Table 5. **Effects of fine-tuning longer.** We provide additional results of CNNs when fine-tuned not for 150 epochs but 1000 epochs.

PT Method	ATL	SBM	ISL	HNT	HAN	MSF	TPC	YBM	COS	ACD	AMO	KIT	ID Mean	OOD Mean	Mean
nnU-Net def. 1k	58.70	59.98	78.40	62.98	53.37	52.19	79.50	58.43	46.19	91.10	88.00	87.21	61.08	88.77	68.00
ResEnc-L (CNN)															
Scratch 1k	58.21	53.43	79.14	65.75	58.24	54.90	79.94	56.12	71.57	92.09	88.73	87.48	64.15	89.43	70.47
VoCo	57.52	58.80	78.08	61.50	57.43	54.76	76.81	58.05	61.49	91.74	88.21	87.89	62.72	89.28	69.36
SwinUNETR	56.64	56.80	77.24	60.83	55.63	53.87	76.17	57.73	62.16	91.28	88.22	87.82	61.90	89.11	68.70
SimCLR	57.79	58.85	78.81	63.81	58.13	55.08	78.75	59.75	62.45	92.11	88.58	88.24	63.71	89.64	70.20
VF	59.48	60.57	78.65	62.00	55.97	54.62	77.76	59.28	68.29	91.87	88.59	88.20	64.07	89.55	70.44
MG	57.65	57.85	77.44	62.00	57.73	53.97	77.79	58.18	61.27	91.63	88.59	87.86	62.66	89.36	69.33
MAE	61.04	62.35	78.71	64.65	59.48	54.73	78.20	59.47	71.87	92.02	88.86	88.13	65.61	89.67	71.63
S3D	59.84	59.84	78.52	62.76	58.31	54.37	78.43	60.24	72.24	91.93	88.78	88.10	64.95	89.61	71.11

Table 6. **Fewer data of higher quality provides similar performance.** Removing pre-training data to conduct MAE pre-training on the filtered datasets learns slightly more powerful representations than when using the full dataset. Moreover, reducing diversity of the training data by only training on T1w, T2w and FLAIR images, shows inferior performance.

			Dice Similarity Coefficient (DSC) [%] on ...														
			Dataset of same anatomical region (ID)									Dataset of OOD region			Average across ...		
PT Method	Data filter	Samples [N]	ATL	SBM	ISL	HNT	HAN	MSF	TPC	YBM	COS	ACD	AMO	KIT	ID	OOD	All
MAE	All	113.921	58.25	62.41	77.89	66.58	55.14	56.84	77.96	60.07	70.85	91.98	86.78	86.12	65.11	88.30	70.91
	IQS 3	91.952	59.33	63.86	77.96	66.64	52.89	56.52	77.17	59.88	73.37	92.11	86.61	85.66	65.29	88.13	71.00
	IQS 2.5	65.048	58.49	65.21	78.57	66.42	54.40	56.50	78.98	60.28	68.90	92.08	86.60	86.12	65.31	88.27	71.05
	IQS 1.5	38.225	58.71	64.69	78.24	66.73	51.54	56.24	79.00	59.12	69.28	92.01	86.60	86.20	64.84	88.27	70.70
	T1w,T2w,FLAIR	71.314	58.91	64.35	77.79	65.80	51.06	56.96	77.61	59.74	69.02	92.08	86.57	85.84	64.58	88.16	70.48

Table 7. **Effects of considering or ignoring anonymized regions on reconstruction based pre-training methods.**

		Dice Similarity Coefficient (DSC) [%] on ...														
		Dataset of same anatomical region (ID)									Dataset of OOD region			Average across ...		
PT Method	Anon. aware	ATL	SBM	ISL	HNT	HAN	MSF	TPC	YBM	COS	ACD	AMO	KIT	ID	OOD	All
MAE	No	58.25	62.41	77.89	66.58	55.14	56.84	77.96	60.07	70.85	91.98	86.78	86.12	65.11	88.30	70.91
	Yes	58.23	64.77	78.02	66.43	54.60	56.04	79.10	61.14	72.07	92.23	86.77	86.04	65.60	88.34	71.29
S3D	No	58.76	64.09	78.05	65.74	52.81	56.08	78.81	59.18	66.66	92.01	86.16	86.01	64.46	88.06	70.36
	Yes	59.01	62.84	77.92	66.07	53.47	56.01	77.59	58.87	69.45	92.04	86.28	85.80	64.58	88.04	70.45

A. Benchmark Results: Key Findings and Insights

Reconstruction based methods perform best for Segmentation: Our evaluation of SSL pre-training methods demonstrates that reconstruction-based approaches outperform other paradigms in 3D medical image segmentation both for ResEncL and for Primus-M. In particular, default MAE-based pre-training achieves the strongest results, consistently surpassing both a from-scratch nnU-Net baseline trained for 1000 epochs and the respective from-scratch trained architectures fine-tuned for 150 epochs (see Table 2). Notably, the MAE-pretrained ResEnc-L models not only outperform their from-scratch counterparts in 150 epochs but continue to improve with longer fine-tuning schedules (see Table 5).

Pre-training accelerates convergence: The most significant performance improvements from pre-training are observed within the first 150 epochs of fine-tuning. For instance, Primus-M improves by 4.8 DSC points and ResEnc-L by 2.47 DSC points within this short training duration. This surpasses the relative improvements compared to the full 1000-epoch training from scratch (see Table 2). This highlights the efficiency of pre-training in enabling faster convergence.

ResEnc-L remains the best overall model: Despite the benefits of pre-training across architectures, the ResEnc-L CNN continues to deliver the highest overall segmentation performance. After 150 epochs of fine-tuning, the MAE pre-trained ResEnc-L achieves a mean Dice score of 70.92, slightly outperforming the Primus-M transformer, which reaches 70.42 (see Table 2).

Pre-Training narrows the performance gap for Primus-M: While ResEnc-L remains the stronger architecture, pre-training provides a substantial boost to the transformer-based Primus-M, significantly reducing the performance gap. Primus-M benefits far more from pre-training than its CNN counterpart, highlighting the importance of self-supervised learning for

Table 8. **Primus-M Transformer fine-tuning schedule results.** Compared to the ResEnc-L fine-tuning, it can be observed that a normal *Warm-Up* schedule exceeds the *Sawtooth* fine-tuning schedule which proved best for CNNs. *: While no-warmup is the default for CNNs, the Primus-M architecture is trained per default with warm-up.

