs. On the SIIM-ACR Pneumothorax dataset we use batch size 64, a warmup period of 100 iterations and do early stopping after 10 epochs. Finally we report the test Dice of the epoch with the best validation Dice.

U-Net Frozen We use the same architecture and loss function as in the *U-Net Finetune* protocol but freeze the pre-trained ResNet50 weights and never train them. Instead we only train the other layers using the same hyperparameters as in the *U-Net Finetune* protocol (except for the warmup period which is not relevant in this setting).

Linear We use the frozen pre-trained ResNet50 (except for its avg pooling and FC layer) to compute 7×7 feature maps. A randomly initialized element-wise linear layer (i.e. a 1×1 convolution) is applied to these feature maps and the results are upsampled to the segmentation resolution using bilinear interpolation to predict the class scores. We then use the binary Dice loss from the Segmentation Models Pytorch library[88]. For the NIH CXR dataset we train each class independently using the binary Dice loss.

For detection tasks we first compute segmentation masks from the detection ground truth using the union of all target bounding boxes per sample and then interpret the task as a segmentation task. Note that for the Object CXR dataset we create bounding box masks only for box and ellipse detection targets but use polygon masks for polygon detection targets.

For training, where we only train the linear layer, we use the Adam[41] optimizer with weight decay 1×10^{-6} The learning rate is tuned individually for each model and task based on the best validation Dice[24]. The learning rate is multiplied by 0.5 if the validation Dice does not decrease for three consecutive epochs. On the *COVID Rural Linear* and the *RSNA Lin. Seg. 1%* tasks we use batch size eight and do early stopping (based on validation Dice) after 20 epochs. On all other *Linear* tasks we use batch size 64 and do early stopping after 10 epochs. Finally we report the test Dice of the epoch with the best validation

[24] We use the micro-averaged Dice score based on this implementation: `https://torchmetrics.readthedocs.io/en/latest/references/modules.html#f1` (Apache-2.0 License)

Dice. Note that for the *NIH CXR Linear* task we use the *macro averaged Dice (Avg Dice)* as metric.

YOLOv3 Finetune We closely follow the architecture[25] of the original YOLOv3[65] but use the pre-trained ResNet50 as its backbone (replacing the Darknet-53 backbone) while randomly initializing all other layers. The backbone features for the three prediction scales are extracted from the outputs of the conv3 (highest resolution), conv4, and conv5 (lowest resolution) blocks of ResNet50, respectively. We use the default anchors presented in their paper but scale them according to our image input size of 224×224.

For training, we use the losses and loss weights from the YOLOv3 paper and train with the Adam[41] optimizer with weight decay 1×10^{-6} The learning rate is tuned individually for each model and task based on the best validation *mean Average Precision (mAP)*. We compute[26] the mAP score following the COCO[45] mAP and with the following Intersection over Union (IoU) thresholds: $0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75$. The learning rate is multiplied by 0.5 if the validation mAP does not decrease for three consecutive epochs. We use a warmup period of 100 iterations in which we do not train the ResNet50 backbone but only the other layers with a learning rate of 1×10^{-3} after which we train the whole model (including the ResNet50). On the *RSNA YOLOv3 Finetune 1%* task we use batch size eight and do early stopping (based on validation mAP) after 20 epochs. On all other *YOLOv3 Finetune* tasks we use batch size 64 and do early stopping after 10 epochs. Finally we report the test mAP of the epoch with the best validation mAP.

YOLOv3 Frozen We use the same architecture and loss functions as in the *YOLOv3 Finetune* protocol but freeze the pre-trained ResNet50 weights and never train them. Instead we only train the other layers using the same hyperparameters as in the *YOLOv3 Finetune* protocol (except for the warmup period which is not relevant in this setting).

