

Pruning of MedSAM's ViT-B Encoder Across Medical Imaging Modalities

Compression-performance scaling study

In partial fulfilment of the requirements for the degree of
Master of Science in Biomedical Engineering – Medical Physics Track
at Delft University of Technology

by

Louis-Franz Alberts

<i>Supervisor:</i>	Qian Tao
<i>Daily Supervisor:</i>	Yidong Zhao
<i>Department:</i>	Imaging Physics (ImPhys)
<i>Date:</i>	1 September 2026

ABSTRACT

Unlike modality or structure-specific segmentation models such as U-Net, foundation models like Medical Segment Anything Model (**MedSAM**) segment across imaging modalities and anatomical structures without retraining, at a fixed cost. **MedSAM** pairs a Vision Transformer Base Model (**ViT-B**) image encoder with a prompt encoder and a mask decoder. The encoder dominates, accounting for 95.66% of the parameters and, at 966 giga Floating Point Operations (**FLOPs**), for almost all of the computation of a single forward pass, incurred for every image regardless of the modality segmented. This work characterises that cost analytically, deriving a block's **FLOPs** from its retained attention heads and Multi-Layer Perceptron (**MLP**) neurons rather than by profiling, and then asks how much of the encoder is redundant for medical image segmentation.

Redundancy is studied at two levels. At the level of the individual weight, redundant weights are zeroed while the weight-matrix shapes are left unchanged, so no computation is saved. At the level of the whole computational structure, removing an attention head or an **MLP** neuron physically reduces the weight count. It is the only level at which inference is accelerated. Sparsity is induced during training by a penalty on the loss: the Lasso ℓ_1 penalty drives individual weights towards zero, paired with unstructured pruning, whilst Sparse Group Lasso (**SGL**) groups the weights of each head and neuron and drives whole groups towards zero, paired with structured pruning. As both penalties are non-smooth at zero, they are imposed via Proximal Gradient Descent (**PGD**), in which the gradient step optimises the task loss and a closed-form shrinkage sets small weights to exactly zero. Both pruned regimes are compared against a dense task-loss-fine-tuned reference and a variant with the encoder removed entirely, the latter serving as a floor that tests whether the encoder is needed at all for a given target.

Evaluated across seven imaging modalities and eleven evaluation subsets, the encoder is found to hold substantial but unequal redundancy. At the individual-weight level, 72.49% of the prunable encoder weights can be zeroed while maintaining accuracy comparable to the baseline, but the irregular sparsity yields no computational savings. As whole structures, 91.7% of the prunable encoder weight volume can be removed, reducing the model from 93.74 to 15.82 million parameters and the encoder cost from 966.32 to 81.08 giga **FLOPs**, which does come at a cost of the Dice Similarity Coefficient (**DSC**).

This cost is not uniform across structures. Some targets retain usable accuracy with the encoder fully removed, such as the ultrasound foetal head and the fundus optic disc, both above 92% **DSC**, whilst others depend heavily on it, most notably the Chest X-ray (**CXR**) lung fields, whose median **DSC** falls to 54.84%. The per-structure cost ranges from the optic cup, which improves by 3.08 points under compression, to GlaS (testB), which falls by 8.60. The **MedSAM** encoder therefore carries redundancy that can be removed for real savings in size and speed. However, the extent of removal depends on the structure and modality being segmented.

CONTENTS

Abstract	i
1 Introduction	1
1.1 Medical Image Segmentation and Deep Learning Solutions	1
1.2 Computational Cost and Efficient Segmentation	2
1.3 Research Questions	4
2 Background	5
2.1 SAM and MedSAM	5
2.2 The ViT-B encoder	5
2.2.1 From patch to token	6
2.2.2 The residual stream	7
2.2.3 The multi-head attention unit	7
2.2.4 The MLP unit	9
2.2.5 Global and windowed attention	9
2.3 Network pruning	9
2.3.1 Unstructured and structured pruning	9
2.3.2 The pruning criterion	10
2.4 Sparse optimisation and proximal methods	10
2.4.1 The Lasso and Sparse Group Lasso penalties	10
2.4.2 Proximal gradient descent	11
3 Methods	12
3.1 Promoting Sparsity of the MedSAM Encoder	12
3.1.1 Model and Prunable Parameters	12
3.1.2 Individual Weight Sparsity by Lasso	13
3.1.3 Structural Pruning by Sparse Group Lasso	14
3.1.4 Proximal Gradient Descent	15
3.2 Analytical Derivation of Computational Cost	16
3.2.1 Floating Point Operations	16
3.2.2 Inference Time	17
3.3 Evaluation Framework	17
3.3.1 Segmentation Accuracy	17
3.3.2 Accuracy and Cost Trade-off	17
3.4 Implementation Details	17
3.4.1 Datasets	17
3.4.2 Preprocessing	18
3.4.3 Finetuning Protocol	21
3.4.4 Pruning Sweeps	21
3.4.5 Hardware and Software	22

4	Results	23
4.1	Task-loss finetuning	23
4.1.1	Zero-shot vs Task-loss finetuned	23
4.2	Lasso regularisation	25
4.2.1	Training dynamics	25
4.2.2	Weight distribution	26
4.2.3	Unstructured pruning	28
4.3	SGL Regularisation	29
4.3.1	Training dynamics	29
4.3.2	Weight Distribution	29
4.3.3	Structured Pruning	29
4.4	Regime Comparison	32
4.4.1	Encoder-removed vs Lasso vs SGL vs Task-loss finetuned	32
4.4.2	Qualitative behaviour of the predictions	36
5	Discussion	38
5.1	Mechanism of the proximal operators	38
5.1.1	Element-wise and group removal	38
5.1.2	Interaction of the penalty strengths	39
5.2	The extent of redundancy	40
5.3	Dependence on imaging modality	41
5.3.1	Qualitative behaviour of the predictions	42
5.4	Limitations	42
6	Conclusion	44
A	Appendix	46

ABBREVIATIONS

ViT	Vision Transformer
SAM	Segment Anything Model
MedSAM	Medical Segment Anything Model
DSC	Dice Similarity Coefficient
CT	Computed Tomography
MRI	Magnetic Resonance Imaging
MLP	Multi-Layer Perceptron
CXR	Chest X-ray
FLOPs	Floating Point Operations
ViT-B	Vision Transformer Base Model
LV	Left Ventricle
RV	Right Ventricle
MYO	Myocardium
PET	Positron Emission Tomography
RGB	Red Green Blue
MHA	Multi Head Attention
GELU	Gaussian Error Linear Unit
L2	Euclidean Norm
SGL	Sparse Group Lasso
IQR	Inter Quartile Range
PGD	Proximal Gradient Descent
BB	Bounding Box
GT	Ground Truth
SGD	Stochastic Gradient Descent

1

INTRODUCTION

1.1. MEDICAL IMAGE SEGMENTATION AND DEEP LEARNING SOLUTIONS

Medical image segmentation is the practice of partitioning an image into anatomical structures or pathological regions. It forms the basis for diagnosis, treatment planning, and the quantitative analysis of disease across a wide range of clinical tasks [1]. Accurate segmentation allows a structure to be measured, localised, and tracked over time, so the quality of the segmentation directly affects the clinical decision it supports [2]. Delineation was traditionally performed by hand through manual tracing by a trained expert, a time-consuming process that varies across observers and limits its use in routine practice [3]. Automated deep learning methods have been proposed to reduce this burden and have led to significant speed-ups in the segmentation process. A deep learning pipeline for cardiac segmentation, for example, has been reported to cut the delineation time for the right ventricle by 87% relative to fully manual analysis, whilst producing delineations judged fit for clinical use [3]. Automation of this kind speeds up the clinical workflow and improves the consistency of the result.

The most widely used deep learning model for this task is the U-Net [4], which established the encoder-decoder design that dominates the field, pairing a contracting path that captures context with an expanding path that recovers spatial detail [5]. Whilst the U-Net is clinically useful, its training design fits a separate model to each task, modality, structure, and dataset. This specialisation is also a limitation. Having seen data for only a specific use case, such a model struggles to generalise to unseen anatomies or imaging modalities, with segmentation performance degrading significantly when it is applied to data from a different distribution. Adapting it requires labelled data from the new domain and a further training run, which is costly and often unattainable when such data are not readily available [6].

Recent studies have shifted toward developing *foundation models* for medical image analysis [7], which are trained once on a large, diverse dataset and then adapted to many downstream tasks without task-specific retraining [8]. Rather than fitting a separate net-

work to each modality, structure, and dataset, a single model acquires enough general representation during pretraining to transfer across tasks, either directly or with only light adaptation. In medical image segmentation, this shift is realised by Medical Segment Anything Model (**MedSAM**), a foundation model that segments anatomical structures across imaging modalities from a single prompt, rather than a specialist model trained per task [9].

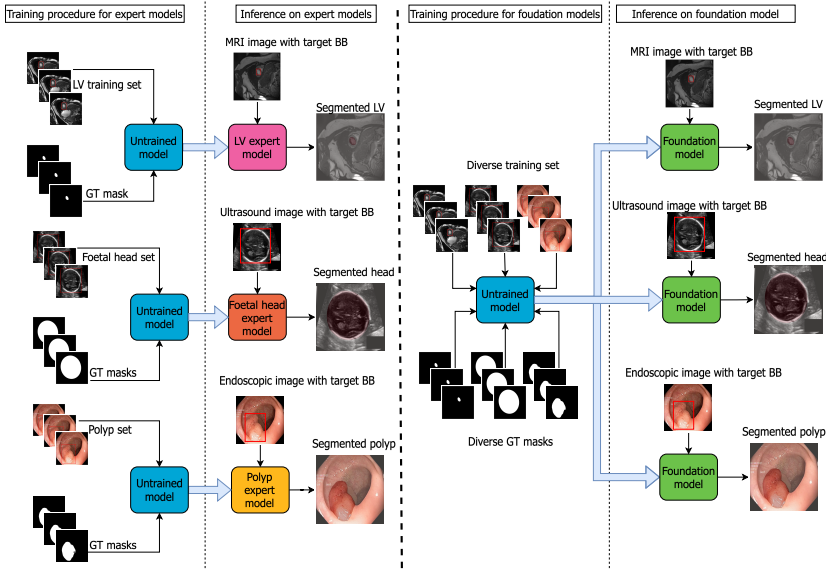


Figure 1.1: Left: Overview of training and inference for task specific (expert) models. Right: Overview of training and inference for foundation models. Bounding Box (BB), Ground Truth (GT), Left Ventricle (LV)

1.2. COMPUTATIONAL COST AND EFFICIENT SEGMENTATION

This generality comes at a computational cost. Foundation models derive their transferability from scale, both in the breadth of their pretraining data and in the size of the networks that absorb it, and this size is carried through to every inference. For **MedSAM**, the cost is concentrated in the image encoder and incurred in full for every image, regardless of the modality or structure being segmented. It also falls hardest where the need is greatest, as the regions with the least access to medical expertise are often those with the least access to the computational resources such a model requires [10]. For deployment in clinical settings, where computing and memory are limited, this fixed cost is the principal obstacle to use. Reducing it, without sacrificing the generality that makes the model useful, is the problem this thesis addresses.

The cost of large models is a general problem, and a range of compression methods has been developed to address it. These are commonly grouped into four families: pruning, which removes parameters or whole structures; quantisation, which lowers the numerical precision of the weights; knowledge distillation, which trains a smaller model to imitate a larger one; and low-rank factorisation, which decomposes weight matrices into smaller ones [11]. This work takes the pruning route, which is among the oldest and most direct of the four and rests on a long-standing finding that a large fraction of a trained network can be removed with little loss of accuracy [12].

Pruning divides by what it removes, and the literature has moved from the individual weight to the whole structure. Early magnitude pruning removes individual weights, setting to zero those with the smallest absolute values [12]. This unstructured approach can yield high sparsity but no direct speed-up, as the tensor keeps its dense rectangular shape and standard hardware still stores and multiplies through the zeros; realising a saving requires specialised sparse-matrix hardware or kernels [13]. It reveals how many parameters a task does not need, not how much computation can be saved. Later work, therefore, turned to structured pruning, which removes whole units, such as channels, filters, or, in transformers, attention heads, so that the weights themselves are rebuilt at a smaller size, and the model genuinely runs faster on ordinary hardware [14], [15]. The two regimes are intended to measure different things: the redundancy present in principle, and the redundancy that can be realised as a saving in practice. Most studies commit to one, chosen for the goal at hand. Studying both on the same model, as a way of separating the two, is less common and is the approach taken here.

The choice of regime determines what a pruning method removes, but not which weights or units it removes; for this, a criterion is needed. Criteria of increasing sophistication have been proposed. The simplest is magnitude, judging a weight or group by its absolute value alone and removing the smallest [12]. More elaborate criteria estimate the effect of removal on the loss directly, approximating it through a first- or second-order Taylor expansion and removing the structures whose deletion is predicted to matter least [16]; Fisher-information scores, including the boundary-aware estimate used by MedCore, belong to this family [17]. A different approach does not score a fixed model at all but instead deliberately produces small-magnitude weights during training, adding a sparsity-inducing penalty to the objective so that unimportant weights and groups are driven towards zero and can then be removed at little cost [14].

These approaches have since been applied to segmentation foundation models, though efforts to make Segment Anything Model (SAM) efficient have largely bypassed pruning in favour of replacing the encoder outright. MobileSAM distils the heavy Vision Transformer (ViT)-H encoder into a compact TinyViT encoder, reducing the encoder parameters by roughly two orders of magnitude while retaining the original prompt encoder and mask decoder [18]. EfficientSAM instead uses masked image pretraining to teach a lightweight encoder to reconstruct the original features, then finetunes it for segmentation [19]. Both instances replace the encoder with a newer, smaller network, so its cost is fixed at design time and cannot be swept to trade accuracy for size along a controllable axis. Pruning offers exactly this control, as the extent of removal can be varied. However, it has been applied to SAM only rarely, and where it has, in combination with other techniques rather than in isolation [20]. None of these efforts addresses the medical domain,

where **MedSAM** is the model of interest.

The one study to prune **MedSAM** directly is MedCore, a structured pruning framework that removes attention heads and Multi-Layer Perceptron (**MLP**) units [17]. It frames the task as extracting a medical core: the structures worth keeping are those that became important during the adaptation of **SAM** to **MedSAM**, identified by comparing the removal of a structure with its reset to the original **SAM** weight, and those that carry the segmentation boundary, identified by a boundary-aware Fisher estimate [17]. A central argument of the work is that region-overlap metrics such as the Dice Similarity Coefficient (**DSC**) can remain high. At the same time, boundary fidelity is lost, so it reports boundary metrics alongside the **DSC** and shows that attention-head pruning damages the boundary more than **MLP** pruning [17]. MedCore establishes that **MedSAM** can be pruned while maintaining strong regional and boundary performance, but it leaves three questions open. Its evaluation covers three modalities, so how the cost of pruning varies across a wider range is not established. It removes structures by importance scoring, so whether the sparsity induced by regularisation during finetuning yields a comparable result, and how either compares to a simple magnitude criterion, is not examined. Moreover, it prunes only whole structures, so the relationship between the redundancy removable at the level of individual parameters and that removable as whole units is not addressed.

These three gaps, the dependence on modality, the comparison of regularisation against magnitude, and the two senses of redundancy, motivate the present study. It treats redundancy across modalities as the object of study rather than compression as an end, measuring how much of the **MedSAM** encoder is redundant, whether that redundancy is of one kind or several, and whether the cost of removing it depends on the imaging modality being segmented. It does so by inducing sparsity through regularisation and comparing it against a magnitude criterion, across both the unstructured and structured regimes, over seven imaging modalities.

1.3. RESEARCH QUESTIONS

This leads to the central question of this thesis:

*To what extent is the **MedSAM** image encoder redundant for medical image segmentation, and does the cost of removing that redundancy depend on the imaging modality?*

This question is addressed through three sub-questions:

1. How much of the encoder is redundant, and in what sense? Redundancy is measured both at the level of individual parameters and at the level of whole computational structures.
2. Does the cost of removing this redundancy depend on the imaging modality being segmented?
3. How far can the encoder be reduced while maintaining a respectable segmentation accuracy, measured by the **DSC**?

2

BACKGROUND

2.1. SAM AND MEDSAM

Segmentation foundation models approach the task differently from the per-task networks of [chapter 1](#). Rather than fitting a model to a single structure, a single model is trained to segment any object indicated by a user prompt. The [SAM](#) is the first such model for natural images. As shown in [Figure 2.1](#), it is built from three components: an image encoder that maps the input image to a dense feature representation, the image embedding; a prompt encoder that maps a user prompt, such as a point or a bounding box, to a prompt embedding; and a lightweight transformer-based mask decoder that combines the image and prompt embeddings to predict the segmentation mask [\[21\]](#). This division of labour is what allows the model to segment without retraining. The image embedding depends only on the image, so it is computed once and reused for any number of prompts, whilst the prompt encoder and mask decoder are light and evaluated once per prompt. The cost of the model is therefore concentrated in the image encoder, which is both the largest of the three components and the only one evaluated per image rather than per prompt. [SAM](#) was trained on natural images and transfers poorly to the medical domain, where targets often have weak boundaries and low contrast. [MedSAM](#) adapts [SAM](#) to this domain by fine-tuning it on a large medical dataset comprising 1,570,263 image-mask pairs across 10 imaging modalities [\[9\]](#). The architecture is retained, with one simplification to the prompt: [MedSAM](#) is driven by bounding-box prompts alone, which are well-suited to delineating anatomical structures and lesions. The result is a single model that segments across modalities, and it is the model studied in this work.

2.2. THE ViT-B ENCODER

The image encoder of [MedSAM](#) is a Vision Transformer Base Model ([ViT-B](#)), and is the part of the model pruned in this work. To see what can be removed, it helps to follow a single input(image) through the encoder, from the raw pixels to the feature vector that leaves it. The encoder follows the [ViT](#) design, which treats an image not as a grid of pixels

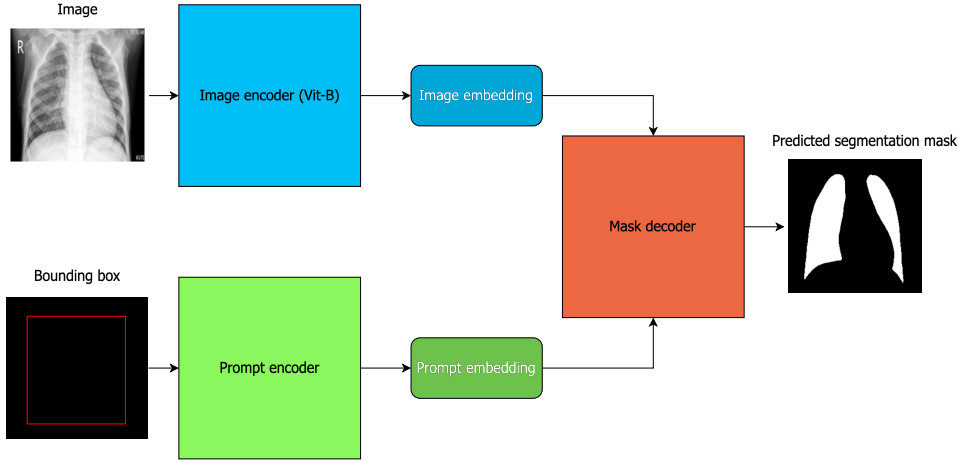


Figure 2.1: Overview of the [MedSAM](#) architecture. The image encoder maps a $1024 \times 1024 \times 3$ image to a $256 \times 64 \times 64$ image embedding and is evaluated once per image. The prompt encoder maps a bounding-box prompt to two 256-dimensional corner tokens and is evaluated once per prompt, alongside the lightweight mask decoder that fuses the two embeddings to predict the segmentation mask.

but as a sequence of patches, and processes that sequence with a transformer [22]. An overview of the computational units within the [ViT-B](#) encoder are shown in [Figure 2.2](#).

2.2.1. FROM PATCH TO TOKEN

The input image, of size 1024×1024 with three colour channels, is divided into a grid of non-overlapping 16×16 patches by a two-dimensional convolution. A learned kernel of size 16×16 , applied with a stride of 16 across the three input channels, produces 768 output channels, so each patch is mapped to a feature vector termed a token:

$$\mathbf{t} = \mathbf{W}_{\text{embed}} * \mathbf{p}, \quad (2.1)$$

where $*$ denotes convolution, $\mathbf{p} \in \mathbb{R}^{3 \times 16 \times 16}$ is a single patch across the three input channels, $\mathbf{W}_{\text{embed}} \in \mathbb{R}^{768 \times 3 \times 16 \times 16}$ is the patch embedding kernel, and $\mathbf{t} \in \mathbb{R}^{768}$ is the resulting token. Applying this convolution across the whole image yields a 64×64 grid of tokens. These tokens are collected into a single matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$, where each of the $N = 4096$ rows is one token, and each row holds $D = 768$ feature values. A learned positional embedding of matching shape is then added, so that every token carries a record of where in the image its patch came from, which the transformer would otherwise have no way of recovering [22]. The matrix $\mathbf{X}$ passes through the twelve transformer blocks shown in [Figure 2.2](#), and every operation acts on it as a whole rather than on any token in isolation. Each block applies the same two main units in turn, a [MHA](#) unit and an [MLP](#) unit.

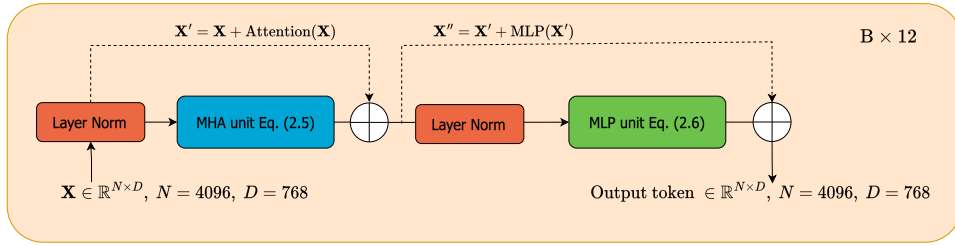


Figure 2.2: Computational graph of a single transformer block in the ViT-B encoder. The token matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ enters from the left and passes through two residual sub-blocks in turn. In the first, layer normalisation is applied and the result is passed to the Multi Head Attention (MHA) unit (Equation 2.5), whose output is added back to form $\mathbf{X}' = \mathbf{X} + \text{Attention}(\mathbf{X})$. In the second, layer normalisation is applied again and the result is passed to the MLP unit, whose output is added back to form $\mathbf{X}'' = \mathbf{X}' + \text{MLP}(\mathbf{X}')$. The residual stream retains its width of $D = 768$ throughout, so the block returns a token matrix of the same shape it received. The block structure B is repeated 12 times in sequence.