Dataset	PT Method / FT Schedule	VoCo	SwinUNETR	SimCLR	VF	MG	MAE	SimMIM	Mean
SBM	Default*	42.03	47.27	59.61	64.81	62.95	70.86	66.27	59.12
	Frozen	18.56	21.50	25.26	39.78	34.62	46.83	43.74	32.90
	Warm-Up	44.16	47.65	59.92	66.03	62.79	71.02	66.16	59.67
	Valley	37.55	41.64	54.96	65.74	59.11	67.42	67.67	56.30
	Sawtooth	38.19	43.20	54.11	64.12	61.27	66.88	68.10	56.55
ATL	Default*	48.23	48.49	56.42	63.39	58.15	61.48	59.80	56.57
	Frozen	25.88	28.99	39.90	32.91	47.18	57.57	49.59	40.29
	Warm-Up	48.53	47.72	56.60	63.39	59.43	61.39	60.29	56.76
	Valley	48.48	48.87	51.97	63.66	57.81	60.95	60.74	56.07
	Sawtooth	48.04	48.79	52.81	63.00	59.17	61.19	60.69	56.24
AMO	Default*	67.28	67.43	75.96	85.73	82.95	87.93	88.00	79.33
	Frozen	10.23	12.73	21.52	10.74	29.40	37.67	11.02	19.04
	Warm-Up	67.71	67.96	76.43	85.40	83.04	88.25	87.79	79.51
	Valley	66.02	66.51	75.88	84.81	82.91	87.80	88.39	78.90
	Sawtooth	66.01	66.72	76.28	85.03	83.12	87.84	88.22	79.03
KIT	Default*	68.31	68.00	77.47	82.61	82.00	86.34	84.41	78.45
	Frozen	22.02	24.14	33.67	23.87	33.10	40.82	27.40	29.29
	Warm-Up	71.52	69.08	80.58	82.14	80.00	86.45	83.78	79.08
	Valley	66.84	68.72	78.70	83.02	80.72	86.29	85.54	78.55
	Sawtooth	66.33	69.13	78.59	80.40	80.38	85.66	84.55	77.86
Average	Default*	56.46	57.80	67.37	74.14	71.51	76.65	74.62	68.36
	Frozen	19.17	21.84	30.09	26.83	36.07	45.72	32.94	30.38
	Warm-Up	57.98	58.10	68.38	74.24	71.32	76.78	74.50	68.76
	Valley	54.72	56.43	65.38	74.31	70.14	75.61	75.58	67.45
	Sawtooth	54.64	56.96	65.45	73.14	70.99	75.39	75.39	67.42

architectures that struggle to learn effective representations from scratch. Compared to 1000 epochs of from-scratch training, MAE pre-training followed by 150 epochs of fine-tuning improves Primus-M by 3.43 DSC points, whereas ResEnc-L sees only a 0.44-point gain (see Table 2). Given the previously observed performance disparity between CNNs and transformers in 3D medical image segmentation [35], these findings suggest that vision transformers, with appropriate pre-training, are increasingly capable of achieving competitive segmentation performance.

Contrastive pre-training offers limited improvements for full model fine-tuning but is best when keeping a frozen encoder for CNNs: Contrastive learning-based SSL methods—such as VoCo, SwinUNETR pre-training, and SimCLR—provide little to no benefit for segmentation when fine-tuning the entire model (see Table 4). These approaches either fail to surpass or only slightly improve upon the 150-epoch from-scratch baseline and, in some cases, they even degrade performance, particularly for the transformer-based Primus-M architecture. However, when fine-tuned with a frozen decoder, SimCLR pre-trained models outperform reconstruction-based methods for CNNs in segmentation (see table 4).

A frozen Transformer Encoder is worse than a frozen CNN Encoder: Keeping the encoder frozen during fine-tuning results in an even greater performance drop in performance for Transformers than for CNNs (see Table 2 and Tab. 4). While the ResEnc-L decoder allows for some adaptation, the Primus-M transformer has significantly fewer trainable parameters in its light-weight decoder stage. In this setting, fine-tuning is not far away from being linear probing, since only *TransposedConv-Norm-Act* layers are trained. This severely limits the decoder's model capacity, which makes adapting to new tasks more difficult. This underscores the necessity of full fine-tuning for Transformer architectures or the adaptation of Low-Rank adaptation fine-tuning strategies.

Longer fine-tuning improves OOD datasets: A comparison between the 150 epoch and 1000 epoch fine-tuning schedules reveals a trade-off between adaptation and preservation of pre-trained representations. Longer fine-tuning proves beneficial

for datasets where pre-training initially provided little advantage over 1000 epoch from-scratch training (see Table 5 and Table 2). This effect is particularly pronounced in out-of-distribution (OOD) datasets, such as KIT and AMO, which differ in modality, as well as datasets with a large number of target classes, like HAN and AMO, where convergence generally takes more time.

Longer fine-tuning can degrade performance where pre-training was already effective: In cases where pre-training already yielded strong improvements in shorter fine-tuning schedules, extending fine-tuning can lead to performance degradation. Notably, datasets such as SBM and HNT experience a decline when trained for 1000 epochs. These findings suggest that while extended fine-tuning can enhance generalization for OOD datasets, it may also override beneficial pre-trained representations in cases where pre-training was already highly effective.

Contrastive pre-training methods excel in classification: For ResEnc-L and Primus-M, contrastive pre-training consistently outperforms reconstruction-based approaches in classification tasks (see Table 3). While MAE pre-training still provides an improvement over training from scratch, it performs worse than all other methods. Meanwhile, other reconstruction-based methods like MG and S3D achieve moderate classification performance but still fall short of contrastive approaches. This finding reinforces that contrastive learning produces more generalizable global feature representations, making it particularly advantageous for classification tasks.

Balancing data quality and diversity is crucial for optimal performance: While the overall impact of filtering for high-quality images is limited, it does lead to slight performance improvements. Using only the top 57% of images with better quality results in a modest improvement of 0.15 DSC points (see Tab. 6). However, applying a stricter filter—retaining 62% of images by selecting only T1w, T2w, and FLAIR sequences—leads to a performance drop of 0.43 DSC points. This suggests that improving image quality can enhance results, but reducing dataset diversity negatively impacts performance. Notably, our quality filtering approach was relatively simple, leaving room for more refined selection strategies.

Impact of anonymization on reconstruction-based methods: Anonymization modifies image features, potentially impacting reconstruction-based pre-training methods that depend on pixel-level details. Processes like skull-stripping, defacing, or intensity masking may remove valuable anatomical structures, affecting representation learning. Incorporating masks during training to exclude these artificially altered regions from loss calculations slightly improves downstream performance (see Tab. 7). Unsurprisingly, this benefit is observed only for in-distribution datasets covering the head region, as these datasets are directly affected by anonymization processes that alter facial structures or skull regions. In contrast, for out-of-distribution datasets where such modifications are less relevant, anonymization has no noticeable effect.

B. OpenMind Dataset Details

While we provide broad information on the OpenMind dataset in the main manuscript, we extend this and go into more details in the following sections. First we provide more details on the Image Quality Score in Appendix B.1, provide details of the DWI Preprocessing in Appendix B.2 and lastly provide a broad overview of the dataset distribution in Appendix B.3.

B.1. Image Quality Score

The Image Quality Score is intended as a measurement of the suitability of data for pre-training to remove low-quality images from the pre-training dataset and hopefully increase downstream method performance. We want to denote that the idea of images being of higher or lower utility for pre-training is largely pre-training method-dependent. For example, images that exhibit large amounts of noise will be less suitable for reconstruction-based pre-training methods (e.g. MAEs), due to the presence of noise being impossible to predict for the model. Contrastive methods (e.g. SimCLR) on the other hand may be positively affected due to learning invariance of their representations to such noise. However, downstream datasets – which are application focused – are often composed of high-quality images with minimal noise to optimize outcomes, such as maximizing the accuracy of CyberKnife radiation therapy. Under these aspects, the Image Quality Score can be interpreted as a measure that quantifies the similarity of the pre-training dataset to the downstream datasets that represent real-world clinical applications.

