STAMP: Simultaneous Training and Model Pruning for Low Data Regimes in Medical Image Segmentation

Nicola K. Dinsdale[a,b], Mark Jenkinson[a,c,d], Ana I. L. Namburete[b]

[a] *Wellcome Centre for Integrative Neuroimaging, FMRIB, Nuffield Department of Clinical Neurosciences, University of Oxford, UK*
[b] *Oxford Machine Learning in NeuroImaging Lab (OMNI), Department of Computer Science, University of Oxford, UK*
[c] *Australian Institute for Machine Learning (AIML), School of Computer Science, University of Adelaide, Adelaide, Australia*
[d] *South Australian Health and Medical Research Institute (SAHMRI), North Terrace, Adelaide, Australia*

Abstract

Acquisition of high quality manual annotations is vital for the development of segmentation algorithms. However, to create them we require a substantial amount of expert time and knowledge. Large numbers of labels are required to train convolutional neural networks due to the vast number of parameters that must be learned in the optimisation process. Here, we develop the STAMP algorithm to allow the *simultaneous* training and pruning of a UNet architecture for medical image segmentation with targeted channelwise dropout to make the network robust to the pruning. We demonstrate the technique across segmentation tasks and imaging modalities. It is then shown that, through online pruning, we are able to train networks to have much higher performance than the equivalent standard UNet models while reducing their size by more than 85% in terms of parameters. This has the potential to allow networks to be directly trained on datasets where very low numbers of labels are available.

Keywords: Model Pruning, Segmentation, Low Data

1. Introduction

Semantic segmentation in medical imaging is vital for the understanding and monitoring of the progression of disease. For instance, the accurate segmentation of the hippocampus is essential for volumetric and morphological assessment, as hippocampal atrophy, observed through MRI, is one of the most validated biomarkers of Alzheimer's disease [1]. Manual segmentation, however, is time consuming and difficult. Several structures have ambiguous boundaries, making consistent delineation of the structures between raters hard to achieve. There is, therefore, a need for automated segmentation methods, capable of reliably providing accurate segmentations of the structures of interest.

Deep learning-based methods have become state-of-the-art for medical image segmentation, with most methods being based on the UNet architecture [2, 3, 4]. Methods either using UNets directly, or highly inspired by the UNet, have been applied to the spectrum

of medical imaging segmentation tasks across a range of modalities: for example, in MRI [5, 6, 7, 8], in CT [7, 9], in X-Ray [7, 10] and in ultrasound [9, 11].

A major limitation of these methods, and with deep learning-based methods in general, is the large amount of labelled data required to train the models. In medical imaging, manual delineation by domain experts is considered to be the 'gold standard' for labels. Therefore, to produce a dataset large enough to train a network for segmentation is expensive, requiring large amounts of expert time to curate, and expert domain knowledge. Furthermore, the labels required often do not form part of standard clinical practice, and therefore have to be produced specifically for the network training. Thus, there is a need to develop methods that can work in data domains where low numbers of labelled data points are available.

One reason why large training dataset sizes are needed is that the models contain large numbers of parameters – many more than the number of training examples available. Many methods have been proposed to help to train models in low data regimes, with the most frequently used approaches being model pretraining or transfer learning [12], and data augmentation [13]. Model pretraining uses a related large dataset to initialise the model weights, allowing the optimisation for the target dataset/task to begin from a more informed place. Data augmentation applies random transformations to the data during training, increasing the size and variation of the dataset seen, thus artificially creating more data points to train the model. We propose to investigate the utility of *model pruning*, reducing the number of parameters in the model and so the model complexity, and thus potentially reducing the number of data points required to train the model.

Model pruning for neural networks was first explored in [14], where they introduced the standard pruning framework. A model is trained until it performs acceptably well on a given task and then removing the weights with small saliency - that is, the weights that have the least effect on the error – then finetuning this pruned network. The process is repeated iteratively until a model of the desired size or performance is obtained. This general format is replicated across the literature; however, the details of each stage vary between the study and the applications.

First, calculation of which weights have the least effect on the model error is computationally highly expensive [15] and so an approximation is desirable. Various metrics have been proposed to approximate this saliency, including the second derivative of the gradient [14], the magnitude of the weights or activations [16, 17, 18, 19], and the product of the activation and gradients [15].