2.2.2. THE RESIDUAL STREAM

Before either unit is examined, it is worth noting how the token matrix moves through a block. Neither unit replaces $\mathbf{X}$. Each computes an update and adds it back:

$$\mathbf{X}' = \mathbf{X} + \text{Attention}(\mathbf{X}), \quad (2.2)$$

$$\mathbf{X}'' = \mathbf{X}' + \text{MLP}(\mathbf{X}'), \quad (2.3)$$

where $\mathbf{X}$ is the token matrix entering the block, $\mathbf{X}'$ is the matrix after the attention update has been added, and $\mathbf{X}''$ is the matrix after the MLP update. The representation is carried along a running total that each unit only nudges. This running total is the residual stream, and it keeps $D = 768$ columns from the first block to the last. Because the stream is never resized, pruning cannot touch it. Capacity can be removed only inside the MHA and MLP units, and that is where the pruning in this work acts.

2.2.3. THE MULTI-HEAD ATTENTION UNIT

Attention is the step in which tokens exchange information, each forming a new representation from the tokens it finds relevant. The matrix $\mathbf{X}$ is first projected into a query, a key and a value:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q^\top, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K^\top, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V^\top, \quad (2.4)$$

where $\mathbf{W}_Q$, $\mathbf{W}_K$ and $\mathbf{W}_V$ are learned projection matrices, and $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V} \in \mathbb{R}^{N \times D}$ hold the query, key and value for every token. The three projections are stored together as one tensor, at `tn.qkv` of shape $(3D, D) = (2304, 768)$, which produces all three in a single multiplication. Every query is then compared against every key. The product $\mathbf{Q}\mathbf{K}^\top$ scores how relevant each token is to every other; the scores are scaled and passed through a row-wise softmax to form weights that sum to one, and each token's output is the sum of all value rows weighted by these scores:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \quad (2.5)$$

where $d = D/h$ is the per-head dimension defined below, and scaling by $\sqrt{d}$ stops the scores from growing so large that the softmax becomes too sharp to train [23]. The output is a matrix in $\mathbb{R}^{N \times D}$, each row a token blended with information from the tokens it attended to. This comparison is not performed once over the full feature dimension but h times in parallel, which is what makes the attention multi-headed. As shown in Figure 2.3, the query, key and value are each divided into h slices of size $d = D/h$, and attention is computed independently within each slice, so every head operates on its own d -dimensional subspace and is free to capture a different kind of relationship between tokens [23]. For ViT-B, $D = 768$ and $h = 12$, so each head works on a slice of size $d = 64$. The h head outputs are concatenated back to $D = 768$ and passed through one further tensor, `attn.proj` of shape $(D, D) = (768, 768)$, which mixes the per-head results and returns the matrix to its original size. The head is the natural unit of removal in this part of the block. Each head owns a 64-column slice of the query, key and value projections, so removing a head deletes that slice from `attn.qkv`, and with it the coupled columns of `attn.proj`. The multi-head structure, therefore, partitions the attention unit into h independent pieces, each of which is prunable as a whole. This is the first of the two units pruned in this work.

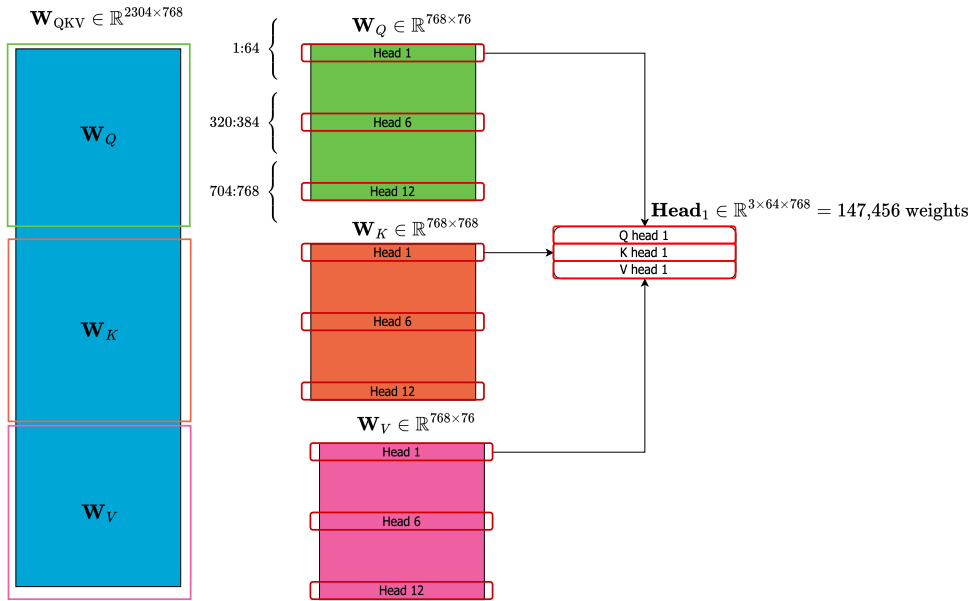


Figure 2.3: Head decomposition of `attn.qkv` weight tensor of shape $(2304, 768)$, viewed as three stacked Q , K and V blocks of 768 rows each. An example of Head_1 is shown, but every block can be decomposed into 12 head slices of 64 rows, and one head group collects the same 64 indexed row slices from all three blocks, giving $3 \times 64 \times 768 = 147,456$ weights per group.

2.2.4. THE MLP UNIT

After attention, the token matrix passes through the **MLP** unit, which acts on each token row independently, with no intertoken exchange. It is two weight matrices with a Gaussian Error Linear Unit (**GELU**) activation between them. The first weight tensor denoted as `mlp.lin1` of shape $(4D, D) = (3072, 768)$, expands each token from $D = 768$ up to $4D = 3072$ values. The activation is applied, and the second weight tensor, `mlp.lin2` of shape $(D, 4D) = (768, 3072)$, contracts it back to 768:

$$\text{MLP}(\mathbf{X}) = \text{GELU}(\mathbf{X}\mathbf{W}_{\text{lin1}}^{\top})\mathbf{W}_{\text{lin2}}^{\top}, \quad (2.6)$$

where $\mathbf{W}_{\text{lin1}}$ and $\mathbf{W}_{\text{lin2}}$ are the learned weight matrices, and **GELU** is applied element-wise to the expanded representation. The 3072 values in the expanded middle are the hidden units, or neurons. The neuron is the natural piece to remove here. Each neuron is one row of `mlp.lin1` and the matching column of `mlp.lin2`, so removing a neuron deletes both at once. This is the second of the two units pruned in this work.

2.2.5. GLOBAL AND WINDOWED ATTENTION

One detail remains. Comparing every token against every other means the attention scores $\mathbf{Q}\mathbf{K}^{\top}$ form an $N \times N$ matrix, so the cost of attention grows with the square of the sequence length N . With $N = 4096$ tokens, performing this full comparison in every block would be expensive. Most blocks are therefore restricted to windowed attention, in which the tokens are partitioned into small non-overlapping windows and each token attends only to the others within its own window [21]. Only four of the twelve blocks retain global attention, in which every token attends to the full sequence. The four global blocks spread information across the entire image, while the eight windowed blocks keep their comparisons local and more efficient. This split changes the cost of each block, as quantified in [chapter 3](#).

2.3. NETWORK PRUNING

Pruning reduces a model by removing parts of it that contribute little to its output. It rests on the finding, discussed in [chapter 1](#), that a trained network holds more weights than the task requires [12]. What is removed and how it is chosen define the kind of pruning.

2.3.1. UNSTRUCTURED AND STRUCTURED PRUNING

Pruning is commonly divided into two types based on what it removes. Unstructured pruning replaces individual weights with zeros while leaving the tensor's shape unchanged. The result is a sparse tensor, one with many zeros scattered through it, but with the same dimensions as before. Structured pruning instead removes whole groups of weights at once, such as an attention head or an **MLP** neuron, and rebuilds the tensor at a smaller size. The difference matters for what is gained. Unstructured pruning produces a model with fewer non-zero weights but the same shape, so on standard hardware, it runs no faster, as the zeros are still stored and multiplied. Its saving is in principle, not in practice. Structured pruning removes whole units, so the tensors genuinely shrink and the model becomes smaller and faster to run [13]. The two, therefore, measure different things: un-

structured pruning shows how many individual weights are unnecessary, whilst structured pruning shows how much whole-unit computation can be removed for a real saving. Both are used in this work, one for each of the two senses of redundancy set out in chapter 1.

2.3.2. THE PRUNING CRITERION

Whichever kind is used, a rule is needed to decide what to remove. The simplest and most common is magnitude, the assumption that weights with small absolute values contribute the least and can be removed first [12]. A weight tensor $\mathbf{W}$ is pruned by removing the entries, or the groups, of smallest magnitude, up to a chosen fraction. This is the criterion followed throughout this work. The methods differ not in the criterion but in how the small-magnitude weights are produced, which is the subject of section 2.4.

2.4. SPARSE OPTIMISATION AND PROXIMAL METHODS

Magnitude pruning rests on the assumption that low-magnitude weights are the least important and can be removed first [12]. Ordinary training does not encourage this. Without any pressure towards zero, the trained weights keep broadly comparable magnitudes, so a magnitude threshold separates the important from the unimportant only weakly. A sparsity-inducing penalty is added to the training objective to address this, actively driving weakly supported weights towards zero so that they can later be removed at little cost [24]. Two such penalties are used in this work, one for each of the two pruning regimes.

2.4.1. THE LASSO AND SPARSE GROUP LASSO PENALTIES

The first penalty is the ℓ_1 norm of the weights, the Lasso, originally introduced for simultaneous shrinkage and variable selection in regression [25]. It drives individual weights towards zero and corresponds to the unstructured regime:

$$\mathcal{P}_{\ell_1}(\mathbf{w}) = \lambda \sum_i |w_i|, \quad (2.7)$$

where $\mathbf{w}$ is the vector of prunable weights, w_i is an individual weight, and λ controls the strength of the penalty. The ℓ_1 norm penalises each weight independently, so it induces sparsity at the level of the individual weight but imposes no structure on which weights are removed. The group Lasso extends this idea from individual weights to predefined groups, driving whole groups to zero together rather than scattered weights [26]. The Sparse Group Lasso (SGL) combines the two [27], adding the group-wise term to the element-wise one so that both kinds of sparsity are induced at once, corresponding to the structured regime:

$$\mathcal{P}_{\text{SGL}}(\mathbf{w}) = \lambda_1 \sum_i |w_i| + \lambda_2 \sum_l \sqrt{p_l} \|\mathbf{w}_l\|_2, \quad (2.8)$$

where $\mathbf{w}_l$ is the vector of weights in group l , the groups l partition the weights, p_l is the number of weights in group l , and λ_1 and λ_2 control the element-wise and group-wise strengths respectively. The group term is weighted by $\sqrt{p_l}$ so that groups of differing

size are penalised comparably. In the original convention of Simon et al. [27], the two coefficients are tied through a single strength λ and a mixing parameter $\alpha \in [0, 1]$, as $\lambda_1 = \lambda\alpha$ and $\lambda_2 = \lambda(1 - \alpha)$. In this work, they are treated as independent, allowing the two kinds of sparsity to be controlled separately.

2.4.2. PROXIMAL GRADIENT DESCENT

Both penalties are non-smooth. The absolute value in Equation 2.7 and the group norm in Equation 2.8 are not differentiable at zero, which is exactly the value they are meant to drive weights to. Adding such a term directly to the loss and differentiating would place a non-smooth term in the gradient path, where the gradient at zero is ill-defined [28]. Proximal gradient descent avoids this by splitting the update in two [28]. An ordinary gradient step minimises the smooth task loss, and the non-smooth penalty is then applied separately through its proximal operator. This closed-form shrinkage pulls weights towards zero. For the ℓ_1 norm, this proximal operator is the soft-threshold, which shrinks each weight by a fixed amount and sets to zero any weight already smaller than that amount [28]. The penalty does not enter the differentiated loss, so the gradient step is unaffected, and the non-smoothness is confined to the shrinkage. This is the mechanism by which both penalties are imposed in this work; the specific operators and the order in which they are applied are set out in subsection 3.1.4.

3

METHODS

3.1. PROMOTING SPARSITY OF THE MEDSAM ENCODER

3.1.1. MODEL AND PRUNABLE PARAMETERS

The model used throughout this work was retrieved from Ma et al. [9] and made available through their public GitHub repository. MedSAM comprises three components: a ViT-Base image encoder, a prompt encoder, and a mask decoder. The image encoder maps a $1024 \times 1024 \times 3$ input image to a 64×64 image embedding, a sixteen-fold spatial down-scaling. The prompt encoder maps a bounding-box prompt to a positional embedding, and the mask decoder fuses the image and prompt embeddings to produce a segmentation mask. A full description of the underlying architecture is provided in [section 2.1](#) and [section 2.2](#). Pruning in this work is limited to the ViT-Base image encoder, which accounts for the majority of the model’s parameter count and computational cost.

The parameters of the ViT image encoder are distributed across twelve transformer blocks, each comprising a MHA unit and an MLP unit. [Table 3.1](#) reports the composition of a single block. Four weight tensors were designated as prunable: the query, key and value projection (`attn.qkv`), the attention output projection (`attn.proj`), and the two MLP linear layers (`mlp.lin1` and `mlp.lin2`). Together, these constitute 7,077,888 parameters, or 99.81% of a block, of which the MLP layers form two-thirds. The remaining parameters of a block, comprising the layer normalisation scales, all bias terms, and the relative position embeddings, together account for less than 0.2%.

These four tensors, taken across all twelve blocks, form the set of 48 prunable weight tensors used throughout this work. They were selected for three reasons. First, they contain the overwhelming majority of the encoder’s parameters. At 84,934,656 parameters, they account for 94.72% of the image encoder and 90.61% of the full model, so pruning confined to these tensors nonetheless affects almost the entire model. The encoder-level figure falls below the block-level share of 99.81% only due to the one-off patch embedding, positional embedding, and neck, which together contribute 4,523,776 parameters and are not repeated per block. Second, these are the only encoder components whose structure admits meaningful structured pruning. Their rows and columns

map directly to removable units, namely attention heads for `attn.qkv` and `MLP` neurons for `mlp.lin1`. The excluded components are either negligible in size or do not decompose into prunable groups. Third, the four tensors form two coupled pairs, `attn.qkv` with `attn.proj` and `mlp.lin1` with `mlp.lin2`, within which pruning one tensor induces pruning of the other. The mechanism by which this occurs is described in [subsection 3.1.4](#). The twelve blocks are identical in their prunable structure. They differ only in the relative position embeddings, which are larger in the four blocks that employ global attention, as reflected in [Table 3.1](#).

Component	Parameters	Share of block	Status
<code>attn.qkv</code>	1,769,472	24.95%	Prunable
<code>attn.proj</code>	589,824	8.32%	Prunable
<code>mlp.lin1</code>	2,359,296	33.27%	Prunable
<code>mlp.lin2</code>	2,359,296	33.27%	Prunable
Prunable subtotal	7,077,888	99.81%	
Layer norm scales	3,072	0.04%	Excluded
Bias terms	6,912	0.10%	Excluded
Relative position	3,456 / 16,256	0.05%	Excluded

Table 3.1: Per block parameter composition of the [ViT](#)-Base encoder, separating the four prunable weight tensors from the excluded components. Counts are for weights only, so bias terms are listed separately. The relative position embedding takes the smaller value in the eight windowed attention blocks and the larger value in the four global attention blocks (blocks 2, 5, 8 and 11). Shares are computed against a windowed attention block total of 7,091,328 parameters.

3.1.2. INDIVIDUAL WEIGHT SPARSITY BY LASSO

The unstructured regime promotes sparsity in individual weights via the Lasso penalty in [Equation 2.7](#), as defined in [section 2.4](#). The penalty was applied during finetuning to the `attn.qkv` and `mlp.lin1` tensors, driving individual weights towards zero without regard to their position within a head or neuron. As the penalty acts element-wise, the resulting sparsity is scattered throughout each tensor rather than confined to removable units, and the tensor shapes are preserved.

Pruning under this regime measures induced sparsity in individual weights by setting them to zero, leaving the physical weight tensor shapes unchanged. Pruning is applied independently to each prunable tensor based on a per-tensor threshold. This threshold is set as a fraction of the tensor's own magnitude scale, rather than a single global threshold, because the prunable tensors differ in magnitude, both across tensor types and across heads ([MHA](#)) or layers ([MLP](#)). A global threshold would prune them unevenly, removing most of a low-magnitude tensor while barely touching a high-magnitude one. The reference value is taken as the 95th percentile of the absolute weights, discarding the extreme tail to make the threshold robust to a small number of outlier weights. For a pruning fraction ρ , the threshold is ρ times this reference value, and every weight whose magnitude falls below the threshold is set to zero. When at least 95% of a tensor is already exactly zero, the 95th percentile is itself zero; in this case, the reference value falls

back to the maximum absolute weight of the tensor.

Because individual weights are zeroed without altering tensor shapes, the unstructured regime measures sparsity rather than prescribing it, and the model is not physically reduced in size. No inference speed-up or Floating Point Operations (FLOPs) reduction therefore follows from unstructured pruning; the regime instead characterises how much of the encoder can be zeroed before accuracy degrades.

3.1.3. STRUCTURAL PRUNING BY SPARSE GROUP LASSO

The structured regime promotes sparsity across entire computational units via the SGL penalty in Equation 2.8, as defined in section 2.4. As with the Lasso, the penalty was applied during finetuning to the `attn.qkv` and `mlp.lin1` tensors, from which the pruning of `attn.proj` and `mlp.lin2` follows as a consequence, as described in subsection 3.1.4. The SGL coefficients λ_1 and λ_2 were treated as independent rather than tied through a single strength, so that the element-wise and group-wise penalties could be swept separately. The group term drives whole heads and MLP neurons towards zero, producing sparsity that is confined to removable units rather than scattered across the tensor.

Pruning under this regime removes entire computational units rather than individual weights, physically rebuilding the affected weight tensors with a smaller width. Two unit types are pruned: attention heads in the MHA block, and neurons in the MLP block. Removing a unit reduces the model's dimensionality, so the pruned model is genuinely smaller and cheaper to run, in contrast to the unstructured case, where the tensor shapes remain unchanged. Units are ranked within each block by the Euclidean Norm (L2) norm of their weights, using the same groups that were penalised during training. An attention head is scored by the norm of its full `attn.qkv` group, spanning the query, key and value weights for that head; an MLP neuron is scored by the norm of its `mlp.lin1` row. The lowest-norm units are removed first. Ranking is performed independently in each block, so the same number of units are removed from each block, but the specific units removed may differ.

Removing a unit prunes the tensors it feeds. When an attention head is removed, the corresponding input columns of `attn.proj` become redundant and are removed with it; likewise, removing an MLP neuron removes the corresponding input column of `mlp.lin2`. These downstream tensors are never ranked themselves; their pruning follows from the units removed upstream. Heads and neurons are removed together in a fixed proportion. The MHA block has twelve heads, which gives twelve integer steps to prune from, so pruning is controlled by the number of heads removed per block, k , swept from 0 (unpruned) to 11. Each head removed corresponds to a fraction $k/12$ of the heads. The same fraction is applied to the MLP neurons: with 3072 neurons, a fraction $k/12$ removes $3072 \times k/12 = 256k$ neurons per block. Both components are therefore pruned at the same fraction $k/12$, controlled by the single integer k .

The structured regime admits a limiting configuration in which the pruning unit is taken to the extreme and the transformer blocks are removed entirely. The twelve blocks are discarded from the finetuned model, and only the patch embedding, positional embedding and neck are retained. The image is therefore encoded solely by the patch embedding and neck, with no attention or MLP processing in between. The retained components keep their finetuned weights, and the output matches the shape of

the full encoder, so the mask decoder and prompt encoder are unaffected. This configuration serves as the floor of the structured sweep: the point at which the transformer is removed in full, rather than reduced head by head.

3.1.4. PROXIMAL GRADIENT DESCENT

The penalties of Equation 2.7 and Equation 2.8 are non-smooth, and their subgradients at the origin are ill-defined. Adding them to the training objective and differentiating would therefore place a non-smooth term in the gradient path, which is avoided here. Instead, the penalties are imposed via proximal gradient descent [28], in which a gradient step minimises the smooth task loss, and the non-smooth penalty is applied afterwards via its proximal operator. It is important to note that the penalties defined in Equation 2.7 and Equation 2.8 do not induce sparsity themselves, but are used to derive the sparsity-inducing closed-form solutions found below in Equation 3.1 and Equation 3.2. Each iteration proceeds in two stages. First, a standard gradient step is taken through Stochastic Gradient Descent (SGD) on the task loss alone, updating all trainable weights through the optimiser. Second, the penalty's proximal operator is applied directly to the prunable weights, shrinking them towards zero. The penalty does not enter the differentiated loss, so the gradient step is unaffected by it, and the non-smoothness is confined to the closed-form proximal update.

The proximal operator of the ℓ_1 penalty is the element-wise soft-threshold [29]. For a weight w and step size t ,

$$S_\tau(w) = \begin{cases} w - \tau & \text{if } w > \tau, \\ w + \tau & \text{if } w < -\tau, \\ 0 & \text{otherwise,} \end{cases} \quad \tau = \lambda_1 t, \quad (3.1)$$

where t is the learning rate. Each weight is shrunk towards zero by $\lambda_1 t$ and set to zero once its magnitude falls below that threshold. This operator defines the Lasso regime, in which no group term is present. For the SGL, the proximal operator combines element-wise soft-thresholding with group-wise shrinkage, applied in that order [29]. For a group l of post-gradient weights $\mathbf{r}_l$,

$$\mathbf{w}_l = \left(1 - \frac{\lambda_2 t \sqrt{p_l}}{\|S_{\lambda_1 t}(\mathbf{r}_l)\|_2} \right)_+ S_{\lambda_1 t}(\mathbf{r}_l), \quad (3.2)$$

where $S_{\lambda_1 t}$ is the element-wise soft-threshold of Equation 3.1, p_l is the number of weights in group l , $\|\cdot\|_2$ is the Euclidean norm, and $(\cdot)_+$ denotes the positive part. The soft-threshold is applied first, and the result is then scaled towards zero as a group; a group whose soft-thresholded norm falls below $\lambda_2 t \sqrt{p_l}$ is set to zero in its entirety. In the convention of Klosa et al. [29], the two coefficients are expressed through a single strength λ and a mixing parameter α , as $\lambda_1 = \alpha \lambda$ and $\lambda_2 = (1 - \alpha) \lambda$; Equation 3.2 follows under that identification and is written here with λ_1 and λ_2 treated as independent, consistent with Equation 2.8.