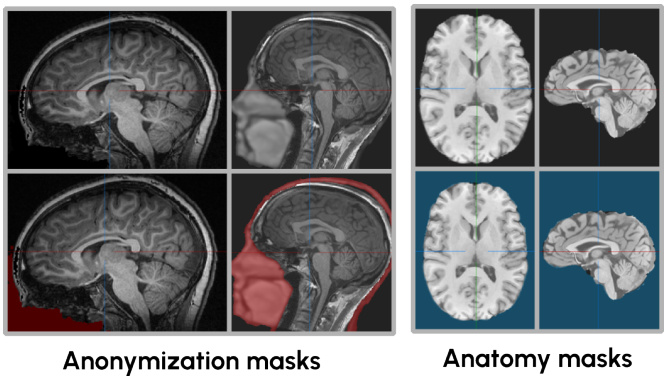

Figure 2. Head-and-Neck scans are often defaced, have the face blurred or have been brain-extracted to guarantee patient privacy. This can potentially harm reconstruction-based SSL methods. We provide anonymization and anatomy masks to allow taking this into account during method development.

Image Quality Score quantification To create the Image Quality Score (IQS), we inspected two images of each modality in a dataset for all 800 datasets composing the OpenMind dataset[4]. The resulting Image Quality Score of the two rated images was extended to all images of the same modality within the same dataset, as individual studies showed to be very consistent in their imaging protocols leading to very similar appearance of all images of the same modality within a dataset. Each of these two images was evaluated on a score from 1 to 5, with 1 indicating clear and sharp imagery and 5 indicating highly compromised image quality, based on their *noisiness* and *blurriness* induced e.g. by subject motion during image acquisition. Artifacts such as visible imaging equipment, the presence of signal voids (visible as shadows, covering a wide area), Gibbs artifacts or other contortions were categorized under *artifact_level*. The *artifact_level* was not included for all DWI-derived images, as the preprocessing steps already involved extensive removal of various artifacts and a brain extraction step, see Appendix B.2 for more details. Furthermore, images that were derived from the same DWI image receive the same Image Quality Score. Images where the field of view (FOV) captured only small regions of the brain or where the anatomy covered only a minor portion of the entire volume were also marked. Lastly, raters were able to tag images showing signs of corruption, such as those where missing data was apparent or entirely erroneous. These scores were determined by two independent raters and then aggregated to create the final Image Quality Score (IQS). Specifically, for each dataset and its modalities, the Image Quality Score (IQS) was calculated as follows:

i) **Aggregation of ratings from all raters**: For each image, values on a linear scale (*noisiness*, *blurriness*, and *artifact_level*) are averaged, while for categorical values, such as the state of corruption or the FOV, the worst possible value is selected to represent the image.

ii) **Calculation of IQS for each image**: First, the average of all numerical values is calculated. However, if the image is marked as corrupted or any numerical value equals 5, the IQS for that image is immediately set to 5. The IQS is increased by one point (capped at 5) if the image's FOV is suboptimal, as described above.

iii) **Modality-Wise IQS Aggregation per Dataset**: For each dataset and modality, each image is assigned the average IQS of these two randomly selected images.

B.2. DWI Preprocessing

Diffusion-weighted imaging (DWI) measures the diffusion of water molecules in tissue across 3D space. The direction of this diffusion process can be influenced by applying additional magnetic field gradients, allowing researchers to identify tissues that are more or less permeable to water diffusion in specific directions. Because the diffusion process is directionally dependent, multiple images are acquired using gradients of varying strengths to, for example, quantify anisotropies in diffusion behavior. As a result, DWIs are typically composed of a stack of 3D images, forming complex 4D images that are challenging to process and integrate. To manage the complexity of these 4D images, they are preprocessed into single 3D image formats that describe specific properties derived from the diffusion measurements in a more interpretable manner. These 3D derivatives of the DWIs allow easier integration into standard SSL pipelines. Specifically, we create T2-weighted, MD, and FA maps from the original 4D DWI stacks.

[4]The image quality assessment process was started with 5 images manually rated, but was reduced to just two images, due to high consistency in image attributes between the different images.

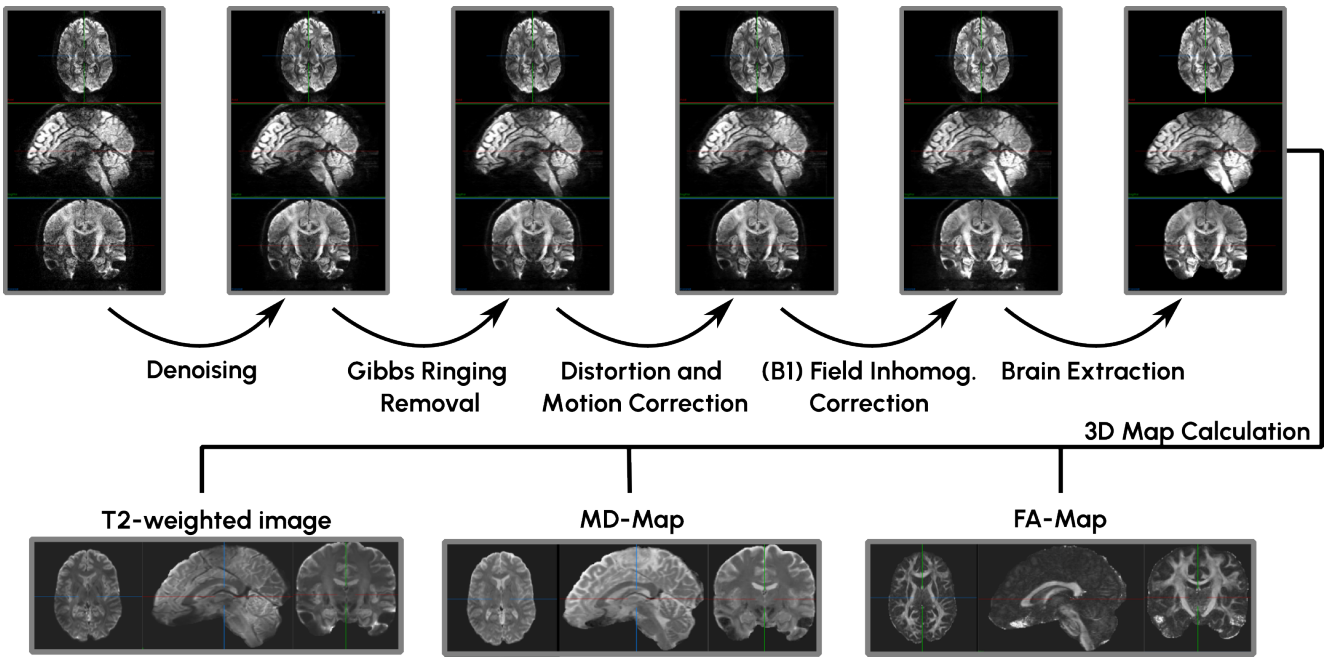

Figure 3. **DWI preprocessing pipeline:** To derive 3D images from the 4D DWI images, they are processed through six steps, denoising, ringing removal, co-registration, field correction, brain extraction, and lastly 3D derivative creation. Best viewed on a screen to see the differences between steps.

