

InvCoSS: Inversion-driven Continual Self-supervised Learning in Medical Multi-modal Image Pre-training

Supplementary Material

A. Datasets

A.1. Pre-training

MIMIC-CXR 2.0.0 Dataset: The MIMIC-CXR 2.0.0 [19] dataset is a large-scale, publicly available collection of chest X-ray images. It contains a total of 377,110 JPEG chest radiographs, which are paired with 227,827 clinical reports. Following the standard preprocessing steps described in [43, 59], all lateral-view chest X-ray images were excluded, as downstream tasks focus solely on frontal-view chest radiographs. After this filtering process, the dataset was reduced to a total of 356,309 frontal-view chest X-ray images. This dataset is commonly used for tasks such as disease classification, image analysis, and other chest X-ray-related downstream applications.

DeepLesion dataset: This dataset comprises 10,594 CT scans from 4,427 participants. We standardized all scans to $1.0 \times 1.0 \times 1.0$ mm spacing, consistent with the methodology in [54]. To generate training samples, we divided each scan into smaller volumes using a sliding window approach, with window sizes of 24 slices and 12-slice intervals along the axial direction. These extracted segments were then normalized to dimensions of $16 \times 192 \times 192$ voxels, yielding a total of 125,070 sub-volumes for analysis. Intensity values were normalized by mapping the Hounsfield unit range of $[-1024, 1024]$ to $[-1, 1]$.

ADNI dataset: We compiled our ADNI dataset by combining scans from the ADNI-1, ADNI-2, and ADNI-GO collections. Subject inclusion was determined purely by diagnostic classification (Alzheimer’s disease, mild cognitive impairment, or normal cognition), with no restrictions on patient demographics or scanner specifications. From each MRI scan, we generated multiple sub-volumes through a segmentation process that extracted 16-slice windows with a stride of 8 slices along the depth direction. After resizing these segments to standardized dimensions of $16 \times 192 \times 192$ voxels, we obtained 59,205 training samples. Intensity normalization was performed by standardizing each volume to zero mean and unit variance.

TCGA dataset: This collection encompasses histopathological images across seven tumor types from The Cancer Genome Atlas: THYM, THCA, BRCA, UCEC, UVM, OV, and MESO. We processed the whole-slide images by extracting contiguous 512×512 patches and resizing them to 224×224 resolution. From each patient’s collection of n patches, we randomly extracted a maximum of 100 patches for inclusion in the training set.

A.2. Downstream

PubMed20k dataset: We utilized a collection of 20,000 RCT abstracts containing 240,000 sentences and a vocabulary of 68,000 words. Each sentence requires assignment to one of five structural labels (background, objective, method, result, or conclusion). Following the established official data split protocol, we used only the raw text sequences for training and evaluating our sentence classification model.

ChestXR Dataset: The ChestXR [1] dataset is designed for the classification of chest X-ray images into three categories: COVID-19, pneumonia, or normal. It consists of an official split of 14,958 training images and 3,432 test images. To create a validation set, 20% of the training images were randomly selected.

QaTa-COV19-v2 (QaTa) Dataset: The QaTa-COV19-v2 [5] dataset is tailored for COVID-19-infected region segmentation in chest X-ray images. It provides an official split of 7,145 training images and 2,113 test images. Similar to [1], 20% of the training data is randomly sampled to serve as the validation set.

LiTS Dataset: The LiTS dataset [2] is a collection of 131 CT scans with annotations of liver and liver tumor segmentation. Data splits followed the protocol established in [54], and preprocessing was performed using the nnU-Net framework.

RICORD Dataset: The RICORD dataset [41] is a collection of 330 chest CT scans designated for binary classification tasks distinguishing COVID-19 positive cases from healthy controls. Each volumetric scan was standardized to $64 \times 192 \times 192$ voxel resolution during preprocessing.

Vestibular Schwannoma (VS) Dataset: The VS dataset provides 242 multi-sequence MRI scans (T2 and T1CE) for segmentation of vestibular schwannoma tumors. In this study, we selected the T1 contrast-enhanced sequence as input for our segmentation model. Data splits followed the protocol established in [54], and preprocessing was performed using the nnU-Net framework.

LA Dataset: The LA dataset comprises 100 gadolinium-enhanced cardiac MRI scans with expert annotations for left atrium segmentation. We adopted both the data splits and preprocessing protocol from [57].

NCH Dataset: The NCH dataset combines the NCT-CRC-HE-100K dataset for training and the CRC-VAL-HE-7K dataset for testing. To establish a validation set, we randomly sampled 20% of the training data from each class category.