The proximal step operates on the weights after the optimiser update and does not contribute to the gradient. The penalty strengths λ_1 and λ_2 are therefore decoupled from the optimiser's own update rule, and the sparsity they induce is measured from the

resulting weights rather than prescribed in advance. The element-wise soft-threshold in Equation 3.1 is applied to the `attn.qkv`, `attn.proj`, `mlp.lin1` and `mlp.lin2` tensors, whilst the group operator in Equation 3.2 is applied only to the `attn.qkv` and `mlp.lin1` tensors. The remaining prunable tensors are pruned as a structural consequence, as established in subsection 3.1.3.

3.2. ANALYTICAL DERIVATION OF COMPUTATIONAL COST

3.2.1. FLOATING POINT OPERATIONS

The computational cost of a model can be quantified by its number of FLOPs, which counts the additions and multiplications required for a single forward pass. As pruning removes attention heads and MLP neurons, the number of these operations falls, and the resulting reduction in FLOPs provides a hardware-independent measure of the savings achieved. This section derives the FLOPs of a single encoder block as a function of the retained heads and neurons, so that the cost can be related directly to the extent of pruning. The pruning methods in this work reduce two quantities within each block: the number of retained attention heads and the retained hidden dimension of the MLP. As the embedding dimension and the residual stream are unaffected by pruning, only the attention and MLP costs vary, and the block cost is expressed in terms of these two variables to isolate the components that pruning influences. The variables are defined as follows:

- h' : number of retained attention heads, $0 \leq h' \leq h$,
- m' : retained MLP hidden dimension, $0 \leq m' \leq 4D$,
- d : head dimension, fixed at $D/h = 64$,
- D : embedding dimension, fixed at 768,
- B' : effective batch, $B n_w$,
- N : tokens per attention window, $H_w W_w$,
- M : window side length, $H_w = W_w$.

Pruning acts only through h' and m' ; all other quantities are fixed properties of the architecture. The attention cost is proportional to the number of retained heads, since every operation within attention is either a per-head computation or a projection onto the retained head dimensions. Collecting the per-head contribution into a single coefficient

$$\kappa_{\text{attn}} = B' \left[Nd + N^2(2d - 1) + (3N^2 - N) + Nd(2N - 1) + 2M^3(2d - 1) + 2M^4 \right], \quad (3.3)$$

the attention cost is written compactly as

$$\text{FLOPs}_{\text{attn}}(h') = 8Dd B' N h' + \kappa_{\text{attn}} h'. \quad (3.4)$$

The first term is the cost of the query, key, value and output projections, which scale with the retained head dimension $h'd$; the second is the cost of the attention operation itself, which is performed independently per head. The MLP cost is proportional to the retained hidden dimension:

$$\text{FLOPs}_{\text{MLP}}(m') = B H_p W_p (4D + 9) m', \quad (3.5)$$

where BH_pW_p is the number of tokens processed, $4D$ is the combined cost per hidden unit per token of the two linear projections, at $2D$ each, and the constant 9 is the cost per hidden unit per token of the GELU activation. The hidden dimension m' is the sole pruning variable, ranging from $4D = 3072$ at the baseline down to zero. The full block cost combines the attention and MLP costs with the fixed cost of the two-layer normalisations and two residual additions, neither of which depends on pruning:

$$\text{FLOPs}_{\text{block}}(h', m') = \underbrace{(16D + 4) BH_pW_p}_{\text{Fixed Floor}} + \text{FLOPs}_{\text{attn}}(h') + \text{FLOPs}_{\text{MLP}}(m'). \quad (3.6)$$

Where the Fixed Floor = Layer Norms($2 \times (7D + 2)$) + Residual Connections($2D$). At the unpruned baseline, $h' = h = 12$ and $m' = 4D = 3072$, recovering the dense block cost.

3.2.2. INFERENCE TIME

Inference time was measured as the wall-clock time for a single forward pass through the full model at a batch size of 1. A warm-up pass was performed beforehand so that the first measured sample was not penalised by device initialisation. Timing was recorded per sample and reported as the mean and standard deviation across the evaluation set.

3.3. EVALUATION FRAMEWORK

3.3.1. SEGMENTATION ACCURACY

Segmentation accuracy was quantified using the DSC, which measures the overlap between the predicted segmentation S and the ground truth G :

$$\text{DSC}(G, S) = \frac{2|G \cap S|}{|G| + |S|}. \quad (3.7)$$

The DSC ranges from 0 to 1, with 1 indicating perfect overlap between the two masks. The DSC was computed across all samples in the validation and test sets. Empty predictions and ground truths were excluded from the reported scores, as they would otherwise yield a perfect score of 1 and skew the reported statistics.

3.3.2. ACCURACY AND COST TRADE-OFF

The effect of pruning was assessed along two dimensions considered jointly: the segmentation accuracy of the pruned model, and the computational cost it incurs. Accuracy is quantified by the DSC, and cost by the number of parameters, the derived FLOPs of subsection 3.2.1, and the measured inference time. A pruning configuration is judged not by either dimension alone but by the accuracy it preserves at a given level of cost reduction.

3.4. IMPLEMENTATION DETAILS

3.4.1. DATASETS

To investigate the effect of pruning across diverse medical imaging modalities, seven datasets were curated for finetuning and evaluation, one per modality. All datasets were drawn exclusively from the Ma et al. [9] (Supplementary Material) external validation

setup, ensuring no overlap with the MedSAM training data and enabling meaningful baseline comparisons with Ma et al. [9]’s reported performance. Datasets were selected such that they did not form part of the MedSAM training set, whilst being publicly available for extraction. Each dataset was selected to represent a different imaging acquisition modality. Of the possible modalities in the external validation corpus, seven were retrievable. The mammography and Positron Emission Tomography (PET) modality were excluded as no external validation set was defined [9]. The UWaterloo dermoscopy dataset [30] was excluded because it was not publicly available at the time of this study.

Dataset	Modality	Segmentation Target
ACDC [31]	MRI	RV, MYO, LV
PAPILA [32]	Fundus	Optic disc and cup
Adrenal-ACC-Ki67-Seg [33]	CT	Adrenocortical carcinoma
Kvasir-SEG [34]	Endoscopy	Colorectal polyp
HC18 [35]	Ultrasound	Foetal head
GlaS 2015 [36]	Pathology	Colorectal glandular structures
COVID-19 Radiography [†] [37]	CXR	Lungs

Table 3.2: Summary of the seven datasets selected for finetuning and evaluation, one per imaging modality. [†] The COVID-19 Radiography dataset contains four classes; only the viral pneumonia class was used in this study. Magnetic Resonance Imaging (MRI); Right Ventricle (RV); Myocardium (MYO); Left Ventricle (LV); Computed Tomography (CT); Chest X-ray (CXR).

3.4.2. PREPROCESSING

IMAGE PROCESSING

The preprocessing pipelines were designed to closely follow the steps used by Ma et al. [9] to ensure that the inputs conform to the MedSAM model’s expected input structure. All images were resized to 1024×1024 and stored as three-channel float32 arrays, resulting in a tensor of size $3 \times 1024 \times 1024$, and all masks were stored as uint8 arrays. Images were resized using bicubic interpolation, whilst masks were resized using nearest-neighbour interpolation to preserve label boundaries [9]. Intensity normalisation was applied per modality, following the conventions reported by Ma et al. [9]. For the CT modality, Hounsfield units were windowed to the soft-tissue range (width 400, level 40) before rescaling to $[0, 255]$. For the greyscale modalities (MRI, ultrasound, and CXR), intensities were clipped to the 0.5th and 99.5th percentiles before rescaling to $[0, 255]$. For the Red Green Blue (RGB) modalities (fundus, endoscopy, and pathology), intensities were left unchanged where they already fell within $[0, 255]$, and rescaled by min-max normalisation otherwise. Single-channel images were replicated across three channels to match the expected input dimensionality. Several datasets required additional handling prior to resizing. The HC18 masks are provided as ellipse outlines, so a hole-filling operation was applied to produce solid foetal-head regions. The Kvasir-SEG masks are stored in a lossy compressed format, so they were binarised at their original resolution before resizing, to prevent compression artefacts from corrupting the mask boundaries. Two independent experts annotate each PAPILA structure. The two contours were rasterised

to binary masks and combined by their union, so that a pixel was labelled foreground if either expert delineated it. This gives the more inclusive of the two annotations. Two datasets contain multiple structures per image. The ACDC masks were stored with one channel per cardiac structure (RV, MYO, and LV), and the GlaS masks with one channel per gland instance, with the number of glands varying across instances. As MedSAM is prompted per object, each channel was treated as an independent segmentation target with its own bounding-box prompt.

DATA SPLITTING

Data was split in two ways: at the patient level or at the image level. When patient identities could be recovered from the filenames, splitting was performed at the patient level, so that all images belonging to a given patient were assigned to the same partition. This prevents data leakage, whereby images of the same patient appearing in both the training and evaluation partitions would inflate the reported performance. For PAPILA, both eyes of each patient were kept together, and for ACDC, all slices of both the end-diastole and end-systole frames of each patient were kept together. Where patient identity was not released, as for Kvasir-SEG and the COVID-19 Radiography dataset, splitting was performed at the image level. The ACDC and GlaS datasets are provided with official train/test partitions by the initial challenge. GlaS provides two test sets, denoted testA and testB. These splits were preserved to remain comparable to prior work, with the training split further divided 90:10 into training and validation subsets. All other datasets were split into 70:10:20 ratios among training, validation, and test sets. A summary of the image-mask composition of the datasets is given in Table 3.3. A fixed random seed was used across all code to ensure split reproducibility.

Dataset	Split level	Images			Masks	
		Train	Val	Test	Val	Test
ACDC	Patient*	1638	203	1001	571	2847 [‡]
PAPILA	Patient	342	48	98	96	196
Adrenal-ACC-Ki67-Seg	Patient	1053	159	666	159	666
HC18	Patient	700	101	198	101	198
GlaS 2015	Slide*	77	8	60 / 20 [¶]	70	666 / 95 [¶]
Kvasir-SEG	Image	700	100	200	100	200
COVID-19 Radiography	Image	942	134	269	134	269

Table 3.3: Data partition sizes for the seven datasets. Image columns give the number of images per split; mask columns give the number of individual binary masks, obtained by expanding each non-empty structure or gland instance into a separately scored mask. The two are equal for single-structure datasets. Training uses a single randomly selected structure per image, so it is reported only for images. *Official partition preserved, with its training partition divided 90:10 into training and validation; all remaining datasets divided 70:10:20. [‡]Comprising RV (881), MYO (989), and LV (977). [¶]GlaS provides two official test partitions, testA / testB.

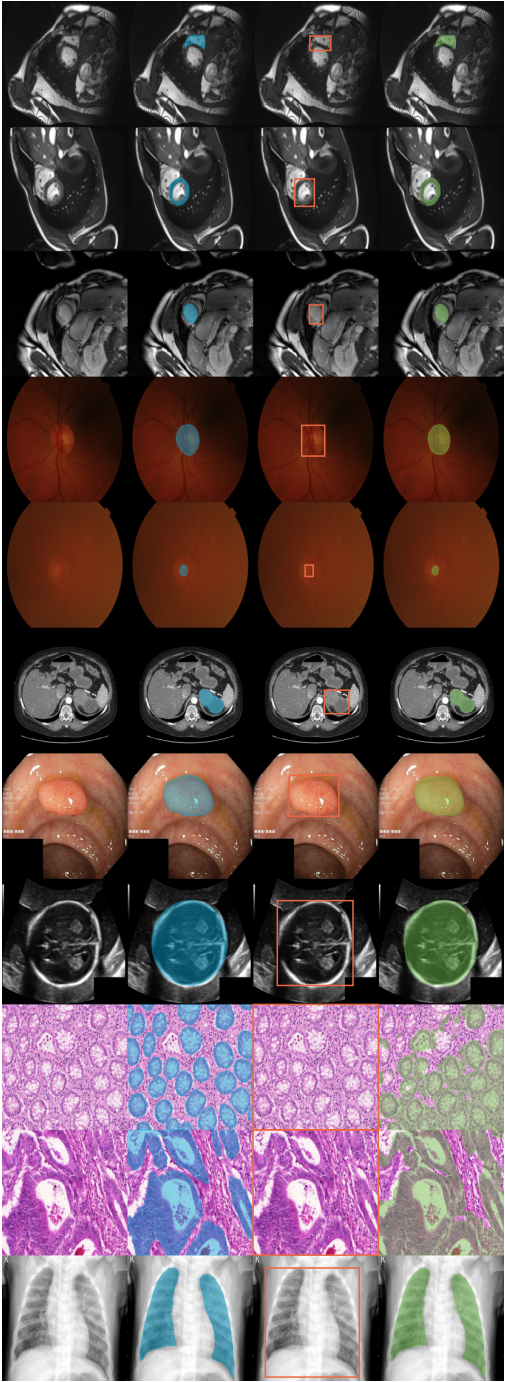


Figure 3.1: Per-structure visualisation of the eleven evaluation subsets. Columns show, from left to right: the input image, the ground-truth mask, the bounding-box prompt derived from the ground truth, and the predicted segmentation from the task-loss-finetuned baseline. Rows correspond, from top to bottom, to: (1) ACDC, RV; (2) ACDC, MYO; (3) ACDC, LV; (4) PAPILA, optic disc; (5) PAPILA, optic cup; (6) Adrenal-ACC-Ki67-Seg, adrenocortical carcinoma; (7) Kvasir-SEG, colorectal polyp; (8) HC18, foetal head; (9) GlaS 2015 testA, glandular structures; (10) GlaS 2015 testB, glandular structures; (11) COVID-19 Radiography, lungs. Masks are shown as a semi-transparent colour fill with a solid contour outline.

3.4.3. FINETUNING PROTOCOL

TASK-LOSS BASELINE

A task-loss finetuning baseline was established prior to the introduction of any sparsity-inducing penalty. The prompt encoder was frozen throughout, whilst the image encoder and mask decoder were finetuned, following the training protocol of Ma et al. [9]. The seven datasets were pooled into a single training set, from which a single structure was randomly sampled per image at each step. Bounding-box prompts were derived from the ground-truth masks and perturbed by up to 20 pixels per edge, sampled independently, to improve robustness to imprecise prompts. The loss function was the unweighted sum of the Dice loss and the binary cross-entropy loss, consistent with Ma et al. [9]. Optimisation was performed using AdamW at a learning rate of 1×10^{-4} , with a batch size of two, for 50 epochs. The checkpoint used for subsequent pruning and evaluation was the epoch that achieved the highest validation **DSC**, computed as a sample-averaged mean of per-sample **DSC** values pooled across the validation splits of all datasets. Validation **DSC** was used for selection rather than validation loss, as it directly measures segmentation overlap.

MODEL SETS

Three sets of models were produced. The baseline was finetuned once on the task loss alone, as described in subsection 3.4.3, and serves as the reference against which all pruning is measured. The Lasso and **SGL** penalties were then applied during finetuning, each over a sweep of penalty strengths, yielding a separate finetuned model for each penalty setting. The encoder-removed configuration required no finetuning of its own, as it is obtained by discarding the blocks from the task-loss-finetuned model.

3.4.4. PRUNING SWEEPS

LAGO SWEEP

For the Lasso penalty, five models were finetuned, one for each penalty strength $\lambda \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. The penalty drives individual weights towards zero during finetuning, and the sweep then measures how far each model can be pruned before accuracy degrades. Each finetuned model was pruned in an unstructured manner, swept over the pruning fraction $\rho \in \{10\%, 20\%, 30\%, 50\%, 70\%, 100\%\}$, and evaluated at each fraction.

SGL SWEEP

For the **SGL** penalty, the two coefficients were swept independently, λ_1 and λ_2 each taking the values $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$, giving a five-by-five grid of twenty-five finetuned models. Sweeping the coefficients separately allows the element-wise and group-wise penalties to be varied against one another, rather than tied to a single strength. The penalty drives whole heads and neurons towards zero during finetuning, and the sweep then removes the lowest-norm ones and measures the effect. Each model was pruned in a structured manner, swept over the number of heads removed per block k from zero (unpruned) to eleven. At each k , the matching number of neurons is removed, so both components sit at the same fraction $k/12$, as described in subsection 3.1.3. Every model was evaluated at every k .

ENCODER-REMOVED

The encoder-removed configuration is a single model. The twelve blocks are discarded from the baseline, and no recovery training is performed, as described in [subsection 3.1.3](#). It is evaluated in the same way as the pruning sweeps and serves as the floor of the structured sweep: the point at which the transformer is removed in full, rather than reduced head by head.

3.4.5. HARDWARE AND SOFTWARE

All experiments were run on a single workstation with an NVIDIA RTX 4090 (24 GB) and CUDA 12.6. The pipeline was implemented in Python 3.10 using PyTorch 2.10.0, and experiments were tracked with Weights & Biases.

4

RESULTS

4.1. TASK-LOSS FINETUNING

The Zero-shot model was evaluated using the pretrained weights provided by Ma et al. [9]. The Task-loss finetuned model referred to in Table 4.1.1 is evaluated following finetuning, using the protocol defined in subsection 3.4.3 on the pretrained model using the training split from Table 3.3.

4.1.1. ZERO-SHOT VS TASK-LOSS FINETUNED

Dataset (subset)	Median DSC (%) (Q1, Q3)	
	Zero-shot	Task-loss finetuned
Adrenal-ACC-Ki67-Seg	94.14 (90.53, 95.60)	95.02 (92.57, 96.59)
Kvasir-SEG	97.26 (96.03, 98.02)	96.55 (93.69, 97.81)
HC18	96.77 (95.65, 97.46)	98.16 (97.69, 98.48)
COVID-19 Radiography	97.52 (96.12, 98.10)	98.29 (97.64, 98.71)
PAPILA (Optic Disc)	97.80 (97.07, 98.16)	96.29 (94.95, 97.31)
PAPILA (Optic Cup)	91.28 (83.38, 94.29)	84.88 (75.01, 88.82)
ACDC (LV)	96.15 (93.46, 97.26)	95.47 (90.34, 97.02)
ACDC (RV)	95.10 (91.78, 96.49)	94.98 (91.49, 96.59)
ACDC (MYO)	32.12 (12.48, 58.79)	89.50 (86.24, 92.09)
GlaS 2015 (testA)	96.12 (94.05, 97.16)	94.04 (90.13, 96.04)
GlaS 2015 (testB)	95.19 (92.86, 97.02)	90.95 (85.05, 95.05)
Inference (ms) (Q1, Q3)	61.00 (60.85, 62.70)	60.86 (60.71, 62.41)
Parameters (M)		93.74
Encoder FLOPs (G)		966

Table 4.1: Median DSC with interquartile range for the zero-shot and task-loss finetuned models across all eleven evaluation structures, alongside median inference time, parameter count and encoder FLOPs. Parameters and encoder FLOPs are identical across both models as neither is altered by finetuning.

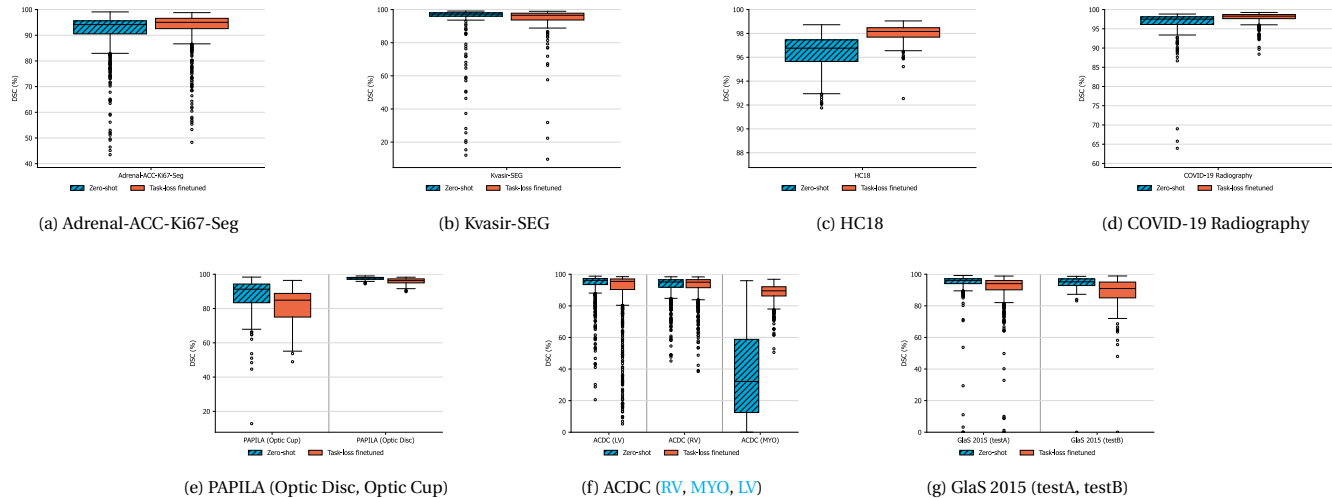


Figure 4.1: Per-sample **DSC** for the zero-shot and task-loss finetuned models across all eleven evaluation structures, grouped by dataset. Boxes span Q1 to Q3 with the median marked; whiskers extend to $1.5 \times \text{Inter Quartile Range (IQR)}$, and outliers are shown as open circles. Single-structure datasets use per-panel adaptive y-axes; the multi-structure datasets PAPILA, ACDC and GlaS 2015 share a y-axis within each panel. As axis ranges differ between panels, absolute comparison across datasets is directed to [Table 4.1.1](#).

Task-loss finetuning was applied to establish the reference model against which all subsequent compression is measured, and its effect on median DSC was neither uniform in direction nor in magnitude across the eleven structures. The results in Table 4.1.1, with per-sample distributions in Figure 4.1, are separated into structures for which finetuning improved segmentation and structures for which it degraded it.

The clearest improvement was on the MYO, where the median DSC rose from 32.12 to 89.50 (+57.38), the largest shift in Table 4.1.1. The change was accompanied by a contraction of the IQR from 46.31 to 5.85, as shown in Figure 4.1f by a much tighter box. More modest gains followed on three of the four single-structure datasets, with the median increasing for HC18 (+1.39), Adrenal-ACC-Ki67-Seg (+0.88) and COVID-19 Radiography (+0.77). In each case, the gain coincided with a reduced IQR, seen in Figure 4.1c, Figure 4.1a and Figure 4.1d, so the distributions both rose and tightened.