Methods also differ as to the mechanism of the pruning. Some methods prune individual model weights or connections (*unstructured pruning*) [14, 17, 20] whereas others prune whole convolutional filters (*structured pruning*) [16, 15, 18, 19]. Unstructured pruning results in networks that are sparse and it is hard to accelerate these models without specialised libraries, making them difficult to apply using general-purpose hardware [15]. Thus, this work will focus on structured pruning: removing whole convolutional filters. We will also follow the work of [16] and remove whole filters from the model during training time, such that the model becomes progressively smaller.

The methods across the literature prune the network after training to convergence, then

fine-tune the resulting model. This, however, requires the model to be trained to convergence – or at least to good performance by some metric [14] – before we are able to prune the network. This leads to long training times and the methods potentially not being applicable if one cannot train the full-sized network to an acceptable performance in the first place. Few methods attempt to prune the model during training [21, 22]. Other methods exist that have been applied to medical imaging, which aim to create networks with inherently reduced numbers of parameters, such as model distillation [11, 23] and the use of separable convolutions [11, 24].

The obvious question, given that it is often possible to prune networks to less than 50% of the original size of the model, is whether the pruning process can be avoided altogether by simply training smaller models directly. It has, however, been shown by several works in the literature that it is not normally possible to train the smaller pruned networks from random initialisation and achieve the same performance [16, 19]. The Lottery ticket hypothesis [17] demonstrates that there exists a subnetwork of the original network that can be trained to the same level of task performance, given that the starting point is a dense network reduced using unstructured pruning if a suitable initialisation could be found. The subnetwork is then a pruned network, taken in isolation, with the rest of the network removed. These networks are very sensitive to initialisation and smaller networks may be achievable through maintaining the weights of the pruned networks, due to coadapted layers from earlier in the pruning process maintaining the complex relationships that were learned. The sensitivity to the initialisation when pruning convolutional filters as opposed to individual parameters is demonstrated in [25].

Existing pruning methods have been validated in high data regimes, primarily on large computer vision classification datasets such as MNIST [26] – [14, 17, 18] – CIFAR10 [27] – [17, 19] – and ImageNet [28] – [16, 18]. Datasets of this size do not exist in medical imaging, where manual annotations are expensive and time consuming to acquire, and datasets are more commonly of the order of hundreds of labelled examples. We therefore propose to explore using model pruning for training in low data regimes: specifically, our algorithm simultaneously trains and prunes a UNet architecture. The utility of pruning in very low data regimes is explored here for a range of segmentation tasks and datasets for medical imaging.

Our contributions, therefore, are as follow:

- We propose an algorithm to *simultaneously* train and prune a UNet. It is, to the best of our knowledge, the first exploration of pruning applied to a UNet for medical image segmentation, removing whole convolutional filters through the pruning process. We demonstrate our technique on several open access datasets and segmentation tasks, across imaging modalities.

- A channelwise targeted dropout is developed, based on that proposed in [29], but adapted to work channelwise and without requiring availability of information about the filters that are most likely to be removed before training. This is similar to the method proposed in [30], where channelwise weighted dropout is used to regularise the

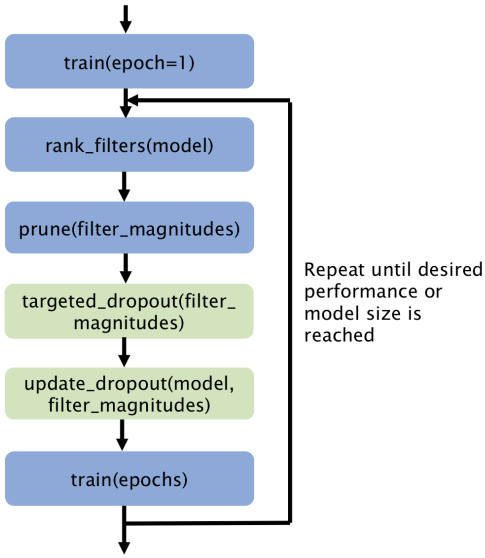


Figure 1: Overall STAMP+ training procedure for training and pruning a model simultaneously. The targeted dropout algorithm is presented in Algorithm 1. Steps in green are removed in ablation experiments.

training of a model, but the normalised filter magnitudes calculated for pruning are used to modulate the dropout. It is demonstrated that this improves the performance and stability of the models during training, by making them more robust to the pruning process.

- Finally, we explore model pruning for very low data regimes, and we show that, through training and pruning the model simultaneously, it is possible to achieve significantly higher performance than can be obtained by directly training UNet models of the equivalent (final) size.