GlaS Dataset: The GlaS dataset comprises 165 H&E-stained histopathological images from colon tissue sections, annotated as malignant or benign cases. We followed the

Algorithm 1 InvCoSS Algorithm

Input: Frozen model f_{T-1} , Saved statistics $\{\mu_t, \sigma_t\}_{t=1}^{T-1}$, Current data \mathcal{D}_T .

Output: Updated model f_T .

```
1: Stage 1: Data-Free Model Inversion
2: Initialize synthetic buffer  $\mathcal{B}_T = \emptyset$ .
3: for each previous task  $t = 1$  to  $T - 1$  do
4:   Retrieve saved feature statistics  $\{\mu_t, \sigma_t\}$ .
5:   for each inversion iteration do
6:     Update  $\mathcal{G}, z$  to minimize  $\mathcal{L}_{\text{Inv}}$  on  $f_{T-1}$  guided by
        $\{\mu_t, \sigma_t\}$  (Eq. 6).
7:   end for
8:   Generate  $\mathcal{D}_t^{\text{syn}}$  and update  $\mathcal{B}_T \leftarrow \mathcal{B}_T \cup \mathcal{D}_t^{\text{syn}}$ .
9: end for
10: Stage 2: Continual Training
11: Initialize  $f_T \leftarrow f_{T-1}$ .
12: for each training step do
13:   Update  $f_T$  using batches  $x \sim \mathcal{D}_T$  and  $x^{\text{syn}} \sim \mathcal{B}_T$  to
     minimize Eq. 4.
14: end for
15: return  $f_T$ 
```

official train-test partition and created a validation set by randomly selecting 20% of training samples from each diagnostic category.

B. Methodology Details of InvCoSS

B.1. Workflow of InvCoSS

In this section, we provide a comprehensive algorithmic description of the proposed InvCoSS framework. The core objective is to enable continual self-supervised learning without retaining raw data from previous tasks. The overall procedure, summarized in Algorithm 1, consists of two distinct phases: Data-Free Model Inversion and Continual Training. **Stage 1: Data-Free Model Inversion** (Lines 1-9). To prevent catastrophic forgetting, we construct a synthetic sample set \mathcal{B}_T that approximates the data distributions of all prior tasks $t \in \{1, \dots, T - 1\}$. Unlike traditional methods that require storing previous task samples, InvCoSS relies solely on lightweight feature statistics $\{\mu_t, \sigma_t\}$ stored for each task and the current pre-trained model f_{T-1} . As shown in Lines 3-8, for each historical task t , we optimize the generator \mathcal{G} and latent codes z to synthesize images. Crucially, the optimization is guided by matching the stored statistics $\{\mu_t, \sigma_t\}$ while passing gradients through the frozen backbone f_{T-1} (Eq. 6). This design significantly reduces storage overhead while effectively recovering domain-specific features.

Stage 2: Continual Training (Lines 10-14). Once the synthetic set \mathcal{B}_T is populated, we proceed to train the current model f_T . We initialize f_T with the weights of f_{T-1} to facilitate knowledge transfer. During training, rather than relying on raw data retention, we implement a data-free knowledge distillation strategy. Specifically, each training iteration uti-

lizes a composite mini-batch comprising real data from the current task \mathcal{D}_T and synthetic samples from \mathcal{B}_T . The model is updated by minimizing a joint objective (Eq. 4) that combines the task-specific loss for new knowledge acquisition and a Knowledge Distillation loss on synthetic samples for knowledge preservation.

B.2. Batch-wise Statistics

Conventional model inversion techniques often exploit the running statistics of Batch Normalization (BN) layers. However, this strategy is incompatible with modern BN-free architectures like Vision Transformers (ViTs). To address this, we compute batch-wise feature statistics at the end of the final training epoch to guide the inversion process. We perform a single forward pass of the trained model f_t over its entire dataset \mathcal{D}_t in evaluation mode. Statistics are extracted from the feature maps of the Transformer encoder blocks. For a feature tensor of shape (B, L, D) , we compute the mean and variance across the batch dimension B over the entire dataset, capturing the feature distribution characteristics essential for synthesizing images with coherent spatial structures. According to [47], we update the statistics matrices μ and σ^2 to obtain batch-wise statistics per-layer in Eq. 3 over mini-batches as follows:

$$(\mu, \sigma^2) = \begin{cases} (\mu_1, \sigma_1^2) & \text{if } b = 1 \\ \left(\frac{N_{b-1}\mu^{(b-1)} + n_b\mu_b}{N_b}, \right. & \text{if } b > 1 \\ \left. \frac{N_{b-1}(\sigma^2)^{(b-1)} + n_b\sigma_b^2}{N_b} \right. & \\ \left. + \frac{N_{b-1}n_b}{N_b^2}(\mu^{(b-1)} - \mu_b)^2 \right) & \end{cases} \quad (9)$$

where N_b denotes the cumulative sample count up to batch b . The aggregated batch-wise statistics are then used as targets in a feature-matching loss, constraining the generator to produce images that replicate the learned feature distributions of the original task.