For the remaining structures, the median declined. The reductions were smallest on the cardiac and single-structure targets, with the LV (−0.68), RV (−0.12) and Kvasir-SEG (−0.71) each falling by under one point; for Kvasir-SEG the box widened, though its low outliers became less extreme, clustering above 85% rather than extending below 70% as in the zero-shot condition (Figure 4.1b). Larger reductions were observed in histopathology and fundus structures. Both GlaS splits fell, testA by −2.08 and testB by −4.24, and in both the lower quartile dropped further than the upper so that the box widened downward (Figure 4.1g); testB additionally developed a denser band of low outliers between 45 and 70 DSC, whilst extreme outliers near 0 persisted under both models. The sharpest degradation was on PAPILA, where the Optic Cup fell by −6.40 to a median of 84.88 against the zero-shot value of 91.28, and the Optic Disc by −1.51; both structures widened their IQR under finetuning (Figure 4.1e).

The cost of the model was unaffected by finetuning. Median inference time differed by only 0.14 ms (61.00 against 60.86), and the parameter count (93.74M) and encoder cost (966 GFLOPs) were identical across both models, as neither is altered by finetuning.

4.2. LASSO REGULARISATION

Lasso-penalised finetuning was applied to the task-loss-finetuned model to concentrate the encoder weights and expose unstructured redundancy. The penalty strength λ was swept across five values, and the effect was traced from the training dynamics, through the resulting weight distribution, to the accuracy achieved at each level of measured sparsity.

4.2.1. TRAINING DYNAMICS

The training dynamics in Figure 4.2 established a trade-off between the two losses that scaled with λ . The reduction in sparsity loss (Figure 4.2a) grew with the penalty strength, whilst the two weakest penalties left it effectively unchanged: at $\lambda = 10^{-5}$ it rose slightly, from 4,120,100 to 4,199,624, and at $\lambda = 10^{-4}$ it held approximately constant. A gradual reduction to 3,423,033 was observed at $\lambda = 10^{-3}$, and the two strongest penalties reduced it substantially, to 469,791 at $\lambda = 10^{-2}$ and to 13,196 at $\lambda = 10^{-1}$; for the strongest penalty most of the reduction occurred within the first five epochs, after which the curve flattened. The task loss (Figure 4.2b) showed the converse ordering. The three weak-

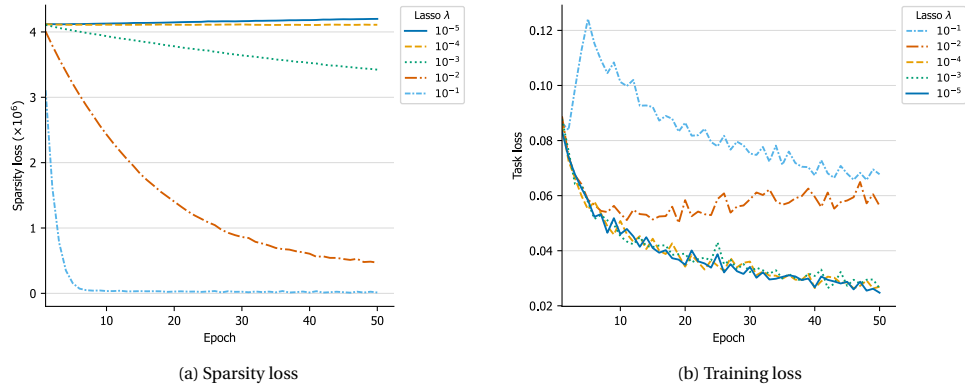


Figure 4.2: Training dynamics of Lasso-penalised finetuning across five penalty strengths λ . (a) The sparsity loss, defined as the summed absolute value of the prunable weights shown in Table 3.1. (b) The training loss is the unweighted sum of the Dice and binary cross-entropy losses.

est penalties converged together near 0.026. They were not separable, whilst the two strongest settled higher, plateauing at 0.0563 for $\lambda = 10^{-2}$ and at 0.0678 for $\lambda = 10^{-1}$; for the strongest penalty, the task loss first rose sharply to a peak of 0.1242 at the fourth epoch before decreasing to its plateau. The penalties that reduced the sparsity loss the most were therefore those that raised the task loss the most.

4.2.2. WEIGHT DISTRIBUTION

The sparsity-task-loss trade-off was reflected in the weight distributions after training, shown per prunable tensor for the shallow, middle and deep encoder blocks at the two extreme penalties in Figure 4.3. The fraction of prunable weights driven exactly to zero by the proximal step of Proximal Gradient Descent (PGD) increased with λ . At $\lambda = 10^{-5}$, no weights reached exactly zero in any tensor, whilst at $\lambda = 10^{-1}$, a substantial fraction was zeroed, seen as the detached bar at the left of each right-hand panel. The zeroed fraction at $\lambda = 10^{-1}$ varied by tensor rather than by block, ranging from 53.0% to 99.3% across the twelve tensors shown, with the attention tensors of the shallow block zeroed most heavily. The effect was therefore present at every depth and was not confined to any one region of the encoder. The distribution of the weights differed between the two penalties: we see a smooth spread with a distinct peak for $\lambda = 10^{-5}$ and a broader, irregular spread for $\lambda = 10^{-1}$. However, neither of these distributions shows a clear bimodal pattern to define a clear pruning threshold. As no such gap was present, the pruning threshold could not be placed by magnitude separation and was instead defined relative to a per-tensor quantile, as described in subsection 3.4.4. The distributions for all twelve blocks at both penalties are provided in Figure A.1 to Figure A.4.

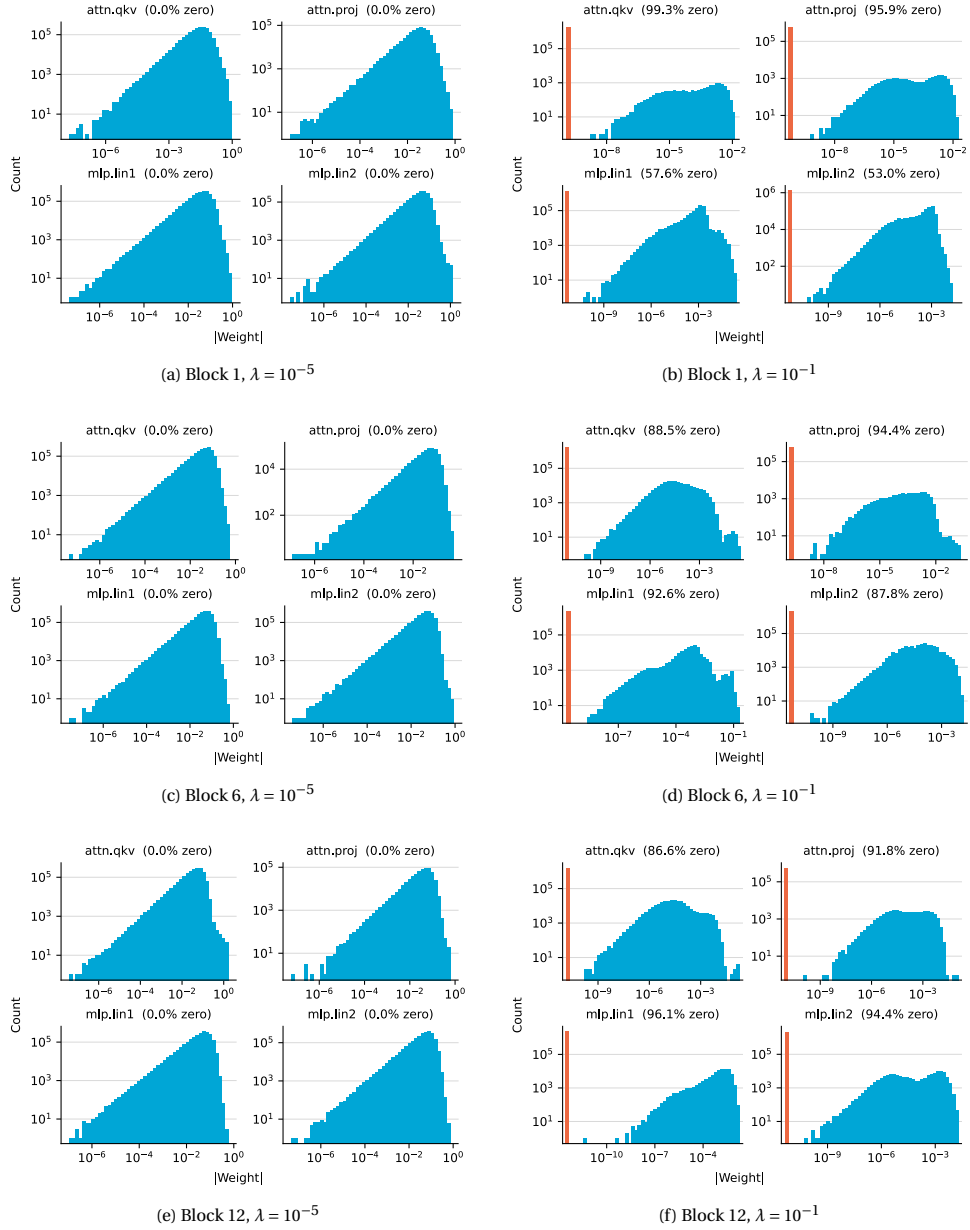


Figure 4.3: Distribution of the absolute prunable weights, shown per tensor (attn.qkv , attn.proj , mlp.lin1 , mlp.lin2), for the shallow (Block 1), middle (Block 6) and deep (Block 12) encoder blocks, at the weakest ($\lambda = 10^{-5}$, left) and strongest ($\lambda = 10^{-1}$, right) Lasso penalties. Surviving weights are shown on a log-magnitude axis; at $\lambda = 10^{-1}$ the detached bar at the left of each panel is the count of weights driven exactly to zero, with the zeroed fraction given in each panel title. No weights reach exactly zero at $\lambda = 10^{-5}$. The distributions for all twelve blocks are provided in [Figure A.1](#) to [Figure A.4](#).

4.2.3. UNSTRUCTURED PRUNING

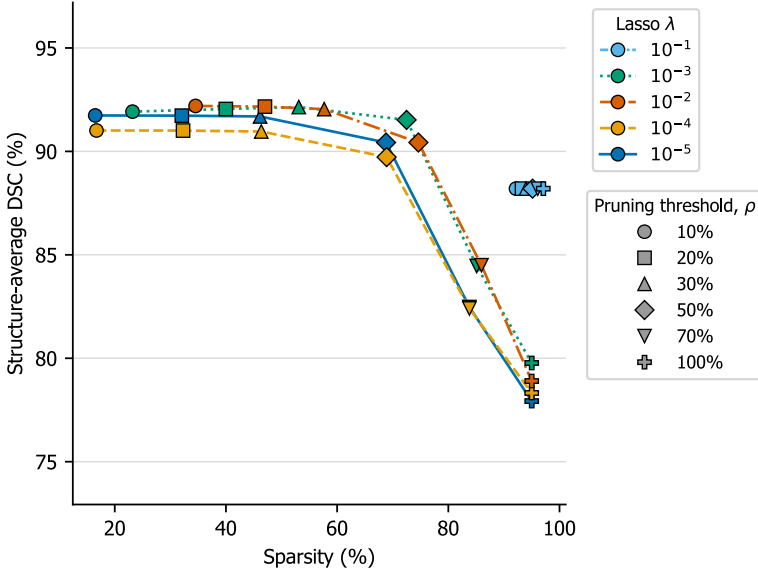


Figure 4.4: Structure-averaged **DSC** against measured sparsity for each combination of Lasso penalty strength λ (colour) and pruning threshold ρ (marker). Structure-averaged **DSC** is the unweighted mean of the eleven per-structure **DSC** values; each per-structure value is the mean over that structure's samples. Sparsity is the fraction of prunable weights zeroed at each threshold.

Applying the per-tensor threshold to these distributions traced out the accuracy against sparsity trade-off in [Figure 4.4](#), shown for each combination of penalty strength λ and pruning threshold ρ . At a fixed threshold, the starting sparsity rose with the penalty strength, so each curve began further to the right: at $\rho = 10\%$ the measured sparsity was 16.49% for $\lambda = 10^{-5}$ and rose through 23.20% at $\lambda = 10^{-3}$ to 92.17% at $\lambda = 10^{-1}$. This ordering follows from the training dynamics, as a stronger penalty had already driven more weight towards zero before any threshold was applied, leaving less for the threshold to set to zero.

The four penalties below $\lambda = 10^{-1}$ shared a common shape, holding a high **DSC** across low and moderate sparsity before falling sharply beyond an elbow. Up to and including $\rho = 50\%$, the **DSC** stayed between 89.72% and 92.20%, declining only slightly as sparsity rose. Beyond the elbow, the decline steepened, the **DSC** falling to between 82.41% and 84.50% at $\rho = 70\%$ and to between 77.93% and 79.77% at $\rho = 100\%$, a total reduction of roughly twelve to fourteen points across the sweep.

The strongest penalty $\lambda = 10^{-1}$ showed no such elbow, as its sparsity already reached 92.17% at the smallest threshold shown and rose only to 97.08% at $\rho = 100\%$, leaving almost no further weight to remove. The **DSC**, therefore, held constant at 88.20% throughout, sitting below the weaker penalties at low sparsity but above them once those had passed their elbow. Balancing this flat high-sparsity response against the higher accuracy of the weaker penalties at moderate sparsity, the operating point was selected at

$\lambda = 10^{-3}$ and $\rho = 50\%$, achieving a sparsity of 72.49% at a structure-averaged DSC of 91.52%.

4.3. SGL REGULARISATION

4.3.1. TRAINING DYNAMICS

The training dynamics of the SGL penalised finetuning are shown only for in Figure A.5 Figure A.9, with each figure fixing λ_1 and sweeping λ_2 . The sparsity loss, defined as the summed absolute value of the prunable weights shown in Table 3.1, decreased over training by an amount that grew with λ_2 , with substantial reductions confined to $\lambda_2 \in \{10^{-2}, 10^{-1}\}$ and little change at weaker group strengths, as seen in Figure A.9. The task loss showed the converse ordering, the stronger group penalties settling higher than the weaker ones, except at $\lambda_1 = 10^{-1}$ (Figure A.5), where all five task loss curves converged together and the reduction in sparsity loss was obtained without a raised task loss.

4.3.2. WEIGHT DISTRIBUTION

Whether a group survived structured pruning was determined by its group L2 norm at the end of training, as a group driven to zero norm carries no weight and is removed without loss. The group norms of the attention-head and MLP-neuron groups are shown for the middle encoder (Block 6) in Figure 4.5, for the three group strengths $\lambda_2 \in \{10^{-1}, 10^{-3}, 10^{-4}\}$ at fixed $\lambda_1 = 10^{-1}$; the corresponding distributions for the shallow and deep blocks are provided in Figure A.10 and Figure A.11 and follow the same pattern. At $\lambda_2 = 10^{-1}$ every attention head was driven to zero norm and the majority of MLP neurons were zeroed, leaving only a small surviving tail, so almost every group became prunable. At $\lambda_2 = 10^{-3}$ the opposite held: no group reached zero norm, the head norms forming a single high-magnitude cluster and the neuron norms a single unimodal mode, with no collapsed groups in either family. Stronger group regularisation therefore produced more zeroed groups, and this ordering was consistent across the shallow, middle and deep encoders. However, the weaker group strength broke that ordering. Although $\lambda_2 = 10^{-4}$ is a weaker regularisation penalty, than $\lambda_2 = 10^{-3}$, it drove a substantial fraction of MLP neurons to zero norm, seen in Figure 4.5 as an orange zero-norm population absent at the stronger $\lambda_2 = 10^{-3}$, and its surviving head norms were smaller in magnitude rather than larger. The effect was present across the shallow, middle and deep blocks alike. The group-norm structure was therefore not monotonic in λ_2 at fixed $\lambda_1 = 10^{-1}$, consistent with the non-monotonic DSC response reported in subsection 4.3.3.

4.3.3. STRUCTURED PRUNING

The mechanics of the structured pruning setup remove whole attention heads and MLP neurons. The sparsity-inducing mechanism of the SGL specific PGD step relies on both the individual (λ_1) and group penalty term (λ_2). Figure 4.6 traces the structured-averaged DSC against the encoder sparsity for a given combination of the two penalties, with λ_1 fixed for each panel.

A response fully insensitive to sparsity occurred in only one region of the sweep, at the strongest individual penalty combined with a strong group penalty. At $\lambda_1 = 10^{-1}$ (Figure 4.6a), the $\lambda_2 = 10^{-1}$ curve held at 89.76% across the entire range, up to the max-

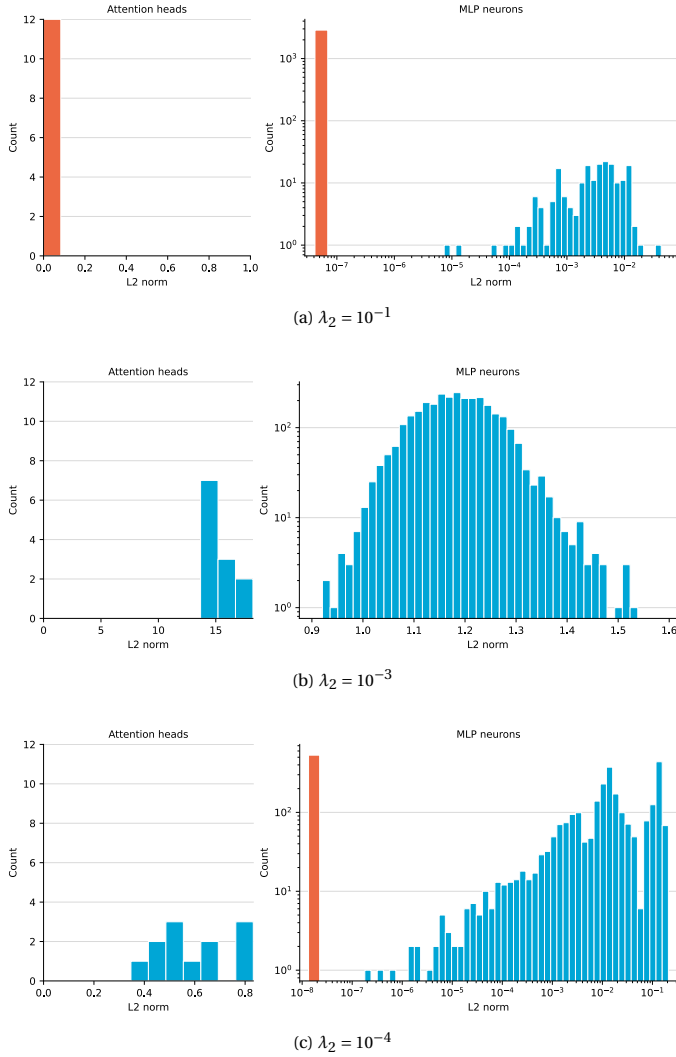


Figure 4.5: Distributions of the group L2 norm for the middle encoder (Block 6) at fixed $\lambda_1 = 10^{-1}$ and the three group strengths $\lambda_2 = 10^{-1}$, 10^{-3} and 10^{-4} . Each panel shows the twelve attention-head groups (left) and the 3072 MLP-neuron groups (right), following the group definitions in Table 3.1. Group norms equal to zero are shown in orange, non-zero norms in blue. The weakest strength $\lambda_2 = 10^{-4}$ zeroed more neuron groups than the stronger $\lambda_2 = 10^{-3}$, for which no group reached zero norm.