2. Methods

Our aim is to train a deep neural network to segment regions of interest from input images while limiting the degree of overfitting, by pruning the model during training. We hypothesize that the trained model should requires *less* training data points to produce good quality segmentations, due to the reduction in parameters that must be optimised. As this work uses medical images, where labels are expensive to acquire, it must function in a low data regime.

To allow the network to be trained and pruned simultaneously, consider the scenario where there is access to a training dataset $\mathcal{D} = \{\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2 ... \boldsymbol{x}_N\}, \boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2 ... \boldsymbol{y}_N\}\}$ where $\boldsymbol{x}_n \in \mathcal{R}^{W \times H \times D \times 1}$ represents an input image and $\boldsymbol{y}_n \in \mathcal{R}^{W \times H \times D \times C}$ is its corresponding target segmentation, and where C corresponds to the number of classes in the target segmentation. A network is trained to predict the target segmentations $\boldsymbol{y}_n$ from the input images $\boldsymbol{x}_n$ and the network is parametrised by $\mathcal{W} = \{(\boldsymbol{w}_1^1, b_1^1), (\boldsymbol{w}_1^2, b_1^2), ...(\boldsymbol{w}_L^{C_L}, b_L^{C_L})\}$ where $\boldsymbol{w}_m^n$ corresponds to the weight kernel for the nth filter of the mth layer and b_m^n is the

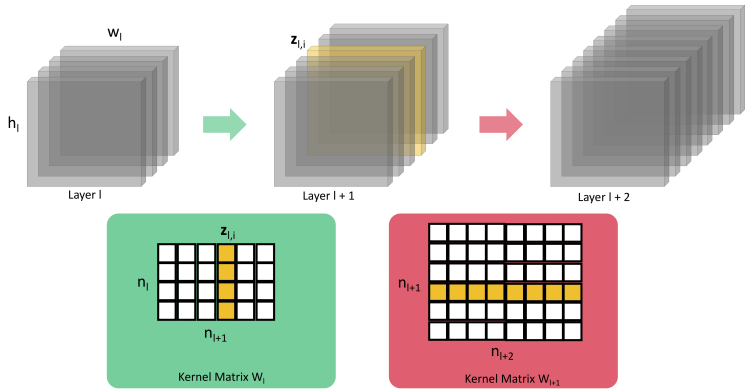


Figure 2: Whole convolutional filters are pruned, based on the magnitude of the corresponding feature maps $\boldsymbol{z}_{l,i}$. This requires the weights to be pruned from the kernel matrix $\boldsymbol{w}_l$ and $\boldsymbol{w}_{l+1}$. The corresponding biases are also pruned but are not shown.

corresponding bias that will be incorporated into discussion of weights from here on. C_L is the number of channels in layer L. The weights are first randomly initialised and are optimised during training by minimising a loss function $\mathcal{L}(\mathcal{D}|\mathcal{W})$ that aims to optimise the performance on the segmentation task and can be chosen completely independently of this pruning process.

During the pruning process, the aim is to refine the parameters of the network $\mathcal{W}$ to a smaller subset of the parameters $\mathcal{W}'$ such that $\mathcal{L}(\mathcal{D}|\mathcal{W}') \approx \mathcal{L}(\mathcal{D}|\mathcal{W})$. To achieve this, we use the observation that the parameters of the smallest magnitude generally have the least impact on the final segmentation produced by the network, and can be removed with the least impact on model performance.

2.1. Training Procedure

The overall training procedure is shown in Fig. 1. The model weights are initialised with Xavier initialization [31] and the model is trained for a single epoch before pruning begins, so that the filters contain some information about the task and not just the initialised weights.

Our pruning strategy removes whole convolutional kernels as shown in Fig. 2, meaning that we do not end up with a sparse representation and are able to use standard libraries and hardware. When pruning whole convolutional filters, we remove the filter from the kernel matrix $\boldsymbol{w}_i$, the corresponding bias, and the filter from the weight kernel matrix $\boldsymbol{w}_{i+1}$. To decide which filters to remove, we consider the feature activation maps, denoted by $\boldsymbol{z}_l^{(k)} \in \mathcal{R}^{W_l \times H_l \times D_l \times C_l}$ where $l \in \{1, L\}$, is the current layer, and C_l is the number of channels at layer depth l. The activation maps are related to the kernel weights by:

$$\boldsymbol{z}_l^{(k)} = ReLU(\boldsymbol{z}_{(l-1)} * \boldsymbol{w}_l^{(k)} + b_l^{(k)}) \tag{1}$$

where ReLU is the activation function used throughout our network, other than for the final convolution. The use of ReLU means that our network contains a substantial number of zero-valued parameters, creating redundancy that can be pruned.