The overall preprocessing pipeline is composed of six steps, displayed in Fig. 3: i) Denoising of each 3D image. ii) Gibbs ringing artifact removal of each 3D image. iii) Co-registration of all images to one reference image to correct for potential distortions and motion artifacts. iv) Field correction of potential B1 field inhomogeneities. v) Brain extraction to remove skull and potentially existing face tissue that is not accounted for in the Map creation. vi) Actual creation of the 3D derivatives, namely a T2-weighted image, an MD-Map and an FA-Map.

B.3. Dataset overview

Dataset origin The final OpenMind dataset is comprised of 113,921 3D volumes of 34,191 subjects sourced from exactly 800 datasets. However, the dataset scale differs greatly, with 12 datasets contributing 50% of all data, 81 contributing 75% and 283 contributing 90% of all data, with the remaining 517 contributing the remaining 10%. While this indicates that many datasets do not contribute a huge quantity of data, they still introduce different niche modalities which may still be of value for the generalization of the pre-trained models. We visualize the cumulative dataset count over an increasing amount of datasets in Fig. 4.

Image Modalities The OpenMind dataset spans 24 Image Modalities – with all but the PET images being MR image variations. While we report these 24 Image Modalities, we want to emphasize that defining MR modalities or techniques is not trivial – e.g., T1-weighted MRIs can be acquired with various repetition and echo times, can be acquired by different scanners (changing image appearance drastically) but can still be considered a T1-weighted image. This complicates the assignment of images to specific categories, so the absolute number should be interpreted accordingly. To arrive at our provided number of 24 modalities, we used the provided BIDS modalities (which is not particularly consistent in naming between datasets) and grouped them into the final 24 categories detailed in Tab. 10. Overall, the majority of modalities are the common structural T1w, T2w and FLAIR images, as well as our diffusion-derived FA and MD-maps, representing about 86% of all data, with the remaining images being composed of various other techniques or modalities.
Overall, the OpenMind dataset consists of 40,720 unique imaging sessions, with a) 23,299 sessions with a single, b) 6,884 sessions with two images, c) 1,384 sessions with three images, d) 4,094 sessions with four images, e) 1,189 sessions with five images, f) 612 sessions with six images, g) 123 sessions with seven images, h) 115 sessions with eight images, i) 106 sessions with nine images, j) 287 sessions with ten images, k) 2,627 sessions with eleven or more images. We denote that the

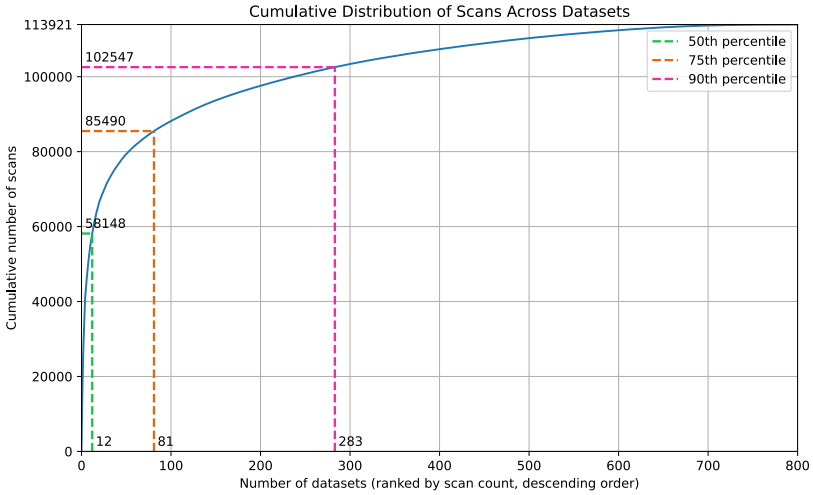

Figure 4. **Cumulative number of image volumes over number of datasets.** 50% of all images of the entire OpenMind dataset originate from 12 datasets, while 81 datasets contribute 75%, 283 contribute 90% and the remaining 517 contribute the remaining 10%.

Table 9. **Top 12 datasets by image count**. Together, these datasets represent approximately half of the total image volume (58,148 images).

Dataset ID	3D Volumes
ds004146 Strike et al. (2022)	16,016
ds003097 Snoek et al. (2021)	11,003
ds004884 Gibson et al. (2024)	6,945
ds004889 Rorden et al. (2024)	6,860
ds004215 Nugent et al. (2025)	3,013
ds004299 Gera et al. (2022)	2,668
ds004856 Park et al. (2024)	2,524
ds003604 Wang et al. (2022)	2,338
ds003416 Cai et al. (2021)	2,020
ds000221 Babayan et al. (2020)	1,805
ds003138 Koschutnig et al. (2023)	1,590
ds002843 Lee and Kable (2021)	1,366

Table 10. **Volume quantity by modality.** Final modality groupings as derived from the OpenNeuro BIDS modality indicators and as known from the own DWI derivatives for FA, MD and T2w maps.

T1w	T2w	FA	MD	FLAIR	MP2RAGE	sbref	ADC
42732	22999	13407	13407	5583	2859	1795	1757
TRACE	UNIT1	inplaneT2	SWI	UNIT1_denoised	minIP	PET	T1map
1715	927	892	784	724	687	653	557
inplaneT1	angio	PDw	T2starw	FLASH	T2map	T2starmap	DWI
529	488	473	409	307	106	76	55

same modality can exist multiple times within the same session, e.g., due to variations in the acquisition parameters.

Metadata categories and availability Moreover, we display the availability and distribution of metadata information like *age*, *sex*, *bmi*, *handedness*, *health status* and *race* in the OpenMind datasets in Fig. 5. It can be observed that the majority of datasets provide information about subject sex and age $> 70\%$, significantly fewer datasets provided information about the handedness (35%), health status (26.4%), bmi (15%) or race (11%). Even fewer studies recorded subject *weight* (1.5%). While we only conducted benchmarking of pure self-supervised learning methods that treat each image by itself, there exist