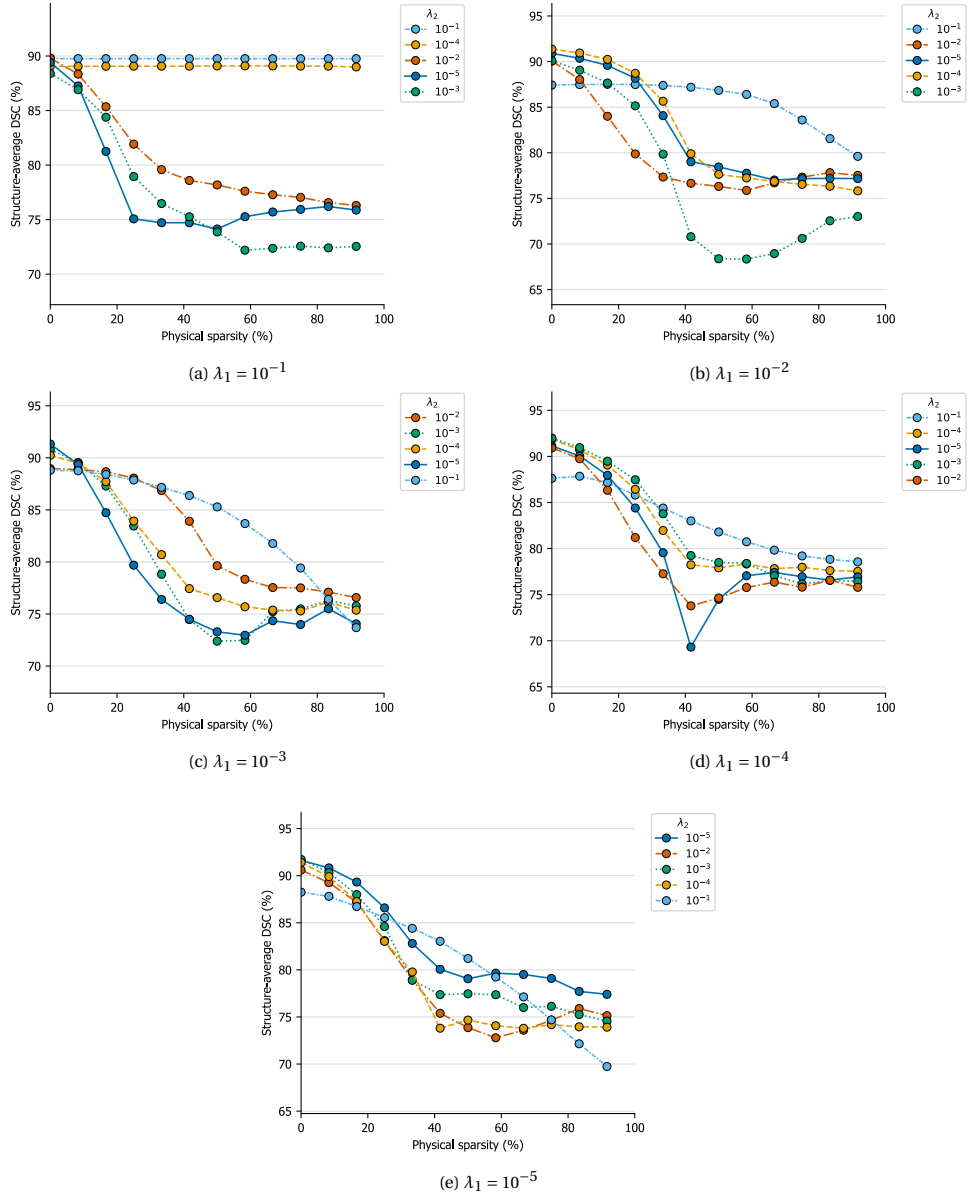


Figure 4.6: Structure-averaged DSC against physical sparsity for Sparse Group Lasso penalised finetuning, with each panel fixing λ_1 and sweeping λ_2 across five values. Physical sparsity is the fraction of prunable weight volume removed by structured pruning of attention heads and MLP neurons, following Table 3.1. Structure-averaged DSC is the unweighted mean of the eleven per-structure DSC values; each per-structure value is the mean over that structure's samples.

imum sparsity of 91.7%, whilst $\lambda_2 = 10^{-4}$ held near 89% over the same range. That the weakest group penalty tracked the strongest one so closely already points to a non-monotonic ordering of λ_2 , since the intermediate $\lambda_2 = 10^{-3}$ and $\lambda_2 = 10^{-2}$ settings, positioned between the two, both collapsed by 50.0% sparsity. This absence of a magnitude pattern across the two penalty terms recurred throughout the ablations, in which the two coefficients did not combine to yield an ordered response. A shared behaviour was nonetheless present across every ablation, with the **DSC** dropping quickly as more heads were pruned and without the distinct elbow seen in the Lasso counterpart (Figure 4.4). Against this common decline, the role of λ_1 became clear, as weakening it removed the flat response of the $\lambda_2 = 10^{-1}$ curve and produced an eventual **DSC** decrease in every panel for which $\lambda_1 < 10^{-1}$. By the weakest individual penalty of $\lambda_1 = 10^{-5}$ (Figure 4.6e) no curve held a **DSC** near its original value beyond 25.0% sparsity, and the collapsed curves settled between roughly 73% and 79% at high sparsity, the lowest endpoint being 69.74% at $\lambda_1 = 10^{-5}$, $\lambda_2 = 10^{-1}$. The selected operating point of $\lambda_1 = 10^{-1}$, $\lambda_2 = 10^{-1}$ sat within the one insensitive region, where the **DSC** held at 89.76% up to the maximum physical sparsity of 91.7%, corresponding to the removal of eleven attention heads per block ($k = 11$).

4.4. REGIME COMPARISON

Three compressed configurations are compared here against the task-loss finetuned model, which serves as the reference throughout. The Lasso operating point ($\lambda = 10^{-3}$, $\rho = 50\%$) and the **SGL** operating point ($\lambda_1 = \lambda_2 = 10^{-1}$, $k = 11$) are those identified in subsection 4.2.3 and subsection 4.3.3. The third configuration discards the **ViT** encoder entirely, retaining only the patch embedding, positional embedding, and neck, and reuses the task-loss-fine-tuned weights of the components it keeps. As all three configurations involve training beyond the pretrained model, the task-loss-finetuned model, rather than the zero-shot model, is used as the reference, so that the comparison isolates the cost of compression from the gain of finetuning. The three configurations separate cleanly in cost, as reported in Table 4.2. Lasso leaves the parameter count, encoder **FLOPs**, and inference time unchanged relative to the reference, as unstructured pruning zeros weights without altering tensor shapes. **SGL** reduces the model from 93.74M to 15.82M parameters, the encoder cost from 966.32 to 81.08 **FLOPs**, and the forward pass from 60.86 to 15.93 ms. Encoder removal reduces these further to 8.59M parameters, 0.60 **FLOPs** and 3.51 ms. The accuracy paid for each saving is the subject of the remainder of this section.

4.4.1. ENCODER-REMOVED VS LASSO VS SGL VS TASK-LOSS FINETUNED

A single ordering held across most of the eleven structures. The Lasso median sat closest to the reference, the **SGL** median below it, and the encoder-removed median furthest below, so accuracy fell as more of the encoder was removed. Two quantities varied on top of this ordering and organise the results that follow: the depth of the encoder-removed drop, which ranged from a few points to over forty, and whether the **SGL** median tracked the Lasso median or broke away towards the encoder-removed floor.

The single-structure datasets followed the baseline ordering with one exception. For Adrenal-ACC-Ki67-Seg, Kvasir-SEG and HC18, the Lasso median decreased slightly (-1.38 ,

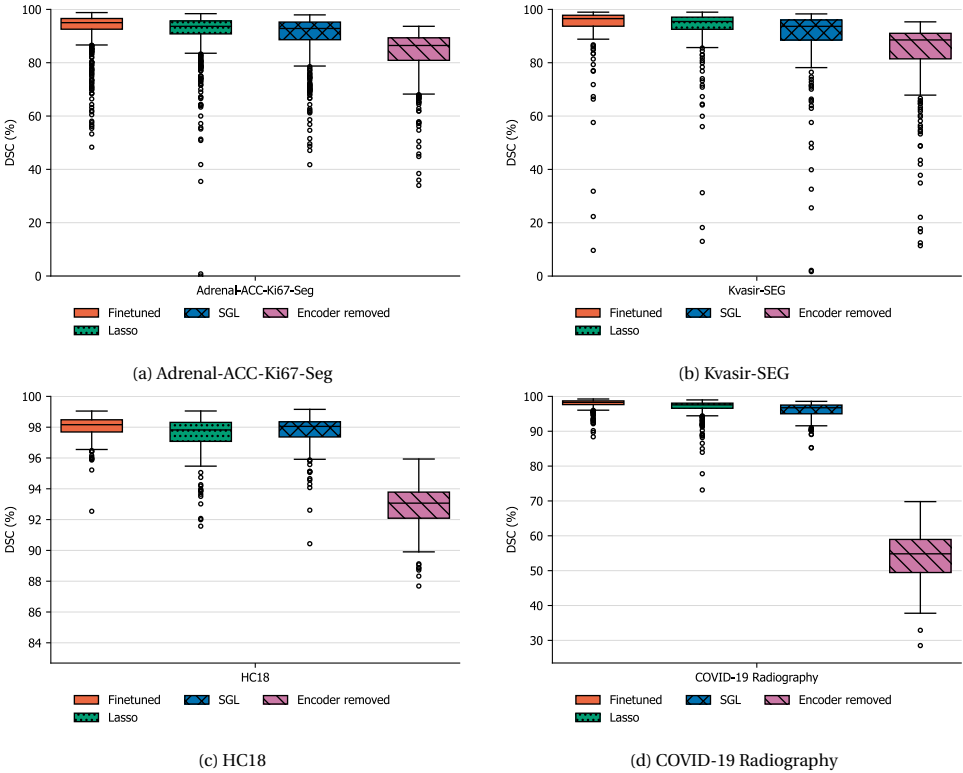


Figure 4.7: Per-sample DSC for the task-loss finetuned, Lasso, SGL and encoder-removed models on the four single-structure datasets. Boxes span Q1 to Q3 with the median marked; whiskers extend to $1.5 \times \text{IQR}$. Moreover, outliers are shown as open circles. Per-panel y-axes are adaptive to show spread; absolute comparison across datasets is directed to Table 4.2.

Dataset	Median DSC (%) (Q1, Q3)			
	Task-loss finetuned	Lasso	SGL	Encoder removed
Adrenal-ACC-Ki67-Seg	95.02 (92.57, 96.59)	93.64 (90.81, 95.68)	92.85 (88.62, 95.25)	86.46 (80.89, 89.34)
Kvasir-SEG	96.55 (93.69, 97.81)	95.46 (92.53, 97.10)	93.64 (88.52, 96.03)	88.55 (81.45, 91.04)
HC18	98.16 (97.69, 98.48)	97.82 (97.08, 98.31)	98.05 (97.36, 98.35)	93.07 (92.09, 93.79)
COVID-19 Radiography	98.29 (97.64, 98.71)	97.57 (96.60, 98.06)	96.74 (95.00, 97.51)	54.84 (49.46, 58.96)
PAPILA (Optic Disc)	96.29 (94.95, 97.31)	96.75 (95.19, 97.63)	96.73 (95.88, 97.29)	92.63 (91.85, 93.27)
PAPILA (Optic Cup)	84.88 (75.01, 88.82)	86.92 (77.78, 90.75)	87.96 (84.68, 90.91)	80.06 (71.74, 85.38)
ACDC (LV)	95.47 (90.34, 97.02)	95.55 (92.62, 96.83)	95.00 (91.39, 96.65)	87.89 (83.49, 91.06)
ACDC (RV)	94.98 (91.49, 96.59)	94.70 (91.15, 96.41)	93.16 (88.82, 95.52)	74.50 (65.84, 80.22)
ACDC (MYO)	89.50 (86.24, 92.09)	89.40 (85.57, 91.94)	86.97 (82.44, 89.86)	45.20 (33.02, 61.92)
GlaS 2015 (testA)	94.04 (90.13, 96.04)	94.30 (91.24, 96.25)	89.12 (83.89, 92.22)	85.40 (80.96, 88.58)
GlaS 2015 (testB)	90.95 (85.05, 95.05)	91.43 (86.44, 94.97)	82.35 (74.75, 89.92)	80.46 (71.28, 86.16)
Inference (ms) (Q1, Q3)	60.86 (60.71, 62.41)	63.49 (63.09, 65.12)	15.93 (15.59, 16.28)	3.51 (2.98, 6.47)
Parameters (M)	93.74		15.82	8.59
Encoder FLOPs (G)	966.32		81.08	0.60

Table 4.2: Median **DSC** per evaluation subset for the task-loss finetuned, Lasso, **SGL** and encoder-removed models, reported as median (Q1, Q3) over per-sample scores from the test set splits defined in Table 3.3. Inference time is reported as the median forward-pass latency (image encoder and mask decoder), pooled across all samples in the eleven subsets. Multi-structure datasets are scored per structure, and multi-split datasets per split, giving eleven entries. The Lasso parameter count and encoder **FLOPs** are unchanged from the task-loss-fine-tuned model, as unstructured pruning zeros weights without altering tensor shapes, and are shown across both columns. Encoder **FLOPs** are calculated from the analytical derivation in subsection 3.2.1.

−1.09 and −0.34), the **SGL** median a little further (−2.17, −2.91 and −0.11), and the encoder-removed median furthest (−8.56, −8.00 and −5.09), so the full drop from removal stayed within nine points. The **IQR** widened under each successive regime in Figure 4.7a, Figure 4.7b and Figure 4.7c, the encoder-removed condition introducing the densest band of low outliers. COVID-19 Radiography was the exception, breaking the ordering only at removal. The Lasso and **SGL** medians stayed close to the reference (−0.72 and −1.55), whilst the encoder-removed median fell to 54.84 (−43.45), the largest single-structure reduction. The box widened accordingly in Figure 4.7d, spanning 49.46 to 58.96 against a reference **IQR** held above 97%.

The two PAPILA structures broke the baseline ordering at the compression penalties themselves, both improving rather than degrading under them. The Optic Disc **DSC** rose under Lasso (+0.46) and **SGL** (+0.44) and fell only on removal (−3.66). The Optic Cup improved further by +2.04 under Lasso and +3.08 under **SGL**, and dropped the least of any structure upon removal (−4.82). Both penalties therefore preserved or improved these medians, in contrast to the reductions seen elsewhere. The spread differed between the two structures in Figure 4.8. The Optic Disc had a narrow **IQR** under all four models, with the encoder-removed box sitting lowest yet remaining the tightest. Whereas the Optic Cup showed a wider **IQR** throughout, and its box contracted progressively from the reference through Lasso to **SGL**, the last yielding the narrowest box and the highest lower quartile, with a band of low outliers between 60% and 75% **DSC**. The encoder-removed Optic Cup box matched the reference in width but shifted downward, with low outliers below 40% **DSC**.

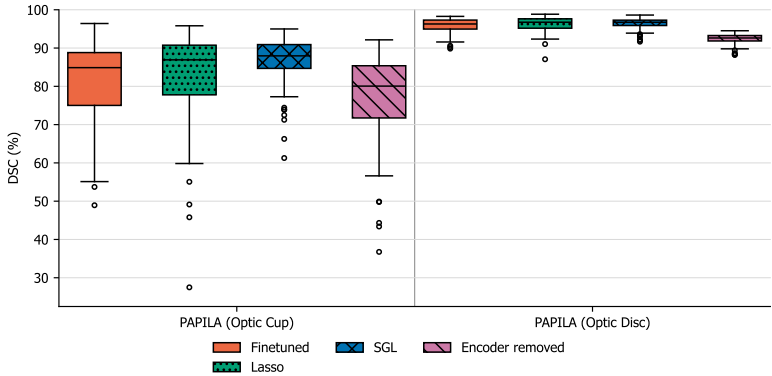


Figure 4.8: Per-sample DSC for the task-loss finetuned, Lasso, SGL and encoder-removed models on PAPILA, reported per structure (Optic Disc, Optic Cup) on a shared y-axis. Boxes span Q1 to Q3, with the median marked; whiskers extend to $1.5 \times \text{IQR}$, and outliers are shown as open circles. Absolute comparison across datasets is directed to Table 4.2.

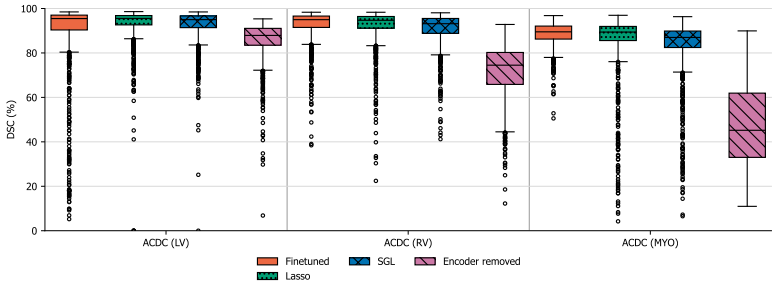


Figure 4.9: Per-sample DSC for the task-loss finetuned, Lasso, SGL and encoder-removed models on ACDC, reported per structure (RV, MYO, LV) on a shared y-axis. Boxes span Q1 to Q3 with the median marked; whiskers extend to $1.5 \times \text{IQR}$, and outliers are shown as open circles. Absolute comparison across datasets is directed to Table 4.2.

The three ACDC structures held the baseline ordering but spanned the widest range of removal costs in the comparison. The **LV** was essentially unchanged under Lasso (+0.08), decreased slightly under **SGL** (−0.47), and fell under removal (−7.58). The **RV** followed the same direction in larger steps (−0.28, −1.82, and −20.48). The **MYO** held under Lasso (−0.10), decreased under **SGL** (−2.53), and fell to 45.20 under removal (−44.30), the steepest drop of the three. The spread tracked the medians in Figure 4.9. All three boxes stayed tight and high under the reference, Lasso, and **SGL** models, each carrying a dense, low-tail heaviest for the **LV**, and widened upon removal in proportion to the median drop. The encoder-removed **LV** box widened moderately whilst holding above 80%, the **RV** box widened further with a dense band of low outliers below 45% **DSC**, and the **MYO** box spanned roughly 33% to 62% **DSC**, the widest in the comparison, with its lower whisker near 10%.

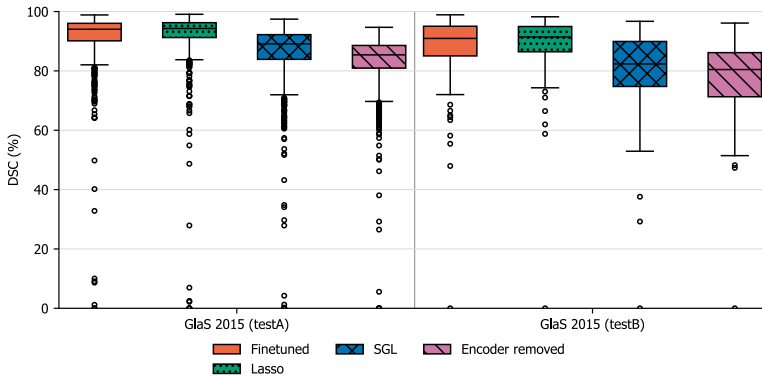


Figure 4.10: Per-sample **DSC** for the task-loss finetuned, Lasso, **SGL** and encoder-removed models on GlaS 2015, reported per split (testA, testB) on a shared y-axis. Boxes span Q1 to Q3 with the median marked; whiskers extend to $1.5 \times \text{IQR}$, and outliers are shown as open circles. Absolute comparison across datasets is directed to Table 4.2.

The two GlaS splits broke the second pattern, the one place where **SGL** left the Lasso median and fell towards the removal floor. On GlaS(testA), the Lasso median rose slightly (+0.26) whilst the **SGL** median dropped (−4.92) and the encoder-removed median fell furthest (−8.64). GlaS(testB) repeated this, with Lasso at +0.48, **SGL** at −8.60, and removal at −10.49. On both splits, the **SGL** reduction therefore approached the encoder-removed reduction, unlike every other dataset, where **SGL** stayed near Lasso. The spread in Figure 4.10 followed the same split. The Lasso box stayed high and tight on both splits, comparable to the reference, whilst the **SGL** box dropped and widened to sit close to the encoder-removed box in centre and spread, the two separating only marginally on GlaS(testB), where the **SGL** box spanned roughly 75% to 90% **DSC**. Extreme outliers near 0% **DSC** persisted under all four models on both splits.

4.4.2. QUALITATIVE BEHAVIOUR OF THE PREDICTIONS

The per-sample scores are accompanied by the predicted masks they were computed from, shown for one representative sample per structure in Figure 4.11, with the full set

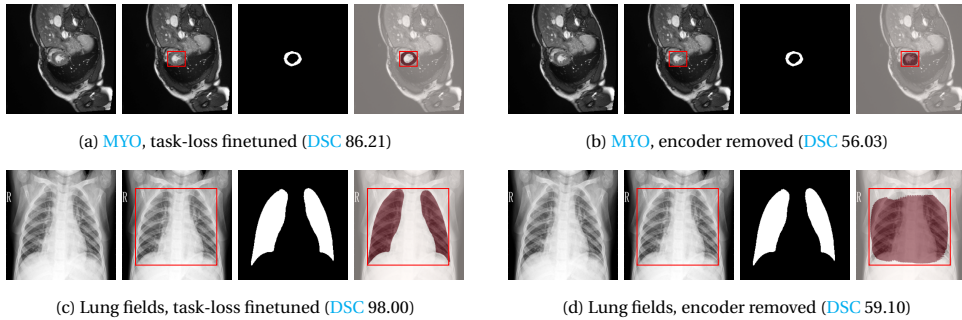


Figure 4.11: Predicted masks for the ACDC MYO and the COVID-19 Radiography lung fields under the task-loss finetuned and encoder-removed models. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The DSC of the displayed sample is reported in each subcaption. The Lasso and SGL predictions, together with the full set of eleven structures, are provided in Figure A.12 to Figure A.22.

of eleven provided in Figure A.12 to Figure A.22. Each panel gives the input image, the bounding-box prompt, the ground-truth mask, and the predicted mask overlaid within the prompt, under the four regimes. The four regimes produced visibly different masks. Under the task-loss finetuned, Lasso and SGL models, the predicted mask followed the target outline in all three cases shown, including the hollow interior of the MYO ring and the separation between the two lung fields of COVID-19 Radiography. Whilst the encoder-removed model produced a visibly different prediction that instead filled the bounding box with a single convex region. Where the target was itself compact and convex, as for the Adrenal-ACC-Ki67-Seg carcinoma, this region matched it closely. Where the target was hollow or multi-part, the filled region departed from it: the MYO ring was filled solid, and the two lung fields were merged into one mass spanning the mediastinal gap. The SGL prediction, provided with the full set in Figure A.12 to Figure A.22, retained these shapes at 91.7% physical encoder sparsity, holding the RV crescent, the hollow MYO ring and the separated lung fields. On GlaS(testA), the SGL prediction tracked the glandular boundary more loosely than on the other structures, following its outline only in part. Across the eleven structures, the compression cost was therefore neither uniform nor fully ordered by regime. SGL matched the Lasso median on nine of the eleven and approached the encoder-removed floor only on the two GlaS splits. Encoder removal fell within nine points of the reference on seven structures, yet exceeded forty points on the MYO and COVID-19 Radiography. The accuracy retained at each level of cost thus depended on the structure being segmented, and the per-structure basis of that dependence is examined in section 5.3.

5

DISCUSSION

5.1. MECHANISM OF THE PROXIMAL OPERATORS

This chapter interprets the results of [chapter 4](#) through the proximal operators that produced them. Both penalties exist to trade sparsity against accuracy. The shape of that trade-off follows from how each operator acts on the weights.

5.1.1. ELEMENT-WISE AND GROUP REMOVAL

The Lasso penalty applies the element-wise soft-threshold of [Equation 2.7](#). Each weight is shrunk toward zero and removed once it falls below the threshold. The operator acts on every weight independently, so no weight is shrunk relative to any other. It therefore produces no separated high-magnitude mode. The weight distributions in [Figure A.1](#) and [Figure A.3](#) confirm this, forming a single mode at small magnitudes at every penalty strength. The absence of a gap is why the pruning threshold had to be set at a per-tensor quantile in [subsection 3.1.2](#). A distribution without a gap cannot be cut by magnitude separation.

Unstructured pruning zeroes individual weights but leaves the tensor shapes unchanged. As a result, no physical speed-up is realised, but the achieved sparsity provides an indication of the model's theoretical redundancy. Removing scattered low-magnitude weights degraded the output gradually rather than removing any unit outright. For the weaker penalties, this held the [DSC](#) between 89.72% and 92.20% up to the 70% threshold in [Figure 4.4](#). Beyond that point, the [DSC](#) declined sharply, falling to between 77.93% and 79.77% at $\rho = 100\%$, as the threshold reached the bulk of the weight population. The strongly regularised $\lambda_1 = 10^{-1}$ was the sole exception, as the model had enough induced sparsity that the threshold made no difference to its performance. For the other cases, the gradual degradation indicates that a large fraction of individual weights each carried little of the segmentation performance, which is the parameter redundancy quantified in [section 5.2](#).