We use the observation that weights with the smallest impact on the final prediction are those with the smallest magnitudes [14], and so the weights, $\boldsymbol{w}_l^{(k)}$, and biases $b_l^{(k)}$, corresponding to the smallest magnitude filter activations, $\boldsymbol{z}_l^{(k)}$, are pruned from the network. To assess the overall magnitude of the filters, the $L2$ norm, averaged across all training data points (calculated from an additional forward pass), is considered:

$$\Theta_{L_2}(\boldsymbol{z}_l^{(k)}) = \frac{1}{N}\sum_{i=1}^{N} || \ \boldsymbol{z}_{l,i}^{(k)} \ ||_2 \tag{2}$$

The $L2$ norm is used as it is computationally simple and provides stable performance, but other metrics could be used to evaluate the magnitude of the filters: it was found that the performance was robust to the choice of metric used (see Section 3.1).

These values must then be normalised across the layer depth, because the values are at different scales at different depths [15]. Therefore, a simple $L2$ normalisation across the values at each layer is employed.

$$\hat{\Theta}_{L_2}(\boldsymbol{z}_l^{(k)}) = \frac{\Theta_{L_2}(\boldsymbol{z}_l^{(k)})}{\sqrt{\sum_j (\Theta_{L_2}(\boldsymbol{z}_j^{(k)}))^2}} \tag{3}$$

To prune the filter, it is removed entirely from the model architecture, rather than just set to zero, and so the model architecture reduces in the number of parameters as the training progresses. Practically, this is achieved by creating a smaller model with the selected filter removed and then reloading all of the weights apart from those corresponding to the pruned filter.

2.2. Adaptive Channelwise Targeted Dropout

To make the model more robust to the pruning, *adaptive channelwise targeted dropout*, based on [29] and [30], is introduced. In [29] dropout is applied to units which are *a priori* believed to be the least useful, thus encouraging the network to learn a representation that is more robust to *post-hoc* sparsification. In [30] weighted channelwise dropout is applied during training to regularise the model training. Global Average Pooling is used to determine the importance of individual filters, and a different dropout value is applied to each filter. Here, we apply the same dropout value to all of the filters at a given layer depth such that the information carried by the pruned filters is removed, and not redistributed to other filters at that depth, otherwise the model will continue to overfit to the training data. Like [30] the dropout is applied channelwise, rather than parameter-wise [29], so that the network is prepared for the removal of whole convolutional filters; thus, channelwise or *spatial* dropout [32] is added to our model.

We do not know which filters are the most likely to be pruned *a priori* and, due to not having a pretrained model which is then pruned, the filter magnitudes cannot be used as a basis to predict this before the process begins. Therefore, unlike in [29], the targeted dropout values cannot be based on a known probability distribution. The dropout values are therefore changed adaptively during training, based on the calculated filter magnitudes,

Algorithm 1: Adaptive Targeted Dropout Algorithm. The *index* function returns the index (or position) of the Θ for the kth filter at layer l within the full sorted list of Θ values across the network. From this, the average index at each layer depth is calculated and used for determining the dropout probability ($\boldsymbol{p}$) for each layer. The output $\boldsymbol{p}$ is a vector of probabilities.

Input: Normalised filter magnitudes for every filter at every depth, output from pruning algorithm: $\boldsymbol{\Theta} = [\hat{\Theta}_{L_2}(\boldsymbol{z}_l^{(k)}),\ [l,\ k]]\ \forall\ l \in \{1, L\}, k \in \{1, C_l\}$

Result: Targeted dropout values: $\boldsymbol{p}_l\ \forall\ l \in L$

$\boldsymbol{\Theta}_{sorted} = sort_descending(\boldsymbol{\Theta})$ ▷ according to $\hat{\Theta}_{L_2}(\boldsymbol{z}_l^{(k)})$ values

for $l \leftarrow 1$ *to* L **do**

  $\boldsymbol{\mu}_l = \frac{1}{C_l}\sum_{k=1}^{C_l} index(\boldsymbol{\Theta}_{sorted}, [l, k])$ ▷ Find index for $[l, k]$

end

$\boldsymbol{\mu} = \boldsymbol{\mu}/max(\boldsymbol{\mu})$

$\boldsymbol{p} = \boldsymbol{\mu} \times b_{drop}$ ▷ Multiply by base dropout value

$\hat{\Theta}_{L_2}(\boldsymbol{z}_l^{(k)})$. For the first iteration, all dropout values are set equal to the base dropout value b_{drop} (the maximum dropout value applied) and then the values are recalculated after each pruning iteration, as the new magnitudes $\hat{\Theta}_{L_2}(\boldsymbol{z}_l^{(k)})$ are calculated.

