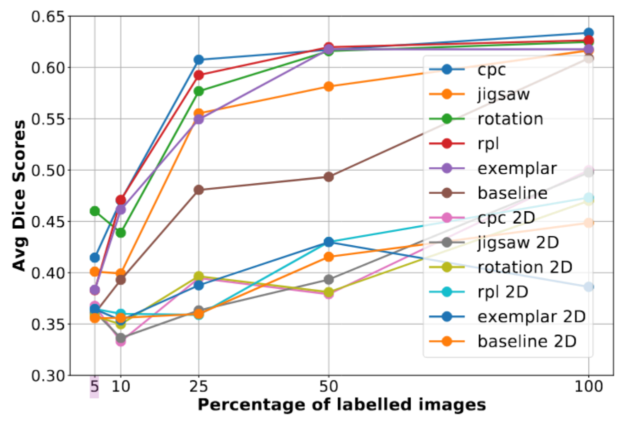
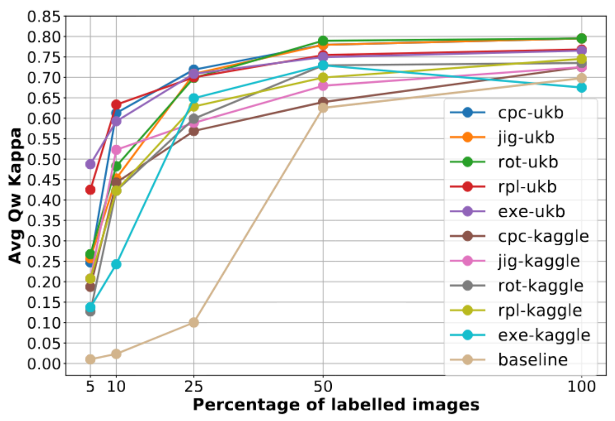
Introduction

Medical imaging models often struggle when labeled data is scarce. Self-supervised learning (SSL) addresses this by creating proxy tasks on unlabeled data to learn useful features. In “3D Self-Supervised Methods for Medical Imaging” [1], Taleb et al. extend five SSL tasks into the 3D domain, pretraining 3D UNet encoders on large volumes. They demonstrate that these 3D pretext tasks substantially improve downstream segmentation and classification accuracy, especially in small data regimes. Their key contributions are: 1) Formulation of five 3D SSL tasks, 2) Open-source implementations of SSL tasks, and 3) Comprehensive evaluations on pancreas tumor segmentation, brain tumor segmentation, and diabetic retinopathy classification, empirically demonstrating efficiency gains produced by SSL.

Chosen Result

We recreated the pancreas segmentation data-efficiency curves (left, from Figure 3 of Taleb *et al.*) on a 3D UNet [2] for four of the five SSL tasks (i.e. jigsaw, rotation, rpl, and exemplar). We pretrained the model on unlabeled CT volumes and finetuned on the same subsampled sets of labeled scans from the Medical Segmentation Decathlon dataset [3], assessing downstream performance with Dice score. We also reproduced the diabetic retinopathy data-efficiency curves (right, from Figure 4 of Taleb *et al.*) on a DenseNet-121 backbone [4]. We pretrained the model on 2D UK Biobank fundus images [5], and we measured downstream classification performance with Quadratic Weighted Kappa. The finetuning procedure for both tasks involves training on 5%, 10%, 25%, 50%, and 100% of the training data and measuring downstream performance. This result is central to the paper’s main claim that self-supervised pretraining markedly boosts performance in low-annotation regimes, providing a clear, quantitative benchmark for comparing the SSL task effectiveness. By faithfully reproducing both results, we confirm that self-supervised pretraining substantially accelerates convergence and improves accuracy in low-annotation regimes, and validates our PyTorch implementations.





Methodology

After self-supervised pretraining, we adapt each backbone to its downstream task:

3D Segmentation (3D UNet): We attach the pretrained 3D UNet encoder to a decoder mirroring its down-sampling path. In the first 25 epochs, only the decoder weights are updated (encoder frozen) using a weighted Dice loss; thereafter, we fine-tune the entire network. We evaluate segmentation quality via Dice score on proportionally subsampled CT scans.

2D Classification (DenseNet‑121): We replace the final classification layer with a new head matching the number of diabetic retinopathy grades. During finetuning on APTOS 2019, we freeze the DenseNet feature extractor for the first 5 epochs, training only the new head with cross‑entropy, then unfreeze all layers for joint training. We measure validation performance using Quadratic Weighted Kappa.
To enable our models to learn from unlabeled medical data, we implemented four self‑supervised "proxy" tasks. Each task defines a simple classification or embedding objective on transformed inputs. By solving it, the encoder must discover meaningful anatomical features that later transfer to downstream segmentation or classification. Below is an overview of each 3D task and how it was realized in our codebase. Note that we implement analogous tasks for the 2D tests as well.

3D Rotation Prediction: In this task, each input volume is randomly rotated by the identity rotation or one of three distinct 90° increments around the x, y, or z axis. The rotated scan is passed through the 3D UNet encoder and a small classification head, which is trained with a 10‑way cross‑entropy loss to predict the rotation applied. By forcing the model to identify orientation changes, it must internalize organ shapes and spatial landmarks, which directly benefits downstream semantic segmentation.