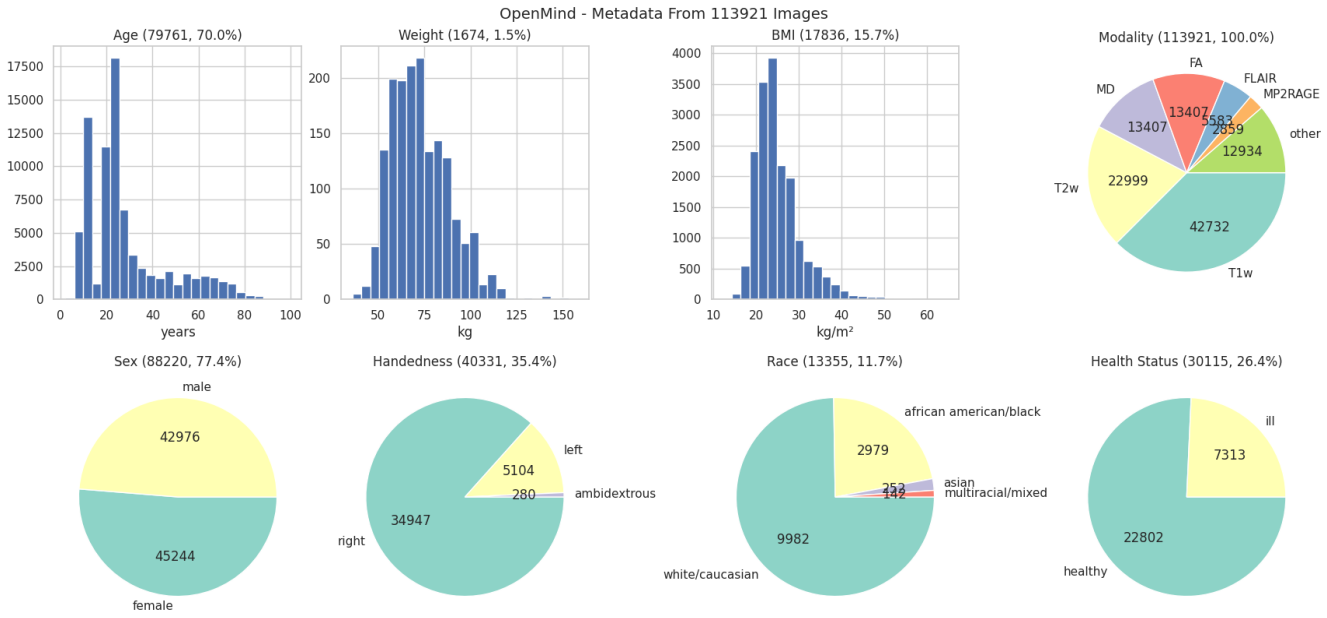

Figure 5. **Metadata of the OpenMind Dataset:** A total of 113,921 images from 800 datasets were curated and standardized, incorporating key metadata categories such as patient age, weight, BMI, sex, race, and health status, along with imaging modality. To enhance clarity, each pie chart and histogram in the figure only includes scans for which the respective metadata was available. The total number of available cases for each category is displayed above each graphic. Moreover, we denote that this number refers to images and not subjects of which there are only 34k. Therefore, this reflects the image-metadata pair availability instead of a per-patient score, which would not allow knowing the amount of scans with metadata.

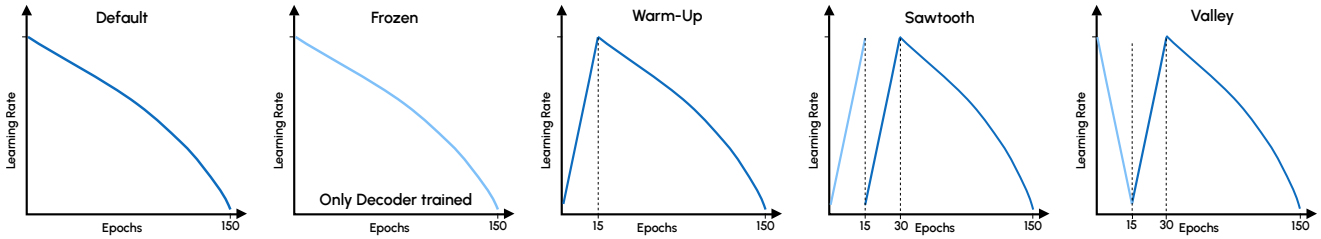

Figure 6. **Overview of fine-tuning schedules.** The plot illustrates the learning rate (lr) schedules for 150-epoch training runs. For 1000-epoch training, the overall structure remains the same, but each warm-up stage was extended from 15 to 50 epochs. The maximum learning rate was set to 1e-3 for ResEnc-L and 3e-5 for Primus-M. Light blue lines indicate stages where only the decoder was trained, while the encoder remained frozen.

pre-training methods that take meta-data into account to provide more efficient pre-training, which could be explored in future work.

C. Fine-tuning Details

C.1. Segmentation Fine-tuning

All datasets were preprocessed using a fixed cubic 1mm target spacing and z-score normalization mirroring the preprocessing of the pre-training dataset. However, for the pure CT dataset KIT, we applied nnU-Net's default CT normalization, which includes intensity clipping before z-score normalization. Additionally, for TPC, we used nnU-Net's default target spacing, as the 1mm cubic target spacing proved far too coarse for the dataset's thin target structures, which otherwise leads to a significant performance drop. For all nnU-Net default training runs from scratch, we applied the automatically planned, dataset-specific default target spacing and normalization. For fine-tuning, we set the initial learning rate to 1e-3 for ResEnc-L and 3e-5 for Primus-M, reducing the default learning rate of each model by one order of magnitude during the fine-tuning.

Below, we describe the different fine-tuning schedules explored in this work:
Default: This schedule follows nnU-Net's standard polynomial decay learning rate (lr) strategy, providing a gradual reduction in learning rate over time.
Frozen: The learning rate schedule remains the same as in the Default setting, but training is restricted to the decoder, keeping the encoder parameters fixed.
Warm-Up: Fine-tuning begins with a linear increase in the learning rate, allowing for a gradual adaptation before transitioning into the Default schedule.
Valley: Training starts by optimizing only the decoder with a linearly decreasing learning rate. This is followed by a warm-up phase, where the entire network is trained using a linearly increasing learning rate, before finally switching to the Default schedule.
Sawtooth: This schedule employs a two-stage warm-up. In the first phase, only the decoder is trained with a linearly increasing learning rate while keeping the encoder frozen. In the second phase, the entire network undergoes another warm-up with a linearly increasing learning rate before continuing with the Default schedule.
For training runs with 150 epochs, each initial warm-up stage lasts 15 epochs, while for 1000-epoch training, each stage is extended to 50 epochs. In all experiments, except for those reported in Table 4 and Tab. 8, we use the best-performing fine-tuning schedules: Sawtooth for ResEncL and Warm-Up for Primus-M.