Algorithm 1 shows the procedure used to calculate the adaptive targeted dropout values. It should be noted that the indices of the ordered normalised filter magnitudes (i.e. position in the sorted list) are used to calculate the dropout values rather than the $\hat{\Theta}_{L_2}(\boldsymbol{z}_l^{(k)})$ values directly, as the discrete values were found empirically to better encode the relative likelihoods of the filter being pruned from the model. When the new dropout values, $\boldsymbol{p}$, are calculated, the model architecture is updated with the new spatial dropout values, with the same value being applied to all filters in a layer.

2.3. Model Training

After pruning and updating the targeted dropout values, the model is then trained using standard backpropagation. The number of epochs between filter prunings, described as *recovery epochs*, is a hyperparameter that needs to be decided, but a value of 2 is used unless otherwise stated. The effect of this hyperparameter on the segmentation performance will be explored.

In the experiments, the network is trained until it cannot be pruned any more, to allow exploration of the model's performance, even as the model becomes very small. A model is considered to have reached the pruning limit when only a single filter remains at each depth and so to prune further would be to break the model. In practice, the model could be trained with standard early stopping, to save the best performing model.

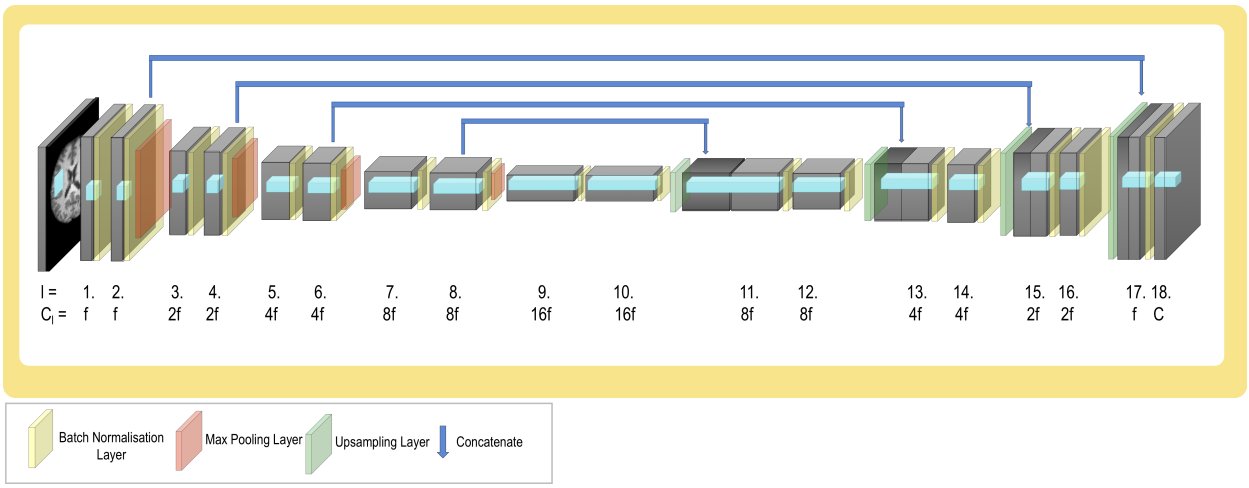


Figure 3: UNet model architecture - it follows the standard pattern of halving resolution and doubling filters at each depth. l corresponds to the layer depth, C_l is the number of channels in that layer and C is the number of classes in the output segmentation. f is the initial number of filters, and is varied across experiments, but is 4 unless otherwise stated.

2.4. Network Architecture

Across the experiments, a standard 2D or 3D UNet architecture is explored [2, 3] (Fig. 3) with two blocks of convolutional layers at each depth, each with a batch normalisation layer [33]. The UNet architecture is considered as it is widely used throughout medical image segmentation, and many winning architectures in segmentation challenges are based on the it. Only the basic UNet is considered, but it is expected that the findings should generalise well to UNet-derived (encoder-decoder) architectures.