The [SGL](#) penalty adds a group stage that is applied in a fixed order. The operator of [Equation 3.2](#) applies the element-wise soft-threshold first. It then scales each group

toward zero as a whole and removes the group once its thresholded norm falls below the group threshold. A group is therefore removed on the strength that remains after the element-wise term has eroded it. Its survival depends on both λ_1 and λ_2 , not on λ_2 alone. This coupling is examined in [subsection 5.1.2](#).

The group penalty induced a steeper trade-off for [SGL](#). Structured pruning removes entire heads and neurons, then rebuilds the tensors with a smaller width. The computation of those units is not thinned, but severed significantly. Accuracy, therefore, falls faster with sparsity than under Lasso. This is reflected in the consistently lower [DSC](#) of [SGL](#) at matched effort in [Table 4.2](#). The reward for the steeper cost is a genuinely smaller and faster model, with parameter counts dropping from 93.74M to 15.82M, [GFLOPs](#) dropping from 966.32 to 81.08, and inference time dropping from 60.86 to 15.93 ms. Lasso leaves the tensor shapes and the computational cost unchanged. The two operators, therefore, sit at opposite ends of the trade-off. Lasso induces sparsity cheaply but yields no physical saving, whilst [SGL](#) yields a real saving at a steeper cost in accuracy.

5.1.2. INTERACTION OF THE PENALTY STRENGTHS

For the structured pruning experiments, the penalty coefficients did not act independently in the sweeps. The element-wise term sets a dead zone, the range within which a weight is zeroed, and a larger strength widens it. The group term then acts only on what the dead zone leaves behind. The element-wise strength, therefore, has a downstream effect on the group strength, and not the reverse.

This is visible first in the training dynamics. In [Figure 4.2a](#), the reduction in sparsity loss grew with λ , and at $\lambda = 10^{-1}$ the loss reached its minimum within the first five epochs. The weights were therefore driven to zero during finetuning, before any pruning was applied. The effect on pruning follows in [Figure 4.4](#). The sparsity of the encoder when $\lambda = 10^{-1}$ already exceeded 92% even at the smallest threshold, with the [DSC](#) held flat at 88.20% across the full range. Having such a sparse model already meant raising the threshold removed very little further. Whilst the weaker penalties resulted in a denser weight distribution, holding a higher [DSC](#) at low thresholds, the network still used weights, collapsing once the threshold cut its weights. A wide dead zone, therefore, trades peak accuracy for a slightly lower, flatter, and more predictable response, whilst a narrow one retains accuracy until the threshold forces removal.

The same dependence is seen in the [SGL](#) sweep. The group term removes a group once its norm falls below the group threshold, but the group norm is itself highly dependent on the element-wise term. When λ_1 is large, the groups are small before the group term acts. A modest λ_2 is then enough to remove them, and a larger one changes little. When λ_1 is small, the groups keep their strength, and λ_2 must act against their larger norms to have any effect.

The training dynamics show this coupling before pruning is applied. At $\lambda_1 = 10^{-1}$ ([Figure A.5](#)), the five task-loss curves converged, so the sparsity was obtained without a raised task loss. The wide dead zone had already reduced the groups, and the group term completed their removal at little cost. As λ_1 was reduced, the group penalties settled at increasingly separated task-loss values ([Figure A.9](#)), as the group term now acted against groups that retained their strength.

The structured sweep of [Figure 4.6](#) shows the effect on segmentation accuracy. At

$\lambda_1 = 10^{-1}$ (Figure 4.6a) the strongest group penalty maintained a flat DSC of 89.76% across all sparsity values. Reducing λ_1 removed the flat response. At $\lambda_1 = 10^{-2}$ (Figure 4.6b) every curve declined with sparsity, and at $\lambda_1 = 10^{-5}$ (Figure 4.6e) no curve held its value beyond 25.0% sparsity. With a narrow dead zone, the groups maintained their strength, so structured removal still reduced the network's computation, and accuracy fell regardless of λ_2 .

The ordering by λ_2 was not preserved at fixed $\lambda_1 = 10^{-1}$, as noted in subsection 4.3.2. The flat response was held by both $\lambda_2 = 10^{-1}$ and $\lambda_2 = 10^{-4}$, whilst the intermediate $\lambda_2 = 10^{-3}$ retained intact groups, so a weaker group penalty produced more collapse than a stronger one. This is not expected from the group term in isolation, under which collapse should grow with λ_2 . The result could be attributed here to the way the models were produced. Each configuration is a separate finetuning run, and its checkpoint is selected by validation DSC at an epoch chosen independently for that run. The group-norm distributions in Figure A.10 to Figure A.11 are therefore snapshots of three different models, taken at three different points along three different trajectories. They are not one model observed under three final penalties, so an ordering by λ_2 need not hold. A single training run was performed per configuration, so the anomaly is reported as one realisation rather than a characterised trend. The two coefficients must nonetheless be read together, which is the reason the sweep was conducted over the full grid rather than along a single combined strength, as motivated in subsection 3.4.4.

5.2. THE EXTENT OF REDUNDANCY

The two regimes measure redundancy in two different senses. Lasso measures parameter redundancy, the fraction of individual weights that can be zeroed while the architecture is retained. SGL measures computational redundancy, the fraction of weight volume that can be removed as whole heads and neurons. The two are not directly comparable, as they remove different things, but together they bound the extent to which the encoder is unnecessary for the seven tasks.

Parameter redundancy was high. At the operating point, Lasso reached a sparsity of 72.49% of the prunable encoder weight volume at a structure-averaged DSC of 91.52%, so most individual prunable weights could be zeroed at a small cost in accuracy. This buys no physical savings, however. The tensor shapes are unchanged, so the parameter count, the FLOPs, and the inference time in Table 4.2 are identical to those of the task-loss-fine-tuned model. The redundancy is real but latent, present in the weights without reducing the cost of running the model.

Computational redundancy was higher still, and it was realisable. At the operating point, SGL removed 91.7% of the prunable encoder weight volume as whole units. As the prunable tensors constitute most but not all of the model, this reduced the full parameter count from 93.74M to 15.82M, the encoder FLOPs from 966.32 to 81.08, and the inference time from 60.86 to 15.93 ms. This removal held a structure-averaged DSC of 89.76%, so a large fraction of the encoder's computation could be discarded while largely retaining accuracy.

The two figures diverge because they answer different questions. Lasso can zero a weight wherever one sits below the threshold, whereas SGL can remove a unit only when its entire group is weak. More redundancy is therefore reachable at the level of individual

parameters than at the level of whole units, which is the mechanism set out in [subsection 5.1.1](#). The practical consequence reverses this ordering. The parameter redundancy that Lasso exposes cannot be converted into a smaller or faster model, whilst the computational redundancy that SGL exposes can. The encoder is not fully redundant, however. Removing it entirely reduced the encoder FLOPs to 0.60 but also caused a large loss of accuracy, most severely on COVID-19 Radiography, where the median DSC fell to 54.84% in [Table 4.2](#). The encoder, therefore, carries substantial redundancy that can be removed at little cost, up to a point beyond which the remaining computation is necessary. That point is dataset-dependent, and its per-dataset cost is examined in [section 5.3](#).

5.3. DEPENDENCE ON IMAGING MODALITY

The cost of compression was not uniform across the eleven structures. Some retained reasonable accuracy even when the encoder was removed entirely, whilst others collapsed. The per-dataset cost indicates how much of the encoder's capacity each structure actually used, since a structure that survived removal was served by capacity it did not need. In contrast, one that collapsed relied on the removed computation. The structures fall into three groups.

The first group was little affected by the removal of the encoder. HC18 and the PA-PILA Optic Disc remained above 92% DSC under every regime, including encoder removal, at 93.07% and 92.63%, respectively. The Optic Cup sits with them, holding 80.06% once the encoder is removed, and, in fact, improving under both penalties. For these structures, the encoder held capacity for tasks that were never called on, so losing it cost almost nothing.

The second group came apart more slowly. Adrenal-ACC-Ki67-Seg, Kvasir-SEG, and the ACDC LV followed the task-loss-finetuned model closely under Lasso and SGL, then decreased to between 80% and 89% once the entire encoder was removed. The penalties removed mostly spare capacity, but the encoder as a whole was still doing work, and removing it made that work visible.

The third group could not do without the encoder at all. COVID-19 Radiography fell to 54.84% under encoder removal, and the ACDC MYO and RV to 45.20% and 74.50%, the three steepest drops in the comparison. Here, the encoder's computation was important to the task, and removing it degraded segmentation performance.

Two structures stand apart from the three groups, each for a different reason. The Optic Cup is the clearest single case, as it improved under compression rather than merely surviving it. The DSC increased by +2.04 under Lasso and +3.08 under SGL relative to the task-loss finetuned model, and the SGL box was the narrowest and highest among the four regimes in [Figure 4.8](#). This is consistent with the finetuning result of [subsection 4.1.1](#), where task-loss finetuning reduced the Optic Cup median by -6.40 from the zero-shot model, the sharpest drop of any structure. Finetuning, therefore, harmed this structure, and the sparsity penalty recovered part of that loss. The penalty acts as a regularisation control, and reducing the encoder's capacity suited a structure that the dense model had fit poorly.

The GlaS splits are the opposite case, the one dataset on which SGL costs far more than Lasso. The SGL median fell by -4.92 on testA and -8.60 on testB, approaching the encoder-removed reductions of -8.64 and -10.49, whilst Lasso held at or above the

task-loss finetuned model. The gap between the two penalties was wider here than for any other dataset. The computation removed by structured pruning was not spared, so whole-unit removal was punished while weight-level removal was not. This marks GlaS as a dataset where the encoder's full capacity is necessary for the task at hand.

Taken together, these cases account for the variation in SGL's behaviour across the datasets. Its cost was low on the Optic Cup, moderate on the second group, and severe on GlaS, not because the penalty behaved differently, but because the computation it removed mattered by different amounts.

5.3.1. QUALITATIVE BEHAVIOUR OF THE PREDICTIONS

The predicted masks described in subsection 4.4.2 account for the failure modes behind the quantitative tiers. With the encoder removed, the prediction collapsed into a convex region that filled the bounding box, as the model could no longer resolve structure within the prompt. This was harmless when the target was itself compact and convex, so the Adrenal-ACC-Ki67-Seg carcinoma (Figure A.15) and the PAPILA Optic Disc (Figure A.17) both survived removal. It was destructive when the target was hollow, thin or multi-part. The MYO and RV rings were filled solid (Figure A.13, Figure A.12), and the two lung fields of COVID-19 Radiography were merged into one mass across the mediastinal gap (Figure A.22). These collapses match the tiers of Table 4.2.

The SGL model kept the encoder intact at 91.7% physical sparsity of the prunable encoder weight volume and recovered these shapes, preserving the RV crescent, the hollow MYO ring and the separated lung fields, which agrees with its near-baseline DSC on these structures. The contrast identifies the encoder's role as resolving shape within the prompt, an ability that structured removal retains, and encoder removal does not. The recovery was weakest on GlaS (Figure A.19), where the target is a thin, texture-defined glandular boundary rather than a smooth closed contour, so the SGL prediction tracked it only loosely. This agrees with GlaS being the dataset for which structured pruning incurred the highest cost.

5.4. LIMITATIONS

Several limitations bound the claims of this work. They are set out here so that the findings are read in the right scope. The measured redundancy is for these tasks. All results were obtained on seven datasets under bounding-box prompting. A head or neuron that was spare here may still hold capabilities that MedSAM needs for other modalities, structures, or prompt types. The redundancy is therefore relative to the evaluation, not a fixed property of the model. The reported medians hold, but the tails do not. Under compression, the low outliers grew denser and reached further, as seen in the box plots of section 4.4. A model that holds its median can still fail on individual cases. In a clinical setting, the tail is where the risk lies, so a preserved median is not, on its own, a guarantee of safe behaviour. The study covers one model and seven datasets. This is enough to characterise the patterns reported, but not to prove them in general. The tiers and the mechanisms are offered as interpretation supported by the evidence, not as established law. Each configuration was trained once. A single seed was used per penalty setting, so run-to-run variation was not measured. The non-monotonic case in subsection 4.3.2 is

one consequence of this, and other results may carry a similar variance that a single run cannot separate. The qualitative comparison rests on single examples. The overlays in [subsection 5.3.1](#) illustrate the failure modes but were not quantified across the test set. They support the mechanism rather than measure it.

6

CONCLUSION

This thesis examined how much of the [MedSAM](#) image encoder is redundant for medical image segmentation, and whether the cost of removing that redundancy depends on the imaging modality being segmented. The three sub-questions are answered in turn.

The first asked how much of the encoder is redundant, and in what sense. The answer depends on the sense. Weight by weight, through the Lasso penalty and unstructured pruning, 72.49% of the prunable encoder weights were zeroed. As whole units, through the [SGL](#) penalty and structured pruning, 91.7% of the prunable encoder weight volume was removed as attention heads and [MLP](#) neurons. The two senses are not equivalent in value. The weight-level redundancy is latent, as the tensor shapes remain unchanged and nothing is saved. The unit-level redundancy is realisable, reducing the model from 93.74 M to 15.82 M parameters, the encoder cost from 966.32 to 81.08 GFLOPs, and the forward pass from 60.86 to 15.93 ms.

The second question asked whether the cost of this removal depends on the imaging modality. It does. With the encoder fully removed, the eleven structures fell into three tiers. Some retained usable accuracy without an encoder, such as the foetal head and the optic disc, both above 92% [DSC](#). Others held under both penalties, then dropped once the encoder was removed. A third was heavily dependent on the encoder, most notably the [CXR](#) viral pneumonia lung fields, whose median [DSC](#) dropped to 54.84%. The predicted masks show that segmentation performance was retained for compact and convex targets but lost for hollow, thin, or multi-part ones. The cost of removal is therefore governed by the shape of the target structure.

The third asked how far the encoder can be reduced whilst retaining respectable accuracy. Structured pruning removed most of its computation whilst holding within three points of the finetuned reference, yielding a model roughly six times smaller and four times faster. Removing the encoder entirely went too far, as the resulting accuracy loss was substantial and varied by modality. The reduction is bounded not by a single sparsity figure, but by the point at which the remaining computation becomes necessary, which differs across modalities. The [MedSAM](#) encoder therefore carries substantial redundancy for these seven tasks. Much of it can be removed to achieve real savings in size

and speed, though the extent depends on the modality being segmented.

A

APPENDIX

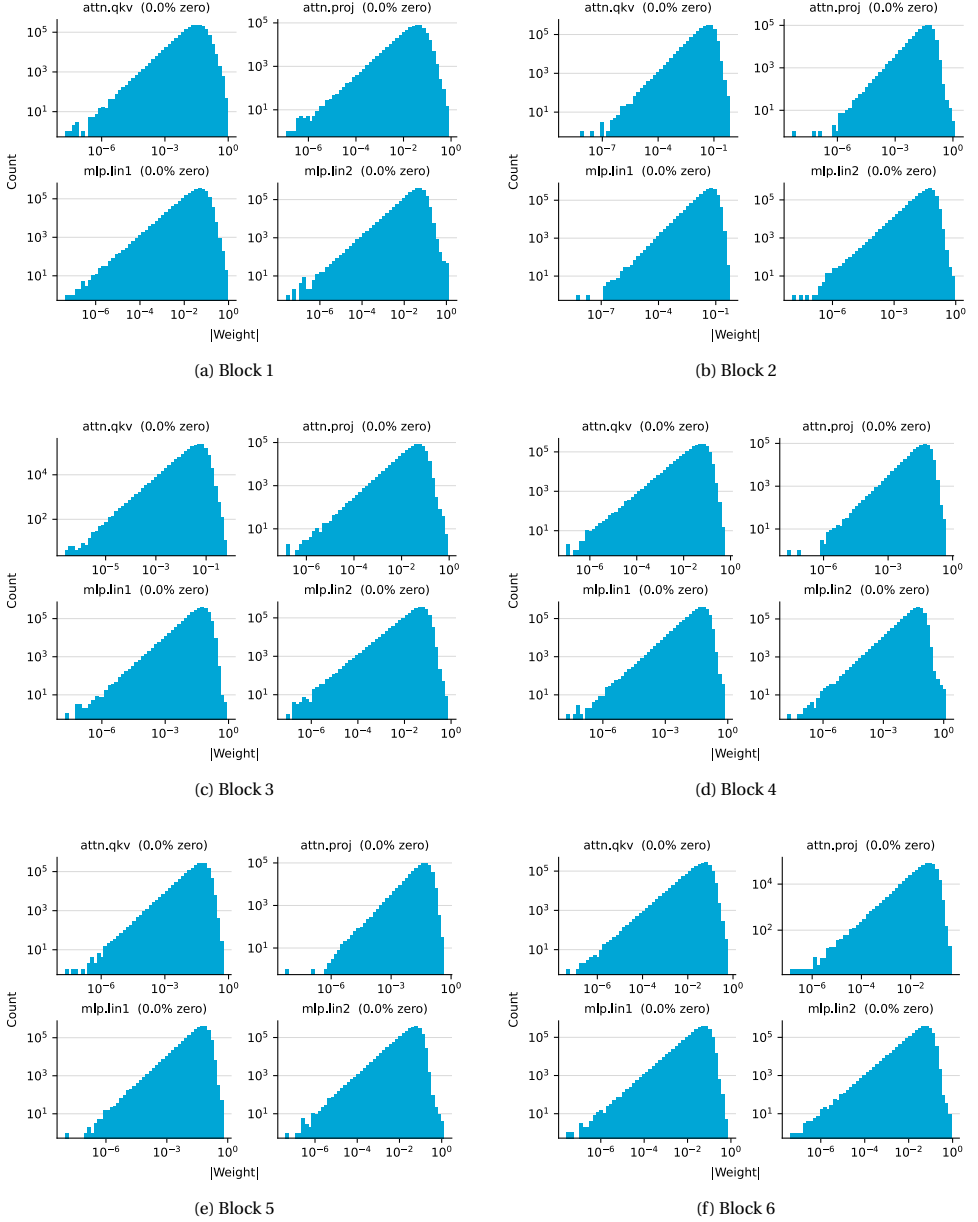


Figure A.1: Distribution of the absolute prunable weights at $\lambda = 10^{-5}$ for blocks 1 to 6. No weights are driven exactly to zero at this penalty strength. Weights are shown on a log-magnitude axis; the zeroed fraction is given in each panel title.

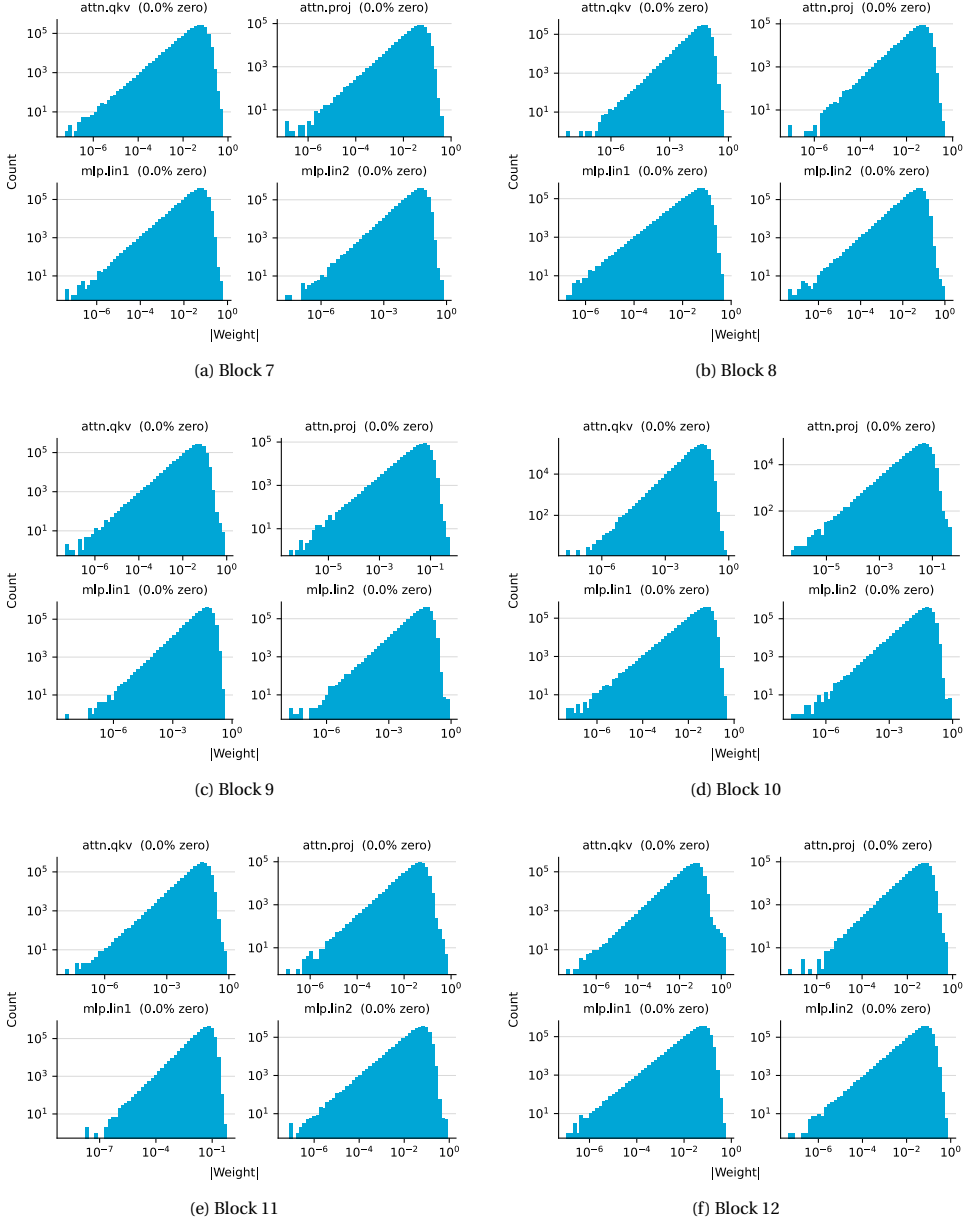


Figure A.2: Distribution of the absolute prunable weights at $\lambda = 10^{-5}$ for blocks 7 to 12. No weights are driven exactly to zero at this penalty strength. Weights are shown on a log-magnitude axis; the zeroed fraction is given in each panel title.