C.2. Classification Fine-tuning

All classification fine-tuning was conducted using an adaptation of the *Image Classification framework*, which supports 3D classification. Preprocessing in this framework follows the standard nnU-Net preprocessing pipeline, with the important distinction that the entire volumes are resized to a smaller size, as classification is performed on entire images rather than sampled patches to guarantee the presence of potentially important visual cues.
For all datasets, models were trained for 200 dataset epochs (In our classification training, an epoch corresponds to a full dataset epoch.) using a learning rate of 1e-4 and a cosine annealing scheduler with a 20-epoch gradual warm-up applied to both the encoder and classification head. Optimization was performed using AdamW with a weight decay of 1e-2. Given the importance of large batch sizes for classification tasks, gradient accumulation over batches was employed to compensate for memory constraints. For the MRN dataset, the volumes were resized to $[32 \times 256 \times 256]$. The Primus-M architecture was trained with a batch size of 8 and gradient accumulation over 48 batches, while the ResEnc-L architecture used a batch size of 16 with gradient accumulation over 12 batches. On the RSN dataset, volumes were resized to $[160 \times 192 \times 192]$. The Primus-M model utilized a batch size of 2 with gradient accumulation over 192 batches, whereas ResEnc-L used a batch size of 4 with accumulation over 48 batches. For the ABI dataset, volumes were resized to $[160 \times 192 \times 224]$. Primus-M was trained with a batch size of 2 and gradient accumulation over 48 batches, while ResEnc-L employed a batch size of 4 with accumulation over 96 batches. Model performance was evaluated using 5-fold cross-validation, and we report Balanced Accuracy and Average Precision as metrics. Detailed configuration files for each dataset are available within the image classification framework repository.

D. Self-supervised learning method details

In this section, we provide a broad overview of the functionality of the different pre-training methods and provide details on hyperparameters of these methods. While some hyperparameters were method specific, we want to denote that we tried to keep hyperparameters as unified as possible to assure fairness. This includes that both ResEnc-L and Primus-M were pre-trained with a batch size of 8 and an input patch size of $[160 \times 160 \times 160]$ for the equal amount of 1000 epochs of 250 steps each. Moreover, the ResEnc-L pre-training learning rate was set to 1e-2, weight decay 3e-5, with an SGD optimizer with Nesterov following a PolyLR decay (see "default" in Fig. 6). The Primus-M architecture was pre-trained with an initial lr pre-training rate of 3e-4, weight decay 5e-2 with an AdamW optimizer following a "Warm-Up" schedule of Fig. 6 with 50 epochs of Warm-Up. Moreover, as different pre-training schedules adapt the architecture for the purpose of their method, we provide a visualization of these adaptations of the ResEnc-L UNet architecture and the Primus-M architecture in Fig. 7. It is to note that irrespective of the decoder being optimized during the pre-training, which would allow to transfer the pre-trained encoder into the segmentation downstream task, we chose to discard the decoder, as previous work showed this to be slightly superior [15]. For pre-training, we used a fixed target spacing of cubic 1mm and applied z-score normalization.

D.1. Volume Contrastive (VoCo)

Method description Wu et al. [22] proposed the Volume Contrast (VoCo) framework, which aims to capture contextual relationships in 3D medical images by contrasting different subvolumes within an image. Specifically, given an input volume,

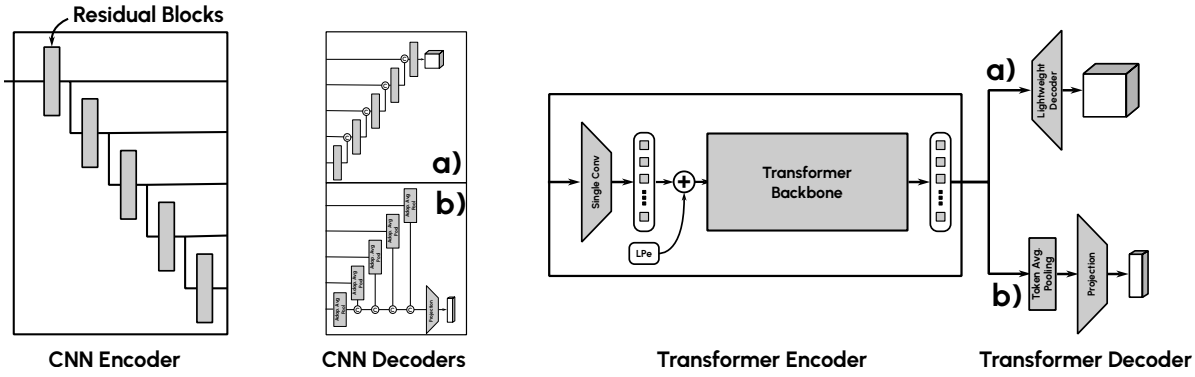

Figure 7. **Architecture configuration during pre-training.** We provide sketches of the different architectural settings the architectures are pre-trained in, depending on the needs of the respective method, as sometimes a dense and sometimes a global output is required. Since the skip-connections of the ResEnc-L architecture provide outputs on different resolutions, these were averaged through adaptive average pooling and concatenated before projection to the final latent (VoCo, SimCLR). For dense-outputs the highest-resolution output was used (MAE, S3D, VF, MG). The Primus-M architecture does not utilize multi-resolution streams, hence either an iterative up-projection was conducted as proposed in the original paper when requiring dense outputs (MAE, SimMIM, MG, VF), or a token average pooling followed by a linear projection layer was used (VoCo, SimCLR). For the SwinUNETR architecture, which requires dense and global outputs, we follow a) for generating dense outputs, except that no skip connections are used, as described in [23], and only a slice/channel for the rotation and contrastive prediction is leveraged, as implemented in their original repository, which we refer to for more details.

they first divide it into b non-overlapping base volumes arranged in a grid-like layout and extract them as individual base crops. In addition, n random crops are extracted from within the same volume. The backbone of the network processes all crops to extract the respective latent features. Following the SimCLR pre-training scheme, the latent features are then passed through a projection head to map them to a different embedding space. VoCo optimizes two objectives. The first involves computing similarity logits between a random crop's embedding p and the embeddings of all n base crops q_i using cosine similarity. These predicted similarity scores (ranging from 0 to 1) should align with the actual overlap proportions between the random crop and each base crop. The second objective introduces a regularization term that encourages the backbone to learn discriminative features between different base crops. This is achieved by calculating the cosine similarity s_{ij} between two different bases q_i and q_j and incorporating it into the loss function, thereby pushing s_{ij} towards a minimum of 0.

Hyperparameter choices The final pre-training settings in [22] utilize a grid of $[4 \times 4 \times 1]$, yielding $b = 16$ base volumes, each with a size of 96^3, in addition to $n = 4$ randomly sampled crops of the same size. This results in a total patch size of $[384 \times 384 \times 96]$. Afterwards, all crops are resized to 64^3. With our isometric spacing of $[1 \times 1 \times 1]$mm and brain MRIs as our pre-training dataset, a total patch size of $[384 \times 384 \times 96]$ proves impractical, given that human brains are far smaller than $38.4cm$ in width. Hence, we skip the downsampling step and directly crop volumes using the final input patch size of the network. Furthermore, we ensure that the grid does not exceed a size of $25.6cm$ in any direction. As the original hyperparameters proved to be not directly applicable we conducted additional tuning experiments on the validation set of the development datasets, ablating the choice of learning rate (*lr*), weight decay (*wd*), grid number (which indirectly affects the individual total patch size) as well as the number of randomly sampled target crops (*n*), see Tab. 11. We started the optimization process with $lr = 1e^{-2}$, $wd = 3e^{-5}$, $n = 4$ and a grid of $b = [4 \times 4 \times 1]$. We keep the crop size and input patch size at 64^3 throughout the entire process, leading to an initial total patch size of $[256 \times 256 \times 64]$. The final configuration proved to be the same as the initial configuration, with ultimately no changes in the hyperparameter settings.