ReLU activations are used throughout, apart from the final layer, where a sigmoid activation function is used to create the segmentations. ReLU is very commonly used in network architectures, and, when we are considering pruning, comes with the added advantage that it encourages the network parameters to become sparse, so filter values on average become lower. Spatial dropout is applied to each convolutional block during training, with the values determined by the Targeted Dropout algorithm explained in Algorithm 1.

The final layer ($l = 18$) contains the same number of filters as output classes C and so clearly cannot be pruned (in all cases the background class is considered as an additional class). Therefore, only filters in layers $1 - 17$ will be considered as candidates to be pruned and therefore for Eq. (1) is evaluated over $l \in \{1, L\}$ where $L = 17$ and the final layer is not considered.

The architecture follows the conventional pattern of halving resolution and doubling features at each depth. The value of f, which determines the number of filters in each layer, is varied in the experiments, allowing exploration of the effect of the initial model size on the pruning process and the model's performance. In our experiments, it is set to 4 unless stated otherwise. The maximum size of the network architecture explored was determined by the available GPU memory. A shallower UNet was also considered, the results of which can be seen in the supplementary material.

2.5. Datasets and Implementation Details

In all of the experiments, 2 epochs of training were used between filter prunings, with a batch size of 16 and an Adam optimiser [34] with a learning rate of 0.01. Five-fold cross-validation (the training data was split 80% training, 20% validation) was used for all experiments and all reported values use a further held-out test set for each dataset.

For the various experiments, two variations of the proposed method and two benchmarks are considered, which will be referred to as follows:

- STAMP - The proposed method, training and pruning the network simultaneously, without the green blocks shown in Fig. 1, the procedure therefore consisting solely of calculating the $\hat{\Theta}$ values and pruning the smallest filter;
- STAMP+ Targeted Dropout (STAMP+) - The proposed method, including the targeted dropout – all of the blocks in Fig. 1. These first two methods allow exploration of the impact of the pruning and the targeted dropout separately;
- *Standard UNet* - The first baseline, in which the UNet model is trained to convergence, without pruning, with the same hyperparameters and a patience of 25 epochs, representing the standard approach;
- *PruneFinetune* - The second baseline, in which a UNet that is trained to convergence, without pruning, is used for initialisation, and then the model is pruned using the same framework as for STAMP+. A patience of 10 epochs was used during the fine-tuning stages. This is the classic pruning regime, allowing exploration of the advantage of STAMP+, when pruning and training simultaneously.

The same data splits were used for evaluating all methods. No augmentation was applied to the data at any stage, so that any increase in generalisability is due to the pruning process alone and is not a function of any augmentation applied. We report Dice scores throughout, but it can be found in the supplementary material that the results were consistent across other common evaluation metrics. All reported statistics are paired t-tests. When a single value is reported for the performance of the pruning methods, the best model is chosen on the basis of results obtained from the validation data. The results are then reported for the held-out testing data, using the selected model. All graphs show the mean results and interquartile range. The base dropout value b_{drop} was set to 0.05 in all cases.

The code was implemented using Python 3.5.2 and PyTorch 1.0.1.post2 [35]. All models were trained using a V100 GPU.

2.5.1. Datasets

Seven medical imaging datasets were chosen for these experiments, spanning a range of tasks and imaging modalities. Basic details are listed in Table 1.

HarP: A single-site dataset with high quality manual segmentations. $64 \times 64 \times 64$ voxel patches were extracted around each hippocampus, providing 200 volumes (left and right hippocampi combined) for training and 70 for testing. T1 images are used as the input. $C = 2$ (foreground and background).

Dataset	Modality	2D/3D	Task	Image Size	Training/ Testing Subjects
HarP [36]	T1 MRI	3D	Hippocampal Segmentation	$64 \times 64 \times 64$	200/70
IXI - HH	T1 MRI	3D	Brain Extraction	$128 \times 128 \times 128$	166/19
IXI - Guys	T1 MRI	3D	Subcortical Segmentation	$128 \times 128 \times 128$	257/65
Medical Decathlon - Cardiac [37]	bSSFP MRI	2D	Right Atrium	320×320	15/5 (split into slices)
Medical Decathlon - Spleen [38]	CT	2D	Whole Spleen	512×512	28/10 (split into slices)
Medical Decathlon - Prostate [39]	Multimodal MR (T2 & ADC)	2D	Whole Prostate	320×320	28/10 (split into slices)
INTERGROWTH-21st Fetal Ultrasound [40]	Ultrasound	3D	Fetal Brain	$160 \times 160 \times 160$	822 / 274

Table 1: Details of the datasets used in pruning experiments.