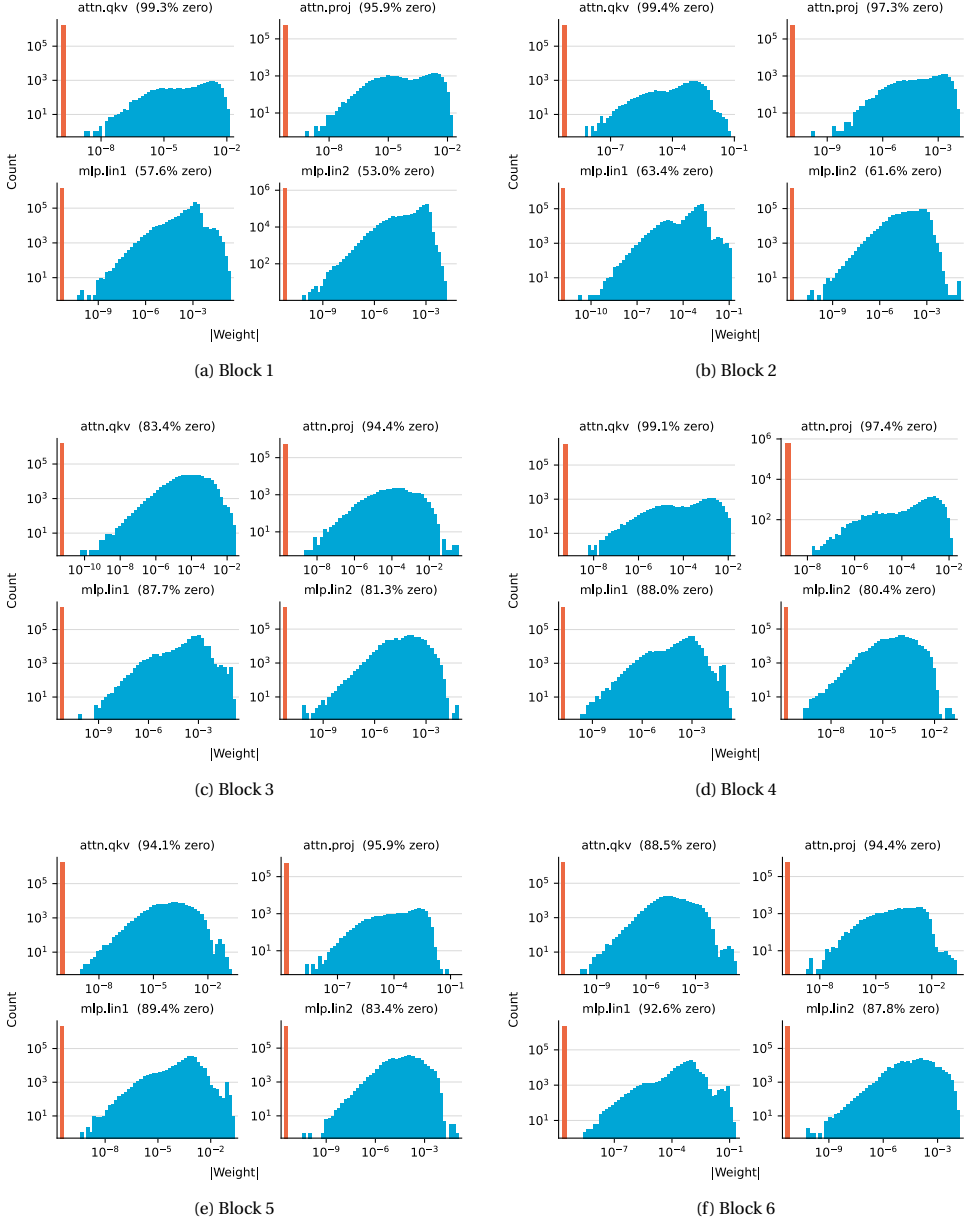


Figure A.3: Distribution of the absolute prunable weights at $\lambda = 10^{-1}$ for blocks 1 to 6. Surviving weights are shown on a log-magnitude axis; the detached bar at the left of each panel is the count of weights driven exactly to zero, with the zeroed fraction given in each panel title.

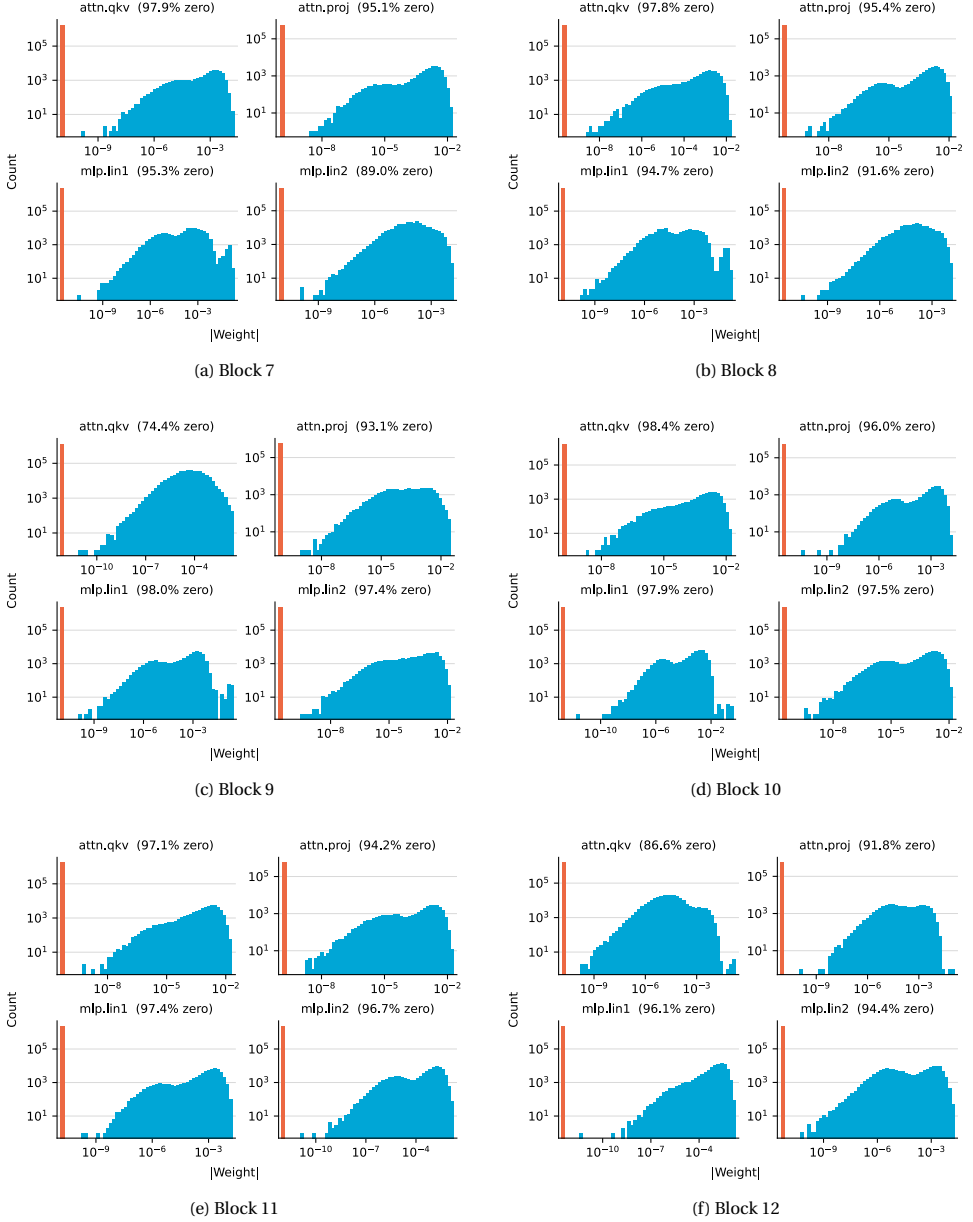


Figure A.4: Distribution of the absolute prunable weights at $\lambda = 10^{-1}$ for blocks 7 to 12. Surviving weights are shown on a log-magnitude axis; the detached bar at the left of each panel is the count of weights driven exactly to zero, with the zeroed fraction given in each panel title.

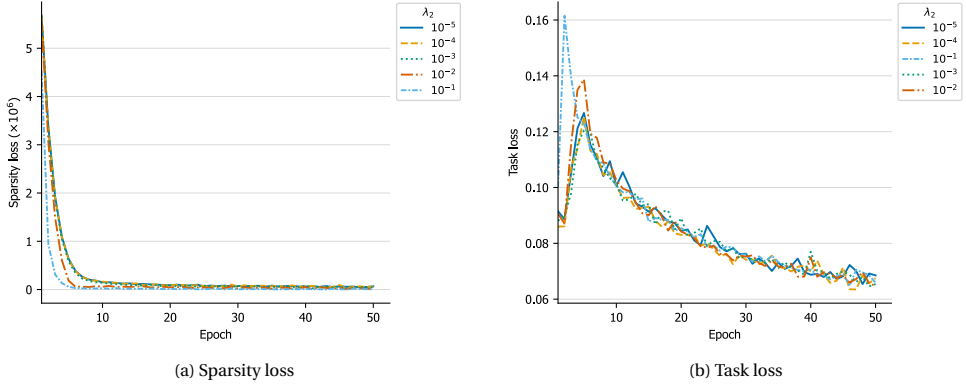


Figure A.5: Training dynamics of Sparse Group Lasso penalised finetuning at $\lambda_1 = 10^{-1}$, sweeping λ_2 across five values. (a) The sparsity loss, defined as the summed absolute value of the prunable weights shown in Table 3.1. (b) The task loss, the unweighted sum of the Dice and binary cross-entropy losses.

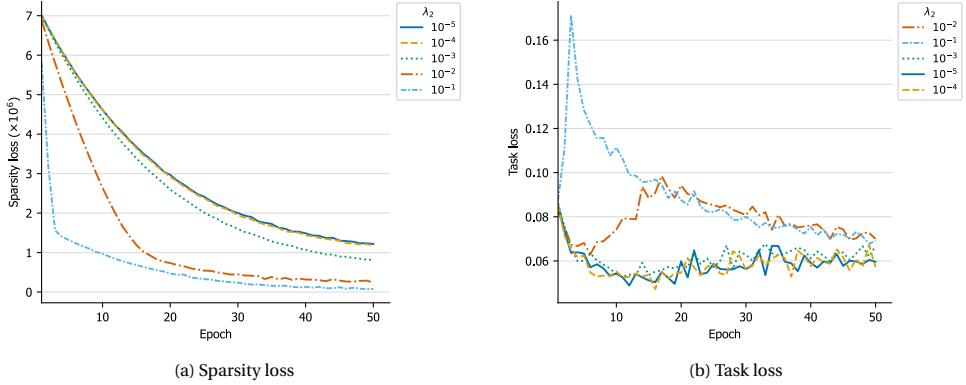


Figure A.6: Training dynamics of Sparse Group Lasso penalised finetuning at $\lambda_1 = 10^{-2}$, sweeping λ_2 across five values. (a) The sparsity loss, defined as the summed absolute value of the prunable weights shown in Table 3.1. (b) The task loss, the unweighted sum of the Dice and binary cross-entropy losses.

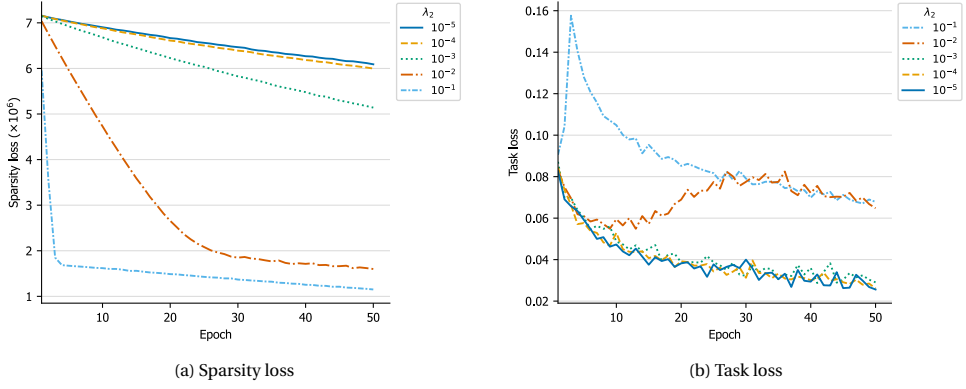


Figure A.7: Training dynamics of Sparse Group Lasso penalised finetuning at $\lambda_1 = 10^{-3}$, sweeping λ_2 across five values. (a) The sparsity loss, defined as the summed absolute value of the prunable weights shown in Table 3.1. (b) The task loss, the unweighted sum of the Dice and binary cross-entropy losses.

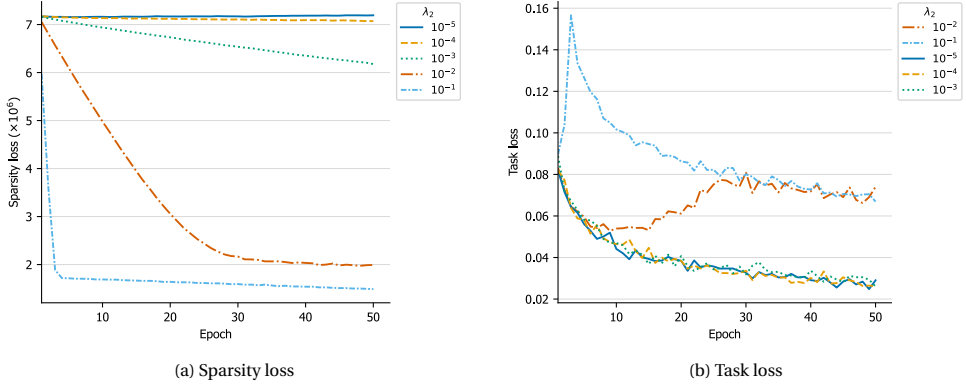


Figure A.8: Training dynamics of Sparse Group Lasso penalised finetuning at $\lambda_1 = 10^{-4}$, sweeping λ_2 across five values. (a) The sparsity loss, defined as the summed absolute value of the prunable weights shown in Table 3.1. (b) The task loss, the unweighted sum of the Dice and binary cross-entropy losses.

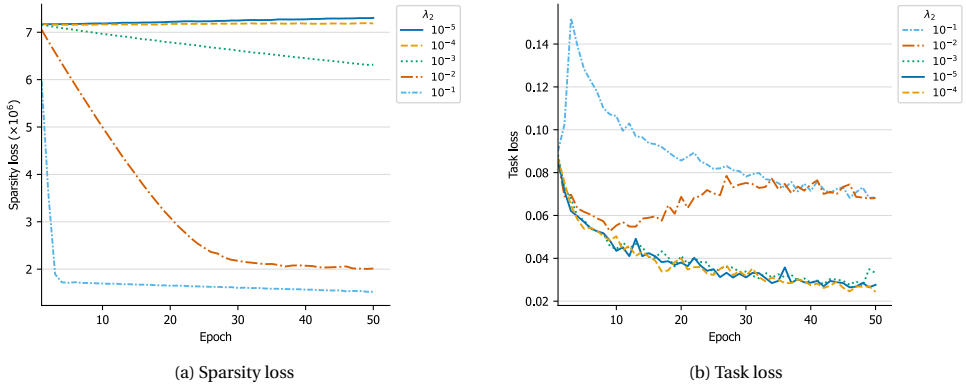


Figure A.9: Training dynamics of Sparse Group Lasso penalised finetuning at $\lambda_1 = 10^{-5}$, sweeping λ_2 across five values. (a) The sparsity loss, defined as the summed absolute value of the prunable weights shown in Table 3.1. (b) The task loss, the unweighted sum of the Dice and binary cross-entropy losses.

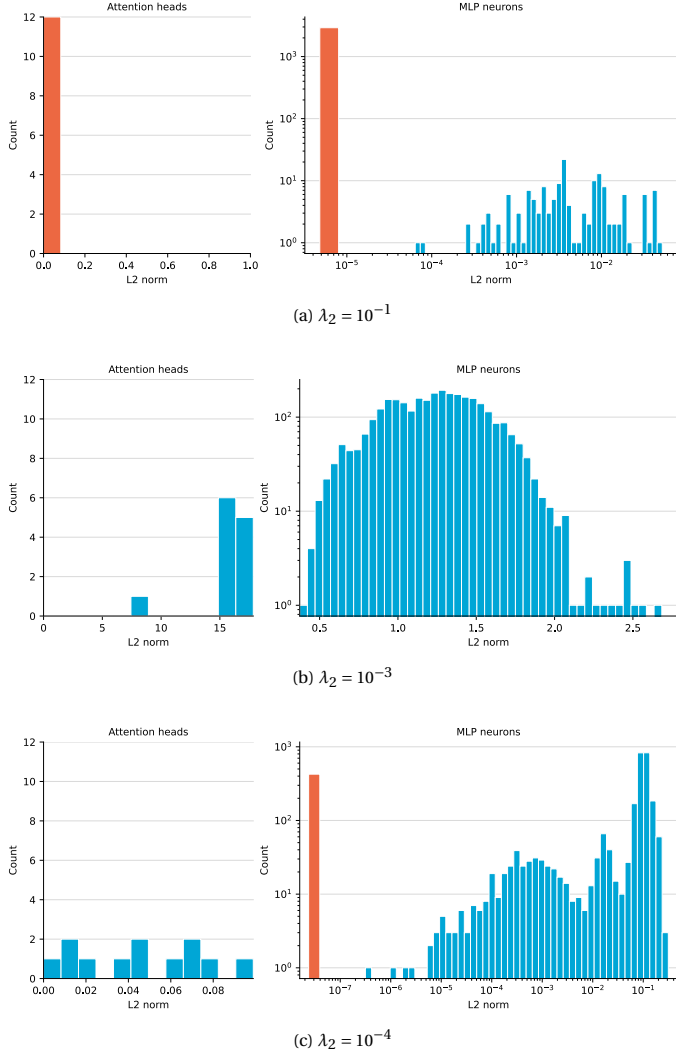


Figure A.10: Distributions of the group L2 norm for the shallow encoder (Block 1) at fixed $\lambda_1 = 10^{-1}$ and the three group strengths $\lambda_2 = 10^{-1}$, 10^{-3} and 10^{-4} . Each panel shows the twelve attention-head groups (left) and the 3072 MLP-neuron groups (right), following the group definitions in Table 3.1. Group norms equal to zero are shown in orange, non-zero norms in blue. The middle-block distributions are shown in Figure 4.5.

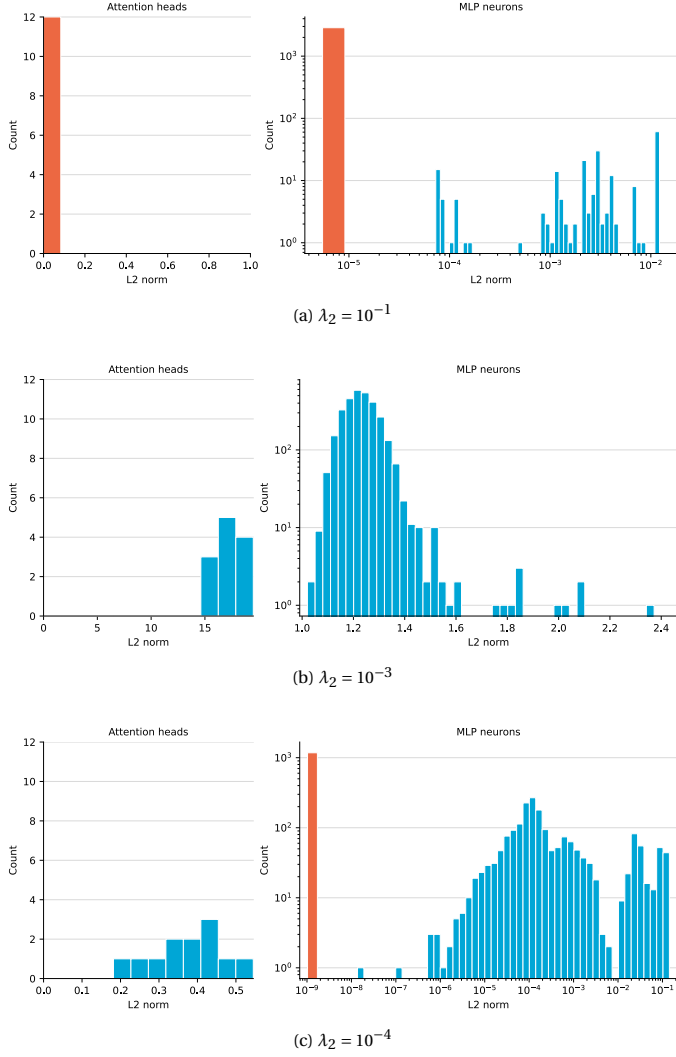


Figure A.11: Distributions of the group L2 norm for the deep encoder (Block 12) at fixed $\lambda_1 = 10^{-1}$ and the three group strengths $\lambda_2 = 10^{-1}$, 10^{-3} and 10^{-4} . Each panel shows the twelve attention-head groups (left) and the 3072 MLP-neuron groups (right), following the group definitions in Table 3.1. Group norms equal to zero are shown in orange, non-zero norms in blue. The middle-block distributions are shown in Figure 4.5.