D.2. Volume Fusion (VF)

Method description Volume Fusion (VF) is a pseudo-segmentation pretext task introduced by Wang et al. [16]. Given two separate input volumes I_f and I_b and a voxelwise fusion coefficient map α, a fused volume X is generated as follows: $X = \alpha \cdot I_f + (1-\alpha) \cdot I_b$. Each voxel's fusion coefficient α_i is drawn from a discrete set $V = \{0.0, 1/K, 2/K, ..., (K-1)/K, 1.0\}$, where K denotes the number of nonzero fusion coefficients. Each unique value in V is treated as a class, resulting in $C = K + 1$ segmentation classes. Since constructing a fused volume requires two input volumes, each mini-batch sample is formed by drawing a pair of images from the pre-training dataset. The corresponding fusion coefficient map is then generated by sequentially selecting patches of varying sizes at random and assigning each patch a fusion coefficient sampled from V.

Table 11. **Optimization of VoCo Hyperparameters.** We optimized Learning Rate, Weight Decay, Grid Size and Number of Random Crops, the most important hyperparameters of the VoCo method in this order. The optimal value for each fine-tuning step is highlighted in gray. *DnC: Did not Converge* during pre-training.

		Dice Similarity Coefficient (DSC) [%] for different hyperparameters											
		1. Learning Rate			2. Weight Decay			3. Grid Size			4. Num. Random Crops		
	Ablated value	1e-2	1e-3	1e-4	3e-4	3e-5	3e-6	3x3x1	4x4x1	4x4x2	2	4	8
	Other values	wd=3e-5; gr=4x4x1; n=4			lr=1e-2; gr=4x4x1; n=4			lr=1e-2; wd=3e-5; n=4			lr=1e-2; wd=3e-5; gr=4x4x1		
Dataset	Epochs												
SBM	150	71.52	70.85	71.35	*DnC*	71.52	65.90	69.50	71.52	72.57	69.71	71.52	72.30
	1000	70.20	73.33	72.98	*DnC*	70.20	67.17	72.00	70.20	70.69	71.45	70.20	71.34
ATL	150	60.42	59.31	60.08	*DnC*	60.42	56.74	60.76	60.42	58.11	59.62	60.42	60.35
	1000	59.62	59.59	58.32	*DnC*	59.62	57.61	58.90	59.62	58.70	58.23	59.62	59.26
AMO	150	86.78	85.52	82.88	*DnC*	86.78	84.16	86.04	86.78	86.27	86.00	86.78	86.69
	1000	88.86	88.65	88.57	*DnC*	88.86	88.17	88.82	88.86	89.03	88.92	88.86	88.90
KIT	150	85.12	82.01	80.14	*DnC*	85.12	80.33	83.93	85.12	83.44	83.93	85.12	83.14
	1000	86.72	84.76	86.40	*DnC*	86.72	84.60	84.93	86.72	84.56	85.79	86.72	85.26
Average DSC [%]	150	75.96	74.42	73.61	*DnC*	75.96	71.78	75.06	75.96	75.10	74.82	75.96	75.62
	1000	76.35	76.58	76.57	*DnC*	76.35	74.39	76.16	76.35	75.75	76.10	76.35	76.19
Average Rank	150	1.00	2.50	2.50	*DnC*	1.00	2.00	2.25	1.50	2.25	2.75	1.25	2.00
	1000	1.50	2.00	2.50	*DnC*	1.00	2.00	2.00	1.75	2.25	1.75	2.00	2.25

Table 12. **Optimization of Volume Fusion Hyperparameters.** We optimized Learning Rate, Weight Decay and Number of Nonzero Fusion Coefficients (K), the most important hyperparameters of the Volume Fusion method in this order. The optimal value for each fine-tuning step is highlighted in gray.

		Dice Similarity Coefficient (DSC) [%] for different hyperparameters								
		1. Learning Rate			2. Weight Decay			3. K		
	Ablated value	1e-2	1e-3	1e-4	3e-4	3e-5	3e-6	2	4	8
	Other values	wd=3e-5; K=4			lr=1e-3; K=4			lr=1e-3; wd=3e-5		
Dataset	Epochs									
SBM	150	73.85	73.78	75.10	76.24	73.78	74.45	74.91	73.78	75.91
	1000	72.43	75.10	72.90	74.09	75.10	72.86	73.33	75.10	75.32
ATL	150	64.99	65.07	61.58	63.61	65.07	63.84	63.64	65.07	63.71
	1000	62.10	62.80	60.72	62.55	62.80	62.62	61.82	62.80	62.69
AMO	150	86.67	86.46	85.20	86.79	86.46	86.08	85.81	86.46	86.34
	1000	89.10	88.95	88.70	89.50	88.95	89.03	89.13	88.95	88.97
KIT	150	84.56	85.41	82.97	83.10	85.41	84.45	83.28	85.41	84.95
	1000	86.21	86.15	85.29	85.65	86.15	86.76	85.25	86.15	85.00
Average DSC [%]	150	77.52	77.68	76.21	77.43	77.68	77.21	76.91	77.68	77.73
	1000	77.46	78.25	76.90	77.95	78.25	77.82	77.38	78.25	78.00
Average Rank	150	1.75	1.75	2.50	2.00	1.75	2.25	2.75	1.50	1.75
	1000	1.75	1.50	2.75	2.25	1.75	2.00	2.25	1.75	2.00

Hyperparameter choices We started the optimization process with $lr = 1e^{-2}$, $wd = 3e^{-5}$. Furthermore, different values of K were investigated, and an initial value of $K = 4$ was employed, following [16]. We used center cropping as the sampling strategy, as it promotes spatial consistency and ensures that corresponding anatomical regions align, while also avoiding volumes with large empty patches in our fused volume. All experiments were conducted on the validation set of the development datasets. In the final configuration, the learning rate was adjusted to $1e^{-3}$, while the other settings remained unchanged. Results are shown in Tab. 12.

D.3. Masked Autoencoders (MAE), S3D and SimMIM

Method description Masked autoencoders are a pre-training paradigm that mask a variable amount of the input image and learn to reconstruct the image based on the remaining, non-masked context of the image. Commonly, the pre-training objective only optimizes reconstructing the masked regions, in (normalized) pixel-space through an L2 loss.