B.3. Detail Architecture of InvUNet

InvUNet is a dual-stream U-Net-based generator designed for high-fidelity image inversion from low-dimensional latent codes. As shown in Fig. 8, the architecture strategically injects the noisy latent vector z at the network bottleneck, forcing it to encapsulate essential semantic information while delegating spatial upsampling to specialized branches. Two complementary pathways operate in parallel: the Memory Cache Branch generates multi-scale structural priors, while the Inversion Branch focuses on semantically guided, high-fidelity reconstruction through skip connections.

Bottleneck Injection. Rather than injecting z at the input layer, InvUNet injects it at the network bottleneck through

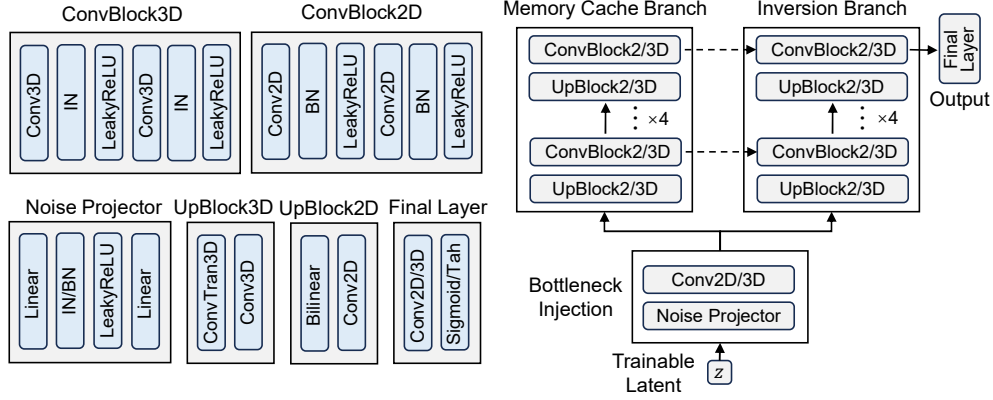


Figure 8. InvUNet architecture overview. Left panels show building blocks: ConvBlock3D and ConvBlock2D consist of two consecutive convolutions with normalization and activation; UpBlock performs $2\times$ upsampling followed by ConvBlock; Noise Projector maps latent vector to bottleneck features; Final Layer produces output. Right side shows the dual-stream architecture with Bottleneck Injection connecting to Memory Cache Branch and Inversion Branch through skip connections.

a Noise Projector module. The latent vector is first linearly projected to match the bottleneck feature map size, then passed through normalization and activation. This design forces the latent representation to encapsulate essential semantic information and prevents direct high-dimensional upsampling of raw noise.

Memory Cache Branch. The Memory Cache Branch operates as a lightweight encoder that generates multi-scale structural priors. Starting from the bottleneck, it progressively upsamples features through successive UpBlock modules. Each UpBlock performs $2\times$ upsampling followed by ConvBlock operations. For 2D images, the branch uses bilinear interpolation and 2D convolutions with Batch Normalization. For 3D volumes, it employs trilinear interpolation and 3D convolutions with Instance Normalization to maintain batch independence. The structural priors generated at each level are stored and later used as skip connections to guide the Inversion Branch.

Inversion Branch. The Inversion Branch is the primary reconstruction pathway that focuses on high-fidelity semantic-guided synthesis. It also progressively upsamples through UpBlock and ConvBlock modules, but additionally receives skip connections from the Memory Cache Branch at each corresponding resolution level. These skip connections enable the Inversion Branch to access structural guidance while focusing its capacity on refining semantic details and high-frequency information. The interplay between upsampling and skip-guided feature fusion ensures effective multi-scale reconstruction.

Final Layer. After reconstruction at full resolution, the features are processed through a final output layer. For 2D image synthesis, this consists of a Conv2d layer with Sigmoid activation to produce images in the range $[0, 1]$. For 3D volumetric synthesis, the final layer consists of a

Conv3d layer with Tanh activation to produce volumes in the range $[-1, 1]$, which is more suitable for medical imaging applications.