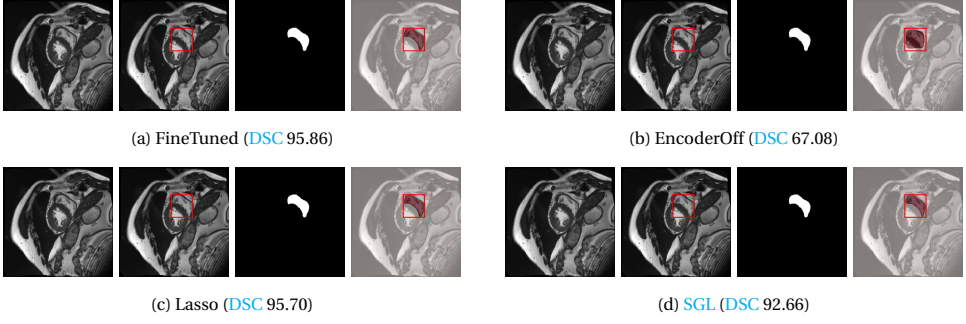


Figure A.12: Predicted masks for the ACDC RV under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

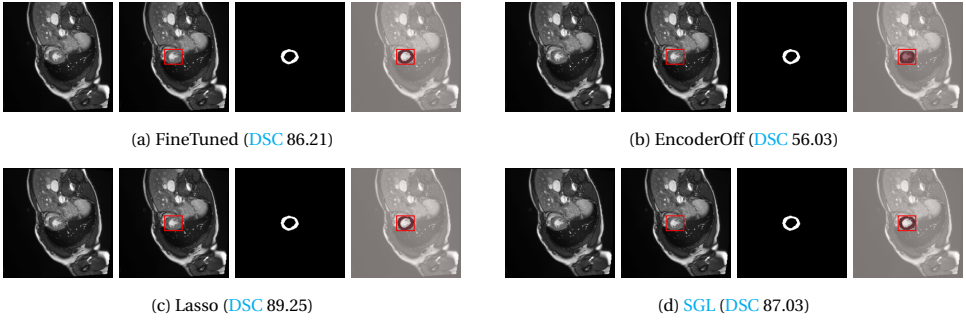


Figure A.13: Predicted masks for the ACDC MYO under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

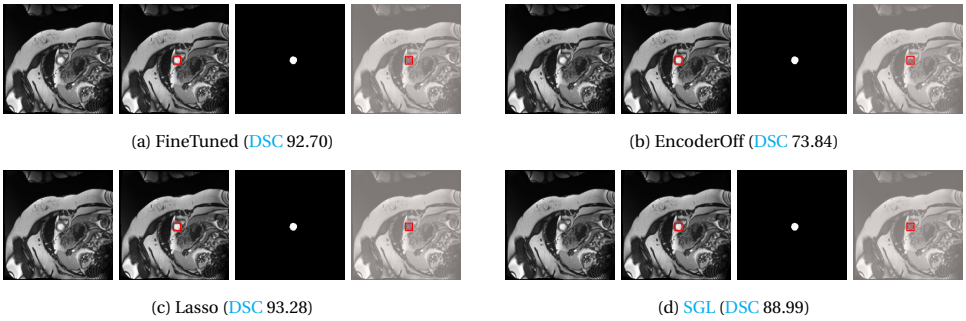


Figure A.14: Predicted masks for the ACDC LV under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

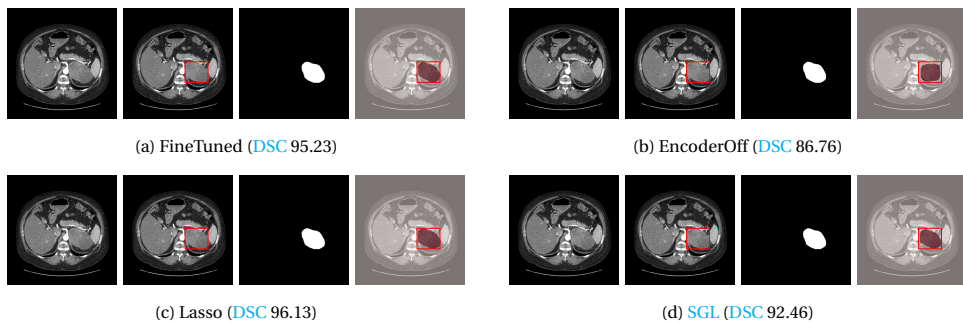


Figure A.15: Predicted masks for the Adrenal-ACC-Ki67-Seg carcinoma under the four ablations: task-loss fine-tuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

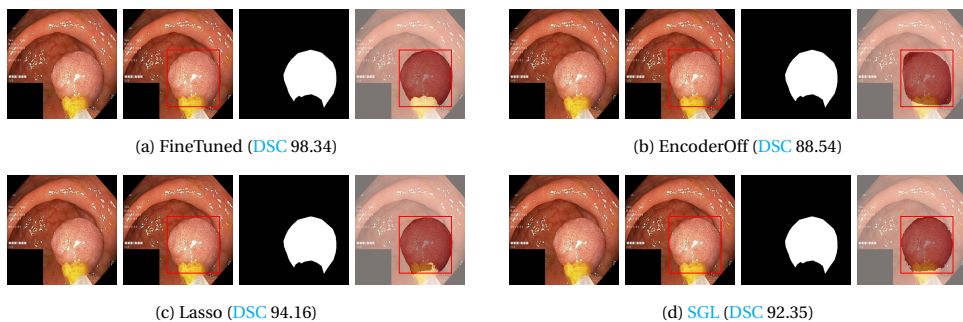


Figure A.16: Predicted masks for the Kvasir-SEG polyp under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

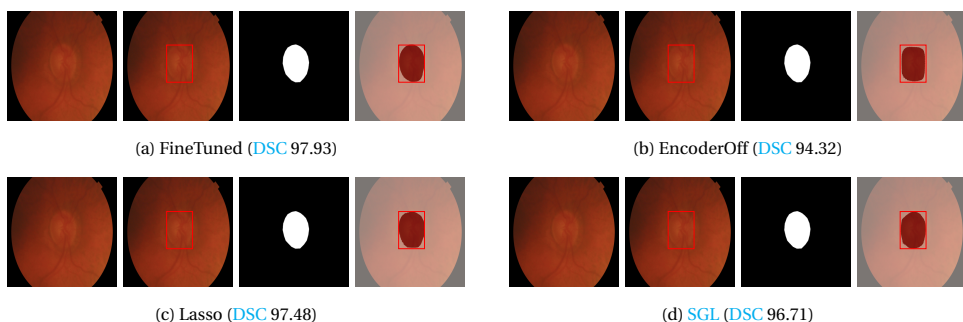


Figure A.17: Predicted masks for the PAPILA Optic Disc under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

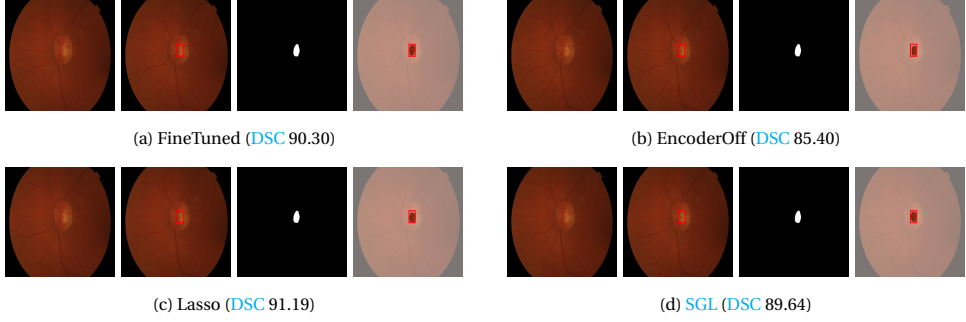


Figure A.18: Predicted masks for the PAPILA Optic Cup under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and **SGL** structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the **DSC** of the displayed sample.

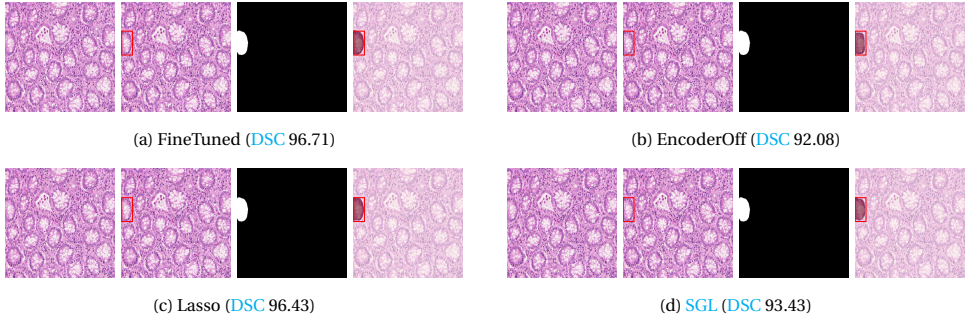


Figure A.19: Predicted masks for the GlaS (testA) glandular boundary under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and **SGL** structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the **DSC** of the displayed sample.

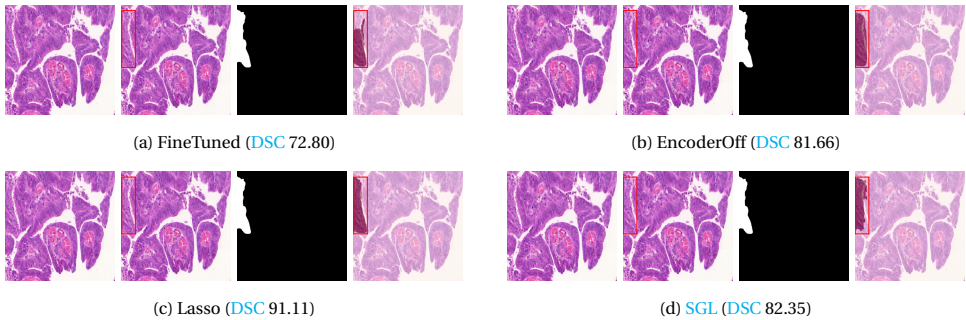


Figure A.20: Predicted masks for the GlaS (testB) glandular boundary under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and **SGL** structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the **DSC** of the displayed sample.

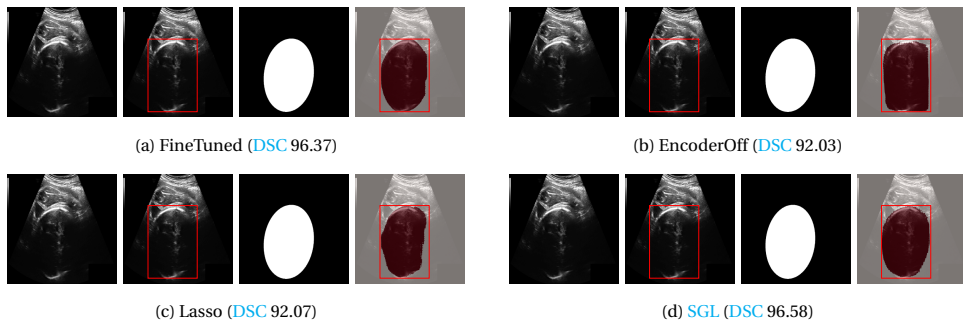


Figure A.21: Predicted masks for the HC18 fetal head under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

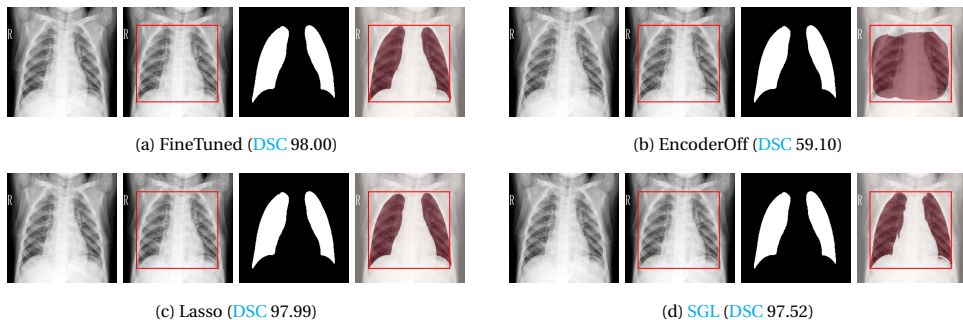


Figure A.22: Predicted masks for the COVID-19 Radiography lung fields under the four ablations: task-loss finetuned, encoder removed, Lasso unstructured pruning and SGL structured pruning. Within each panel, from left to right: input image, bounding-box prompt (red), ground-truth mask, and predicted mask overlaid within the prompt. The reported value is the DSC of the displayed sample.

BIBLIOGRAPHY

- [1] Y. Xu, R. Quan, W. Xu, Y. Huang, X. Chen, and F. Liu, “Advances in medical image segmentation: A comprehensive review of traditional, deep learning and hybrid approaches,” *Bioengineering* 2024, Vol. 11, Page 1034, vol. 11, p. 1034, 10 Oct. 2024, ISSN: 2306-5354. DOI: [10.3390/BIOENGINEERING11101034](https://doi.org/10.3390/BIOENGINEERING11101034). [Online]. Available: <https://www.mdpi.com/2306-5354/11/10/1034/htm%20https://www.mdpi.com/2306-5354/11/10/1034>.
- [2] K. Chupetlovska et al., “Esr essentials: A step-by-step guide of segmentation for radiologists—practice recommendations by the european society of medical imaging informatics,” *European Radiology*, vol. 35, pp. 6894–6904, 11 Nov. 2025, ISSN: 14321084. DOI: [10.1007/S00330-025-11621-1/FIGURES/7](https://doi.org/10.1007/S00330-025-11621-1/FIGURES/7). [Online]. Available: <https://link.springer.com/article/10.1007/s00330-025-11621-1>.
- [3] J. Åkesson, E. Ostensfeld, M. Carlsson, H. Arheden, and E. Heiberg, “Deep learning can yield clinically useful right ventricular segmentations faster than fully manual analysis,” *Scientific Reports*, vol. 13, p. 1216, 1 Dec. 2023, ISSN: 20452322. DOI: [10.1038/S41598-023-28348-Y](https://doi.org/10.1038/S41598-023-28348-Y). [Online]. Available: <https://pmc.ncbi.nlm.nih.gov/articles/PMC9867728/>.
- [4] R. Azad et al., *Medical image segmentation review: The success of u-net*, 2022. arXiv: [2211.14830](https://arxiv.org/abs/2211.14830) [eess.IV]. [Online]. Available: <https://arxiv.org/abs/2211.14830>.
- [5] O. Ronneberger, P. Fischer, and T. Brox, *U-net: Convolutional networks for biomedical image segmentation*, 2015. arXiv: [1505.04597](https://arxiv.org/abs/1505.04597) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1505.04597>.
- [6] L. Zhang et al., “Generalizing deep learning for medical image segmentation to unseen domains via deep stacked transformation,” *IEEE Transactions on Medical Imaging*, vol. 39, no. 7, pp. 2531–2540, 2020. DOI: [10.1109/TMI.2020.2973595](https://doi.org/10.1109/TMI.2020.2973595).
- [7] S. Zhang and D. Metaxas, “On the challenges and perspectives of foundation models for medical image analysis,” *Medical Image Analysis*, vol. 91, p. 102996, Jan. 2024. DOI: [10.1016/j.media.2023.102996](https://doi.org/10.1016/j.media.2023.102996).
- [8] R. Bommasani et al., “On the opportunities and risks of foundation models,” *CoRR*, vol. abs/2108.07258, 2021. arXiv: [2108.07258](https://arxiv.org/abs/2108.07258). [Online]. Available: <https://arxiv.org/abs/2108.07258>.
- [9] J. Ma, Y. He, F. Li, L. Han, C. You, and B. Wang, “Segment anything in medical images,” Apr. 2024. DOI: [10.1038/s41467-024-44824-z](https://doi.org/10.1038/s41467-024-44824-z). [Online]. Available: <http://arxiv.org/abs/2304.12306%20http://dx.doi.org/10.1038/s41467-024-44824-z>.

- [10] J. Ye and H. Tang, *Multimodal large language models for medicine: A comprehensive survey*, 2025. arXiv: [2504 . 21051](https://arxiv.org/abs/2504.21051) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2504.21051>.
- [11] Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, p. 60, Apr. 2023. DOI: [10.3390/computers12030060](https://doi.org/10.3390/computers12030060).
- [12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, Oct. 2015. [Online]. Available: <https://arxiv.org/abs/1510.00149v5>.
- [13] T. Hoefer, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, *Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks*, 2021. arXiv: [2102 . 00554](https://arxiv.org/abs/2102.00554) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2102.00554>.
- [14] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, *Learning structured sparsity in deep neural networks*, 2016. arXiv: [1608 . 03665](https://arxiv.org/abs/1608.03665) [cs.NE]. [Online]. Available: <https://arxiv.org/abs/1608.03665>.
- [15] P. Michel, O. Levy, and G. Neubig, *Are sixteen heads really better than one?* 2019. arXiv: [1905 . 10650](https://arxiv.org/abs/1905.10650) [cs.CL]. [Online]. Available: <https://arxiv.org/abs/1905.10650>.
- [16] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, *Importance estimation for neural network pruning*, 2019. arXiv: [1906 . 10771](https://arxiv.org/abs/1906.10771) [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1906.10771>.
- [17] C. Zhang, S. Xiang, and L. You, *Medcore: Boundary-preserving medical core pruning for medsam*, 2026. arXiv: [2605 . 13688](https://arxiv.org/abs/2605.13688) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2605.13688>.
- [18] C. Zhang et al., *Faster segment anything: Towards lightweight sam for mobile applications*, 2023. arXiv: [2306 . 14289](https://arxiv.org/abs/2306.14289) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2306.14289>.
- [19] Y. Xiong et al., *Efficientsam: Leveraged masked image pretraining for efficient segment anything*, 2023. arXiv: [2312 . 00863](https://arxiv.org/abs/2312.00863) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2312.00863>.
- [20] W. Abebe et al., *Supersam: Crafting a sam supernet via structured pruning and unstructured parameter prioritization*, 2025. arXiv: [2501 . 08504](https://arxiv.org/abs/2501.08504) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2501.08504>.
- [21] A. Kirillov et al., *Segment anything*, 2023. arXiv: [2304 . 02643](https://arxiv.org/abs/2304.02643) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2304.02643>.
- [22] A. Dosovitskiy et al., *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: [2010 . 11929](https://arxiv.org/abs/2010.11929) [cs.CV]. [Online]. Available: <https://arxiv.org/abs/2010.11929>.
- [23] A. Vaswani et al., *Attention is all you need*, 2023. arXiv: [1706 . 03762](https://arxiv.org/abs/1706.03762) [cs.CL]. [Online]. Available: <https://arxiv.org/abs/1706.03762>.

- [24] Y. Ye, G. You, J. K. Fwu, X. Zhu, Q. Yang, and Y. Zhu, "Channel pruning via optimal thresholding," *Communications in Computer and Information Science*, vol. 1333, pp. 508–516, 2020, ISSN: 18650937. DOI: [10.1007/978-3-030-63823-8_58/FIGURES/5](https://doi.org/10.1007/978-3-030-63823-8_58/FIGURES/5). [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-63823-8_58.
- [25] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 58, no. 1, pp. 267–288, Jan. 1996. DOI: [10.1111/j.2517-6161.1996.tb02080.x](https://doi.org/10.1111/j.2517-6161.1996.tb02080.x).
- [26] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 68, no. 1, pp. 49–67, Dec. 2005. DOI: [10.1111/j.1467-9868.2005.00532.x](https://doi.org/10.1111/j.1467-9868.2005.00532.x).
- [27] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *Journal of Computational and Graphical Statistics*, vol. 22, pp. 231–245, 2 2013, ISSN: 10618600. DOI: [10.1080/10618600.2012.681250](https://doi.org/10.1080/10618600.2012.681250). [Online]. Available: <https://www.tandfonline.com/doi/abs/10.1080/10618600.2012.681250>.
- [28] N. Parikh, S. Boyd, N. Parikh, and S. Boyd, "Proximal algorithms," *Foundations and Trends R in Optimization*, vol. 1, pp. 123–231, 3 2013.
- [29] J. Klosa, N. Simon, P. O. Westermarck, V. Liebscher, and D. Wittenburg, "Seagull: Lasso, group lasso and sparse-group lasso regularization for linear regression models via proximal gradient descent," *BMC bioinformatics*, vol. 21, 1 Sep. 2020, ISSN: 1471-2105. DOI: [10.1186/S12859-020-03725-W](https://doi.org/10.1186/S12859-020-03725-W). [Online]. Available: <https://pubmed.ncbi.nlm.nih.gov/32933477/>.
- [30] *Vision and image processing (vip) research group – the university of waterloo's vision and image processing lab*. [Online]. Available: <https://vip.uwaterloo.ca/>.
- [31] O. Bernard et al., "Deep learning techniques for automatic mri cardiac multi-structures segmentation and diagnosis: Is the problem solved?" *IEEE transactions on medical imaging*, vol. 37, pp. 2514–2525, 11 Nov. 2018, ISSN: 1558-254X. DOI: [10.1109/TMI.2018.2837502](https://doi.org/10.1109/TMI.2018.2837502). [Online]. Available: <https://pubmed-ncbi-nlm-nih-gov.tudelft.idm.oclc.org/29994302/>.
- [32] O. Kovalyk, J. Morales-Sánchez, R. Verdú-Monedero, I. Sellés-Navarro, A. Palazón-Cabanes, and J. L. Sancho-Gómez, "Papila: Dataset with fundus images and clinical data of both eyes of the same patient for glaucoma assessment," *Scientific Data* 2022 9:1, vol. 9, pp. 291–, 1 Jun. 2022, ISSN: 2052-4463. DOI: [10.1038/s41597-022-01388-1](https://doi.org/10.1038/s41597-022-01388-1). [Online]. Available: <https://www.nature.com/articles/s41597-022-01388-1>.
- [33] A. W. Moawad et al., *Voxel-level segmentation of pathologically-proven Adrenocortical carcinoma with Ki-67 expression (Adrenal-ACC-Ki67-Seg)*, [Data set], The Cancer Imaging Archive, 2023. DOI: [10.7937/1FPG-VM46](https://doi.org/10.7937/1FPG-VM46). [Online]. Available: <https://doi.org/10.7937/1FPG-VM46>.

- [34] D. Jha et al., "Kvasir-seg: A segmented polyp dataset," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11962 LNCS, pp. 451–462, 2020, ISSN: 16113349. DOI: [10.1007/978-3-030-37734-2_37/FIGURES/4](https://doi.org/10.1007/978-3-030-37734-2_37/FIGURES/4). [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-37734-2_37.
- [35] T. L. van den Heuvel, D. de Bruijn, C. L. de Korte, and B. van Ginneken, "Automated measurement of fetal head circumference using 2d ultrasound images," *PLOS ONE*, vol. 13, e0200412, 8 Aug. 2018, ISSN: 1932-6203. DOI: [10.1371/JOURNAL.PONE.0200412](https://doi.org/10.1371/JOURNAL.PONE.0200412). [Online]. Available: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0200412>.
- [36] K. Sirinukunwattana et al., "Gland segmentation in colon histology images: The glas challenge contest," *Medical Image Analysis*, vol. 35, pp. 489–502, Mar. 2016, ISSN: 13618423. DOI: [10.1016/j.media.2016.08.008](https://doi.org/10.1016/j.media.2016.08.008). [Online]. Available: <https://arxiv.org/abs/1603.00275v2>.
- [37] T. Rahman et al., "Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images," *Computers in Biology and Medicine*, vol. 132, p. 104319, May 2021, ISSN: 0010-4825. DOI: [10.1016/J.COMPBIOMED.2021.104319](https://doi.org/10.1016/J.COMPBIOMED.2021.104319).