MAEs for Transformers: MAE pre-training was popularized through [3] in conjunction with vision transformer architectures for the natural imaging domain (first evaluated in 3D through Chen et al. [18]), which leverage the sequence modeling

paradigm to improve computational efficiency by discarding the masked tokens effectively reducing the sequence length in the transformer. While this allows the encoder to function very effectively, the mask-tokens need to be re-introduced in order to reconstruct a full image, which is generally done by adding an additional transformer decoder stage, which is discarded when transferring to the downstream task.
SimMIM for Transformers: While Transformers are capable of discarding the masked tokens, this is not mandatory. In fact not discarding the tokens, but instead replacing them immediately with learnable Mask Tokens – introduced in 2D by Xie et al. [64] and adapted to 3D by Chen et al. [18]– leads to not needing an additional decoder, which can simplify training. This is generally less computationally efficient and a less common approach.
MAEs for CNNs: While keeping or discarding the masked regions in the Transformer is a choice, in CNNs it is mandatory to keep them as no efficient sparse-convolution methods exist to date. Subsequently, CNN MAEs suffer from the same inefficiencies that the SimMIM transformer methods does. Similar to the SimMIM case, the masked regions gradually shrink as the receptive field of the CNNs allows information to leak into the zeroed areas. Moreover, due to a large area in the input being artificially set to zeros, the Batchnorm statistics are influenced by the masking, which may lead to a shift in norm statistics when moving the pre-trained network to downstream tasks, making this approach less prevalent. Despite this, it is still a viable option for CNN pre-training. An exemplary use-case of such an approach in 3D can be seen in Munk et al. [17].
S3D for CNNs: Given the success of MAEs for Transformers, efforts were made to adapt the CNN architecture to resemble the MAE pre-training paradigm more closely by trying to maintain the masked regions for the entirety of the encoder and filling them at the beginning of the decoder with masked tokens. This also helps to overcome the BatchNorm statistic problems, see Tian et al. [65]. This methodology was adopted by Wald et al. [15], Tang et al. [24] for pre-training 3D medical images.

Hyperparameter choices **S3D for CNNs:** We used the publicly available repository provided by Wald et al. [15], re-using their final configuration due to them training their method on a dataset from a similar body-region. Notable hyperparameters included a pre-training patch size of $[160 \times 160 \times 160]$, learning rate of 1e-2, weight decay of 3e-5, batch size of 8[5], SGD optimizer with Nesterov and momentum 0.99, randomly sampled masking ratio between [60%-90%] with a L2 training loss that is only applied where inputs were masked. Due to the methodological restrictions of needing to mask in the bottleneck, the masked regions consisted of non-overlapping blocks of $[16 \times 16 \times 16]$ voxels that were projected up from the bottleneck.

MAE for CNNs: Our MAE implementation originates from the same repository as S3D [15] as the authors had an MAE implemented as an intermediate development stage. We included this implementation in our experiments as is – again due to the similarity of the dataset. Hyperparameters are identical to those of S3D with the exception of the MAE having a static masking ratio of 75% and a masking block size of $[16 \times 16 \times 16]$.
MAEs for Transformer: While existing MAE implementations for Transformers exist ([18]), they commonly use default 3D ViT's, which have shown to be far from ResEnc-L CNNs as shown in Isensee et al. [34] or Bassi et al. [51]. Instead, we use the recent Primus-M architecture, which provided an improved Transformer configuration. Consequently, we draw inspiration from their hyperparameters and used learning rate 3e-4, weight decay 5e-2, AdamW Optimizer with 1e-8 eps and betas (0.9, 0.98) and DropPath 0.2. We adopt randomly dropping tokens, which are of size $[8 \times 8 \times 8]$ for Primus with a masking ratio of 75%. Since the MAE scheme requires a decoder, we introduced a decoder of depth 2, with embedding depth (864) and number of heads (12) – identical to Primus-M. The masked regions were replaced by learnable embeddings of identical embedding dimensions.

SimMIM for Transformer: Due to the similarity between the MAE and the SimMIM method, we utilized the same hyperparameters but introduced the learnable mask tokens at the very beginning and introduced no additional decoder since this is not needed for SimMIM.

D.4. SimCLR

Method decription SimCLR (Simple Framework for Contrastive Learning of Visual Representations) [66] is a self-supervised learning approach that relies on contrastive learning to pre-train deep neural networks without labeled data. The core idea behind SimCLR is to maximize the similarity between differently augmented views of the same image while minimizing the similarity between views of different images. The training framework consists of the following steps:

[5] In the original repository this was set to 6, but as we developed our models on nodes with 4x40GB A100s we increased this to 8 to evenly distribute the images across all GPUs.

1. Data Augmentation: Each input image is transformed using a set of randomized data augmentations, such as random cropping, color distortion, Gaussian blur, and flipping. These augmentations produce two different views (positive pairs) of the same image.
2. Encoding: The ResEnc-L or Primus-M backbone processes each augmented image, mapping it to a latent representation.
3. Projection: The latents are passed through a linear projection head that maps the representations to a lower-dimensional space where the contrastive loss is applied.
4. Contrastive Loss: The loss function used in SimCLR is the normalized temperature-scaled cross-entropy loss (NT-Xent). This loss encourages the positive pairs (two augmentations of the same image) to be close in representation space while pushing away representations of different images (negative pairs).

By minimizing this training objective, the model learns to generate useful feature representations by learning to become invariant to the augmentations, learning to focus on the semantic meaning, while learning that different categories should be different. After training, the encoder is fine-tuned, and the projection head is discarded.

Hyperparameter choices Contrastive pre-training methods generally require larger batch sizes as they need to learn what to be similar and dissimilar to. We train with a batch size of 32, receiving input crops of $[192 \times 192 \times 64]$ of which we create two augmented sub-crops of dimensions $[64 \times 64 \times 64]$ per input crop with a minimal crop-overlap of 50%. Moreover, the NTXent loss uses a temperature scale of 0.5 as well as a cosine similarity function between the latents. To create the augmented versions, we employ *GaussianNoise*, *GaussianBlur*, *BrightnessMultiplicative*, *ContrastAugmentation*, *SimulateLowResolution*, *Gamma*, *Mirror* and *Rotate90Deg* transforms of the common batchgenerators framework[6]. We refer to the repository for explicit parametrization and potential repetitions of the same augmentations.

D.5. SwinUNETR

Method description SwinUNETR was proposed as a pre-training method for the identically named SwinUNETR architecture and is composed of three components: Image inpainting, Rotation prediction as well as a contrastive training objective. The inpainting itself is a simple L1 loss applied for a masked out image region, the rotation is a rotation of 0°, 90°, 180° or 270° degrees along the z-axis with an MLP used to classify the applied rotation. Lastly, the contrastive coding enforces the linearly projected representations of the encoder to be highly similar or dissimilar if the two sub-volumes belong to the same or a different image, respectively. These three losses are combined with an equal weighting to form the SwinUNETR pre-training.

Hyperparameter choices Adhering to the original hyperparameters we calculate the in-painting loss with an L1 loss function, the rotation classification through a cross entropy loss and the contrastive loss and aggregate the final loss through aggregating the losses with weights $\lambda_{rot} = \lambda_{inpaint} = \lambda_{contrast} = 1$. For the rotation objective, we follow the description in the paper and rotate along the z-axis. For the contrastive objective, we followed the original implementation, which uses random rotations and random cut-outs with a maximum removal rate of 60% (In the paper, they mention a drop rate of 30% but override this in the implementation). As no other model-specific hyperparameters had to be chosen, we trained the model with a batch size of 8 and an input image shape of $[160 \times 160 \times 160]$, which was internally doubled by SwinUNETR through the contrastive loss objective's augmentations, making it train substantially longer than the other methods.

[6] https://github.com/MIC-DKFZ/batchgenerators