Configuration Examples. InvUNet can be applied to both 2D image and 3D volume medical image synthesis. The following Table 4 demonstrates the architecture for both modalities.

Table 4. Architectural configurations of InvUNet for both 2D and 3D synthesis tasks, producing outputs at 224×224 resolution for 2D images and $16 \times 192 \times 192$ resolution for 3D volumes.

Layer	2D Config 224×224		3D Config 16×192×192	
	Spatial Resolution	Channels	Spatial Resolution	Channels
Bottleneck	14×14	256	1×12×12	256
<i>Memory Cache Branch</i>				
MC-1	28×28	128	2×24×24	128
MC-2	56×56	64	4×48×48	64
MC-3	112×112	32	8×96×96	32
MC-4	224×224	16	16×192×192	16
<i>Inversion Branch</i>				
Inv-1	28×28	128	2×24×24	128
Inv-2	56×56	64	4×48×48	64
Inv-3	112×112	32	8×96×96	32
Inv-4	224×224	16	16×192×192	16
Output	224×224	3	16×192×192	1

C. Additional Implementation Details

All experiments are implemented in PyTorch and conducted on a cluster node with 8 NVIDIA RTX 4090 GPUs.

Multi-modal Backbone and Pre-training. To accommodate diverse medical modalities, we employ a dimension-free architecture based on a standard ViT/B encoder. We use dimension-specific tokenizers to process inputs: Byte Pair Encoding for 1D clinical reports, and patch embeddings for 2D (X-rays, Pathology) and 3D (CT, MRI) visual data. For pre-training objectives, we follow modality-specific masking strategies. For text, we randomly mask 15% of words and optimize with Cross-Entropy loss following BERT. For 2D and 3D visual data, we adopt a high masking ratio of 75% and employ Mean Squared Error (MSE) loss on masked regions to reconstruct pixel values.

Modality-Specific Inversion Strategy. We apply our inversion-based replay strategy to all imaging modalities (X-ray, CT, MRI, and Pathological imaging) encountered during CSSL. For the Report modality, we adopt the clustering-based replay method following MedCoSS [54], as textual data is highly compressed and poses minimal privacy or storage risks compared to medical imaging data.

InvUNet Configurations. For image inversion, we employ the proposed InvUNet architecture with a channel configuration of [16, 32, 64, 128, 256] for the decoder path. The specific architectural variants for 2D patches (224×224) and 3D volumes ($16 \times 192 \times 192$) are detailed in Table 4.

Hyperparameters for Synthesis. The optimization of the generator involves precise hyperparameter tuning to balance diversity and fidelity. **Optimization.** We use the Adam optimizer with a learning rate of 2×10^{-4} for the generator parameters and a higher learning rate of 0.05 for the trainable latent vectors z (dimension 512). The loss balancing weights are set to $\alpha_{\text{norm}} = 1$, $\alpha_{\text{img}} = 0.1$, and $\alpha_{\text{rep}} = 0.1$.

2D Synthesis Strategy. For 2D modalities, we synthesize batches of 320 samples, optimizing each for 1,000 steps. To promote distribution diversity, we reinitialize the generator \mathcal{G} and 128-dim latents z after each batch generation.

3D Synthesis Strategy. Due to higher computational costs, we generate batches of 48 samples for 3D data with 512-dim latents z . Optimization runs for 500 steps per batch with an extended initial generation phase of 2,000 steps. We reinitialize the generator every 10 batches to introduce variability.

Repulsive Learning. For the repulsive term, we maintain a feature pool size matching the number of synthetic samples.

D. Visualization

D.1. Qualitative Segmentation Results

To intuitively demonstrate the effectiveness of InvCoSS in alleviating catastrophic forgetting, we visualize the segmentation predictions on five downstream tasks in Fig. 9. As observed, the sequential training baseline (SeqSSL) suffers from significant performance degradation, failing to delineate anatomical boundaries accurately. In contrast, InvCoSS produces precise segmentation masks that capture fine struc-