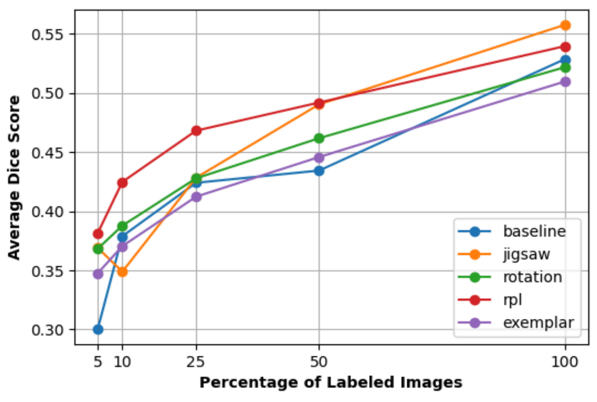
3D Jigsaw Puzzle: Here, we partition each volume into a $3 \times 3 \times 3$ grid of cubic patches and then shuffle them according to one of 100 carefully chosen permutations. The shuffled patches are reassembled as separate input channels to the encoder, and a classification head is trained with a 100‑way cross‑entropy loss to recognize which permutation was used.

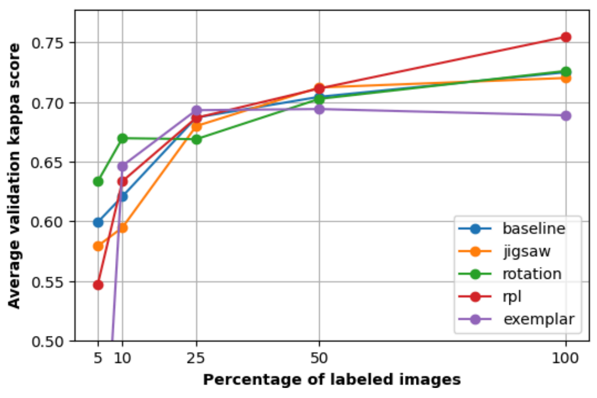
3D Relative Patch Location (RPL): For RPL, we first crop a jittered $3 \times 3 \times 3$ grid of patches from each volume. We then select the central patch as a reference and a second "query" patch from one of the 26 remaining grid positions. By stacking these two patches into a two‑channel input, we task the encoder and a classification head with a 26‑way cross‑entropy loss to predict the query patch's relative grid index.

3D Exemplar Networks: In the exemplar task, we generate two different transformed views of the same volume (using random flipping, rotation, brightness/contrast adjustments, and zoom transformations) and also sample a third volume as a negative example. All three are passed through the shared encoder to produce embeddings. We then apply a triplet loss that encourages the anchor and its transformed "positive" embedding to lie close together in feature space, while pushing the "negative" embedding farther away.

Results & Analysis

Our data efficiency graphs are included below (left is pancreas and right is fundus), where we observe similar trends to the authors. Specifically, all of the pretext tasks, with the exception of exemplar, improve downstream task efficiency.





We see two main discrepancies. First, for the segmentation performance, our dice scores are 5-10 points lower than the authors. We believe this was due to our hardware constraints, which forced us to use a smaller model size than the authors. Since we could not fit the original model in VRAM, we had to half the number of filters in each layer, significantly reducing our model capacity. The second discrepancy we observed was that our exemplar model trended below the baseline (i.e. reduced data efficiency). We attribute this to our naive batch-level negative sampling strategy, which likely did not produce enough hard negatives to learn a good data representation. The authors described multiple negative sampling strategies, but it was unclear from the paper which they used. This was consistent with other pretext tasks, where some subtle details were often underspecified and forced us to take our best guess at how to implement them.

These reproduced results are significant because they support the claim that self-supervised methods improve training efficiency, making them valuable in the medical imaging domain. Additionally, these results indicate that 2D pretext tasks can be generalized to 3D, demonstrating similar efficiency improvements.

Reflection

Our reimplementation of Taleb et al.'s 3D self-supervised methods proved both challenging and enlightening. The most striking lesson was how dramatically pretraining on unlabeled 3D data improved downstream performance when labeled data was scarce, confirming the paper's central thesis that self-supervision can address the annotation bottleneck in medical imaging. We encountered unexpected implementation hurdles with memory constraints when scaling to larger 3D volumes, but despite this, our results largely followed the original paper's. The worse exemplar performance compared to the paper suggests that our negative sampling approach may need to be optimized. In future experiments, we would like to explore more efficient 3D architectures and test them on multimodal datasets as a potential extension of the project.

References

[1] A. Taleb et al., '3D self-supervised methods for medical imaging', in Proceedings of the 34th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 2020.

[2] O. Ronneberger, P. Fischer, and T. Brox, 'U-Net: Convolutional Networks for Biomedical Image Segmentation', in Medical Image Computing and Computer-Assisted Intervention -- MICCAI 2015, 2015, pp. 234–241.

[3] A. Simpson et al., 'A large annotated medical image dataset for the development and evaluation of segmentation algorithms', CoRR, vol. abs/1902.09063, 2019.

[3] M. Karthik and S. Dane, 'APTOS 2019 Blindness Detection', Kaggle, 2019.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, 'Densely Connected Convolutional Networks', in 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269.