References

1. Alsentzer, E., Murphy, J.R., Boag, W., Weng, W.H., Jin, D., Naumann, T., McDermott, M.B.A.: Publicly available clinical bert embeddings. In: ClinicalNLP (2019)
2. Bachman, P., Hjelm, R., Buchwalter, W.: Learning representations by maximizing mutual information across views. In: NeurIPS (2019)
3. Bardes, A., Ponce, J., LeCun, Y.: Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In: ICLR (2022)
4. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv: 1809.11096 (2019)
5. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: European Conference on Computer Vision (2018)

[25] Our implementation is based on `https://github.com/BobLiu20/YOLOv3_PyTorch`

[26] `https://github.com/bes-dev/mean_average_precision` (MIT License)

6. Chaitanya, K., Erdil, E., Karani, N., Konukoglu, E.: Contrastive learning of global and local features for medical image segmentation with limited annotations. In: NeurIPS (2020)
7. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML (2020)
8. Chen, X., He, K.: Exploring simple siamese representation learning. In: CVPR. pp. 15745–15753 (2021). https://doi.org/10.1109/CVPR46437.2021.01549
9. Deng, C., Wu, Q., Wu, Q., Hu, F., Lyu, F., Tan, M.: Visual grounding via accumulated attention. In: CVPR (2018)
10. Desai, K., Johnson, J.: Virtex: Learning visual representations from textual annotations. In: CVPR. pp. 11157–11168 (2021). https://doi.org/10.1109/CVPR46437.2021.01101
11. Desai, S., Baghal, A., Wongsurawat, T., Al-Shukri, S., Gates, K., Farmer, P., Rutherford, M., Blake, G., Nolan, T., et al.: Data from chest imaging with clinical and genomic correlates representing a rural covid-19 positive population [data set]. The Cancer Imaging Archive (2020). https://doi.org/https://doi.org/10.7937/tcia.2020.py71-5978
12. Dinh, L., Krueger, D., Bengio, Y.: Nice: Non-linear independent components estimation. arXiv preprint arXiv: 1410.8516 (2015)
13. Dinh, L., Sohl-Dickstein, J., Bengio, S.: Density estimation using real nvp. arXiv preprint arXiv: 1605.08803 (2017)
14. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: IEEE International Conference on Computer Vision. p. 1422–1430 (2015)
15. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv preprint arXiv: 1605.09782 (2017)
16. Donahue, J., Simonyan, K.: Large scale adversarial representation learning. In: NIPS (2019)
17. Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR (2015)
18. Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with convolutional neural networks. In: NIPS (2014)
19. Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference. arXiv preprint arXiv: 1606.00704 (2017)
20. Ermolov, A., Siarohin, A., Sangineto, E., Sebe, N.: Whitening for self-supervised representation learning. In: ICML. pp. 3015–3024 (2021)
21. Gazda, M., Plavka, J., Gazda, J., Drotár, P.: Self-supervised deep convolutional neural network for chest x-ray classification. IEEE Access pp. 151972–151982 (2021). https://doi.org/10.1109/ACCESS.2021.3125324
22. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. arXiv preprint arXiv: 1803.07728 (2018)
23. Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., et al.: Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. Circulation [Online] **101**(23), 215—220 (2000)
24. Gomez, R., Gomez, L., Gibert, J., Karatzas, D.: Chapter 9 - self-supervised learning from web data for multimodal retrieval. In: Multimodal Scene Understanding (2019)
25. Goodfellow, I., Pouget-Abadie, J., et al.: Generative adversarial nets. In: NIPS (2014)

26. Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., Parikh, D.: Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In: CVPR (2017)
27. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., Valko, M.: Bootstrap your own latent - a new approach to self-supervised learning. In: NeurIPS (2020)
28. He, K., Fan, H., Wu, Y., et al.: Momentum contrast for unsupervised visual representation learning. In: CVPR. pp. 9726–9735 (2020). https://doi.org/10.1109/CVPR42600.2020.00975
29. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. In: ICLR (2019)
30. Hu, R., Xu, H., Rohrbach, M., Feng, J., Saenko, K., Darrell, T.: Natural language object retrieval. In: CVPR (2016)
31. Huang, Z., Zeng, Z., Liu, B., Fu, D., Fu, J.: Pixel-bert: Aligning image pixels with text by deep multi-modal transformers. arXiv preprint arXiv: 2004.00849 (2020)
32. Hénaff, O.J., Srinivas, A., et al.: Data-efficient image recognition with contrastive predictive coding. In: ICML. pp. 4182–4192 (2019)
33. Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., Seekins, J., Mong, D., Halabi, S., Sandberg, J., Jones, R., Larson, D., Langlotz, C., Patel, B., Lungren, M., Ng, A.: Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In: AAAI. pp. 590–597 (2019)
34. JF-Healthcare: object-cxr - automatic detection of foreign objects on chest x-rays. MIDL (2020), `https://jfhealthcare.github.io/object-CXR/`
35. Jia, C., Yang, Y., Xia, Y., Chen, Y.T., Parekh, Z., Pham, H., Le, Q., Sung, Y.H., Li, Z., Duerig, T.: Scaling up visual and vision-language representation learning with noisy text supervision. In: ICML (2021)
36. Johnson, A., Lungren, M., Peng, Y., et al.: Mimic-cxr-jpg - chest radiographs with structured labels (version 2.0.0). PhysioNet (2019). https://doi.org/https://doi.org/10.13026/8360-t248
37. Johnson, A., Pollard, T., Berkowitz, S., et al.: Mimic-cxr, a de-identified publicly available database of chest radiographs with free-text reports. Sci Data **6**(317) (2019). https://doi.org/https://doi.org/10.1038/s41597-019-0322-0
38. Johnson, A., Pollard, T., Mark, R., Berkowitz, S., Horng, S.: Mimic-cxr database (version 2.0.0). PhysioNet (2019). https://doi.org/https://doi.org/10.13026/C2JT1Q
39. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: CVPR (2015)
40. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. In: NIPS. pp. 10215—-10224 (2018)
41. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2014)
42. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv: 1312.6114 (2014)
43. Li, J., Zhou, P., Xiong, C., Hoi, S.C.H.: Prototypical contrastive learning of unsupervised representations. In: ICLR (2021)
44. Liao, R., Moyer, D., Cha, M., et al.: Multimodal representation learning via maximization of local mutual information. In: MICCAI. pp. 273–283 (2021). https://doi.org/10.1007/978-3-030-87196-3_26

45. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV (2014)
46. Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., Tang, J.: Self-supervised learning: Generative or contrastive. IEEE Trans Knowl Data Eng (2021). https://doi.org/10.1109/TKDE.2021.3090866
47. Liu, Z., Stent, S., Li, J., Gideon, J., Han, S.: Loctex: Learning data-efficient visual representations from localized textual supervision. In: ICCV. pp. 2147–2156 (2021). https://doi.org/10.1109/ICCV48922.2021.00217
48. Loshchilov, I., Hutter, F.: Sgdr: stochastic gradient descent with warm restarts. In: ICLR (2017)
49. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)
50. Lu, J., Batra, D., Parikh, D., Lee, S.: Vilbert: pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In: NeurIPS (2019)
51. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research **9**, 2579–2605 (2008)
52. Mahendran, A., Thewlis, J., Vedaldi, A.: Cross pixel optical-flow similarity for self-supervised learning. In: ACCV 2018 (2019)
53. Misra, I., van der Maaten, L.: Self-supervised learning of pretext-invariant representations. In: CVPR. pp. 6706–6716 (2020). https://doi.org/10.1109/CVPR42600.2020.00674
54. Müller, P., Kaissis, G., Zou, C., Rueckert, D.: Radiological reports improve pre-training for localized imaging tasks on chest x-rays. In: [to be published at] MICCAI (2022)
55. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: European Conference on Computer Vision. p. 69–84 (2016)
56. van den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. In: NIPS. pp. 6306—-6315 (2017)
57. den Oord, A.V., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al.: Conditional image generation with pixelcnn decoders. In: NIPS. pp. 4790—4798 (2016)
58. Oord, A.V., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: ICML. pp. 1747—-1756 (2016)
59. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv: 1807.03748 (2019)
60. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019)
61. Pinheiro, P.O., Almahairi, A., Benmalek, R.Y., Golemo, F., Courville, A.: Unsupervised learning of dense visual representations. In: NeurIPS (2020)
62. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A Python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020), `https://nlp.stanford.edu/pubs/qi2020stanza.pdf`
63. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: ICML. pp. 8748–8763 (2021)
64. Razavi, A., van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. In: NIPS. pp. 14837—-14847 (2019)

65. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv: 1804.02767 (2018)
66. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. pp. 234–241 (2015). https://doi.org/10.1007/978-3-319-24574-4_28
67. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
68. Sariyildiz, M.B., Perez, J., Larlus, D.: Learning visual representations with caption annotations. In: ECCV. pp. 153–170 (2020). https://doi.org/10.1007/978-3-030-58598-3_10
69. Shih, G., Wu, C.C., Halabi, S.S., Kohli, M.D., Prevedello, L.M., Cook, T.S., Sharma, A., Amorosa, J.K., Arteaga, V., Galperin-Aizenberg, M., et al.: Augmenting the national institutes of health chest radiograph dataset with expert annotations of possiblepneumonia. Radiology: Artificial Intelligence **1** (2019). https://doi.org/https://doi.org/10.1148/ryai.2019180041
70. Society for Imaging Informatics in Medicine: Siim-acr pneumothorax segmentation. `https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation` (2019)
71. Sowrirajan, H., Yang, J., Ng, A.Y., Rajpurkar, P.: Moco pretraining improves representation and transferability of chest x-ray models. In: MIDL (2021)
72. Sriram, A., Muckley, M., Sinha, K., Shamout, F., Pineau, J., Geras, K.J., Azour, L., Aphinyanaphongs, Y., Yakubova, N., Moore, W.: Covid-19 prognosis via self-supervised representation learning and multi-image prediction. arXiv preprint arXiv: 2101.04909 (2021)
73. Su, W., et al.: Vl-bert: pre-training of generic visual-linguistic representations. In: ICLR (2020)
74. Sun, C., Myers, A., Vondrick, C., Murphy, K., Schmid, C.: Videobert: a joint model for video and language representation learning. In: ICCV (2019)
75. Tan, H., Bansal, M.: Lxmert: learning cross-modality encoder representations from transformers. In: EMNLP (2019)
76. Tang, H., Sun, N., Li, Y.: Segmentation model of the opacity regions in the chest x-rays of the covid-19 patients in the us rural areas and the application to the disease severity. medRxiv (2020). https://doi.org/https://doi.org/10.1101/2020.10.19.20215483
77. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: NIPS (2017)
78. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: ICML (2008)
79. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: A neural image caption generator. In: CVPR (2015)
80. Wang, L., Li, Y., Lazebnik, S.: Learning deep structure-preserving image-text embeddings. In: CVPR (2016)
81. Wang, X., Peng, Y., Lu, L., et al.: Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: CVPR. pp. 3462–3471 (2017). https://doi.org/10.1109/CVPR.2017.369
82. Wang, X., Xu, Z., Tam, L., Yang, D., Xu, D.: Self-supervised image-text pre-training with mixed data in chest x-rays. arXiv preprint arXiv: 2103.16022 (2021)

83. Wang, X., Zhang, R., Shen, C., Kong, T., Li, L.: Dense contrastive learning for self-supervised visual pre-training. In: CVPR. pp. 3023–3032 (2021). https://doi.org/10.1109/CVPR46437.2021.00304
84. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: EMNLP (2020)
85. Wu, Z., Xiong, Y., Yu, S., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: CVPR. pp. 3733–3742 (2018). https://doi.org/10.1109/CVPR.2018.00393
86. Xie, E., Ding, J., Wang, W., Zhan, X., Xu, H., Sun, P., Li, Z., Luo, P.: Detco: Unsupervised contrastive learning for object detection. In: ICCV. pp. 8372–8381 (2021). https://doi.org/10.1109/ICCV48922.2021.00828
87. Xie, Z., Lin, Y., Zhang, Z., Cao, Y., Lin, S., Hu, H.: Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In: CVPR. pp. 16679–16688 (2021). https://doi.org/10.1109/CVPR46437.2021.01641
88. Yakubovskiy, P.: Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch` (2020)
89. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: ICML (2021)
90. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: European conference on computer vision. p. 649–666 (2016)
91. Zhang, Y., Jiang, H., Miura, Y., Manning, C.D., Langlotz, C.P.: Contrastive learning of medical visual representations from paired images and text. arXiv preprint arXiv: 2010.00747 (2020)
92. Zhou, L., Palangi, H., Zhang, L., Hu, H., Corso, J., Gao, J.: Unified vision-language pre-training for image captioning and vqa. In: AAAI (2020)