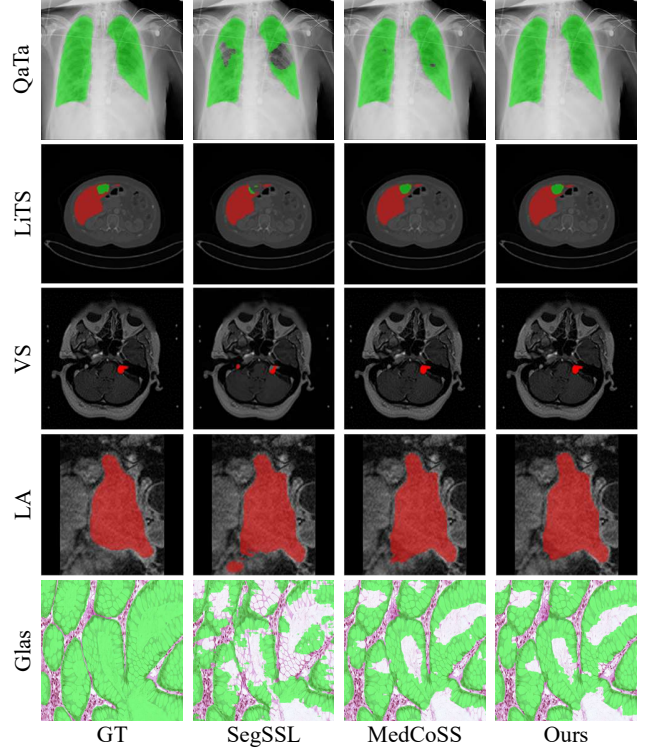


Figure 9. Visualization of segmentation results obtained by SeqSSL, MedCoSS, and Ours. Results for QaTa, LiTS, LA, and VS follow the default training order, while Glas uses the alternative order.

tural details, achieving qualitative performance highly comparable to the data-replay baseline MedCoSS. This consistency holds across different modality orders, verifying that our generated synthetic data successfully retains the necessary semantic knowledge for pixel-level dense prediction tasks.

D.2. Visual comparison of synthetic and real image

Fig. 10 provides a close-up comparison between real and synthetic X-ray images to illustrate the privacy-preserving properties of our method. As highlighted by the blue boxes, real medical images contain distinct high-frequency details, including clear rib textures, bone structures, subtle lesions, and medical instruments. In contrast, our synthetic images, while preserving the global anatomical layout (e.g., lung fields and spine) essential for representation learning, lack these fine-grained, patient-specific specificities. This visual evidence supports our analysis that InvCoSS inherently mitigates privacy risks by discarding sensitive high-frequency details while retaining the semantic information required for effective SSL pre-training.

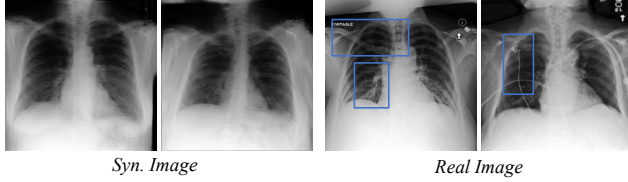


Figure 10. Visualization of real and synthetic images.

Table 5. Performance comparison of different pre-training data orderings. The default order is: Report, X-ray, CT, MRI, Path., and the alt order is: Path., MRI, Report, CT, X-ray.

Metrics		<i>Default Order</i>			<i>Alt Order</i>		
		SeqSSL	MedCoSS	InvCoSS	SeqSSL	MedCoSS	InvCoSS
Pub.	ACC	82.12	83.59	83.63	82.60	83.63	83.78
	AUC	94.91	95.38	95.24	94.83	95.27	95.32
	F1	76.37	77.87	77.81	76.52	77.79	77.83
Che.	ACC	92.65	94.31	94.09	95.57	95.70	95.50
	AUC	97.75	98.83	98.52	99.10	99.25	99.17
	F1	92.01	93.77	93.53	95.08	95.24	95.08
LiTS	DSC	66.57	72.01	72.14	70.60	71.43	71.22
	HD	49.42	36.50	30.36	42.27	36.58	38.75
VS	DSC	86.29	90.12	89.93	83.76	89.29	88.60
	HD	34.90	7.80	10.25	62.23	11.93	15.22
NCH	ACC	95.83	95.76	95.75	92.42	93.75	93.42
	AUC	99.57	99.51	99.61	99.16	99.30	99.23
	F1	94.37	94.01	93.90	89.87	91.19	90.61

E. Pre-training Order of Modality

To investigate the robustness of InvCoSS to different modality ordering, we evaluate performance under two pre-training sequences: the default order of Report, X-ray, CT, MRI, Path., and an alternative order of Path., MRI, Report, CT, X-ray. Table 5 presents results across five representative downstream tasks on SeqSSL, MedCoSS and InvCoSS. InvCoSS demonstrates consistent and robust performance across both orderings, maintaining competitive results with small performance variation.