IXI[1] **- HH**: Hammersmith Hospital data from a 3T MRI scanner. The data were processed using the FSL Anat[2] pipeline. The task considered was brain extraction and the labels were automatically generated using the FSL Anat processing pipeline, which non-linearly registers an atlas to the target image. $C = 2$ (foreground and background).

IXI - Guys: Guy's hospital data from a 1.5T MRI scanner. The data was processed using the FSL Anat pipeline. The target task was subcortical segmentation of three deep grey matter structures: caudate, putamen and thalamus, with the labels automatically generated using FSL FIRST [41] run on the bias-field corrected and registered images output from the FSL Anat pipeline. $C = 4$ (caudate, putamen and thalamus, and background).

Medical Decathlon - Cardiac: 20 MRI datasets covering the entire heart, which were acquired during a single cardiac phase (free breathing with respiratory and ECG gating). This represents a small dataset with large variability, where the task was the segmentation of the right atrium. $C = 2$ (right atrium and background).

Medical Decathlon - Spleen: This dataset was originally reported in [38] and consists of 41 CT scans with automatic segmentations that were manually corrected. This demonstrates the method on a different imaging modality with the dataset having a large range of foreground sizes. The task was the segmentation of the spleen. $C = 2$ (spleen and background).

Medical Decathlon - Prostate: This dataset was originally reported in [39], and consists of 38 Multimodal MRI datasets with labels of the whole prostate. We demonstrate that the method can be applied to multichannel input data, and the dataset requires the segmentation of two adjacent regions with large intersubject variation. The dataset was originally reported in [39]. $C = 3$ (two prostate regions and background).

INTERGROWTH-21st: These data were collected as part of the INTERGROWTH-21st study and represent a healthy fetal cohort, and consist of 1096 Ultrasound sessions that

[1]Data downloaded from: https://brain-development.org/ixi-dataset

[2]For details see: https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/fsl_anat

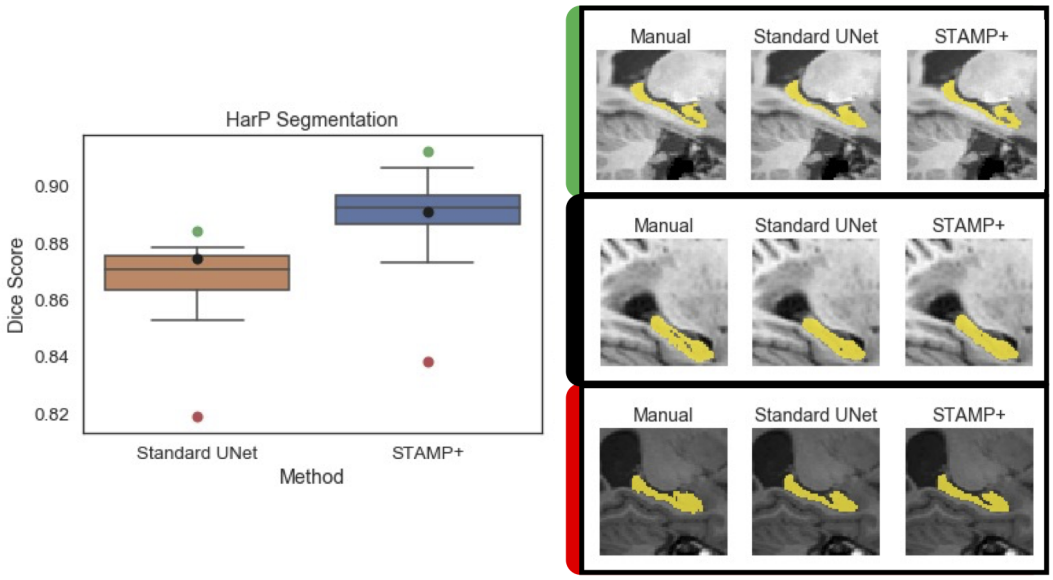


Figure 4: Representative segmentations from the HarP test set. The colour coded boxes correspond to the same colour point on the box plot. The result shown in green (top) is the best segmentation, in red (bottom) the worst, and black (middle) represents a standard result.

were unaligned with varying fields of view. In each scan, the brain occupied $> 50\%$ of the image space. The images had a range of sizes and were resampled, then cropped to create images of the same size, with the brain at the centre of the image. The brain mask labels were created by manually aligning (similarity transform) the age-matching fetal MRI atlas from [42] to each US image. $C = 2$ (fetal brain and background).

3. Results

3.1. HarP Data - Evaluation of the Algorithm

We first used the HarP dataset to investigate the different aspects of the approach when using a dataset with high quality, consistent manual labels. Once the different aspects of the pruning algorithm were understood, the effect of the amount of available training data was explored. Therefore, this section explores the effect of:

- the choice of pruning metric,
- the number of filters,
- the use of the targeted dropout,
- the base value of the targeted dropout out, b_{drop},
- the number of recovery epochs between filter pruning events.

Example segmentations from these experiments can be seen in Fig. 4.

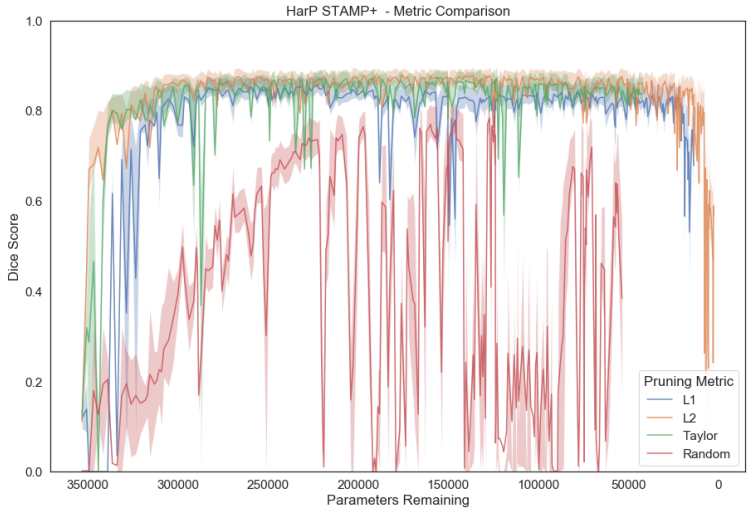


Figure 5: The performance of the STAMP+ algorithm with different choices of pruning metric using the HarP data, with $f = 4$. It can be seen that the three metrics performed similarly, with no significant difference between L2 and Taylor metrics. All three metrics were better than random pruning. Note that the results are plotted against the remaining parameters rather than the pruning iteration.

3.1.1. Choice of Metric

First, we explored the choice of metric used to determine which filters to prune. As stated in section 2, the L2 norm was chosen as the pruning metric and the results shown in Fig. 5 form the basis of that decision. The L2 norm is compared with:

- L1 norm [19]: $\Theta_{L_1}(\boldsymbol{z}_l^{(k)}) = \frac{1}{N}\sum_{i=1}^{N} || \boldsymbol{z}_{l,i}^{(k)} ||_1$
- Taylor – proposed in [15], which utilises the product of the activation and the gradient of the loss function: $\Theta_{TE}(\boldsymbol{z}_l^{(k)}) = ||\frac{1}{N}\sum_{i=1}^{N} \frac{\delta C}{\delta \boldsymbol{z}_{l,i}^{(k)}} \boldsymbol{z}_{l,i}^{(k)}||_1$
- Random: $\Theta_{Rand} \sim Uniform(0, 1)$

where all the metrics but Random are normalised according to Equation 3. This normalisation ensures that the last filter at any depth is not pruned. With Random pruning, the condition that the last kernel at a given depth cannot be pruned is explicitly coded. Figure 5 shows the Dice scores on the test dataset at each pruning iteration for models trained using *STAMP+* and with each metric in turn. For each metric, the mean value across the test set is shown as the solid line, and the shaded region indicates the interquartile range. The results are plotted against the remaining parameters rather than the pruning iteration. This is done because different pruning iterations remove different numbers of parameters, depending on network location from which the filter is removed: for instance, if it also leads to filters being removed across the skip connection.

It can first be seen that all three metrics perform better than randomly pruning channels, as would clearly be expected. As the models start from random initialisation, the performance initially for all four metrics is poor, then improves rapidly for all metrics except the

random one as the model training continues. It can then be seen that all three of the metrics perform comparably on this task, with no significant difference between the performance of the L2 and Taylor metrics (L2 vs Taylor: $p = 0.07$, L2 vs L1: $p = 0.003$, L2 vs Random: $p = 5.7 \times 10^{-11}$). As the L2 norm is computationally more efficient than the Taylor metric, it is used throughout the remaining experiments.

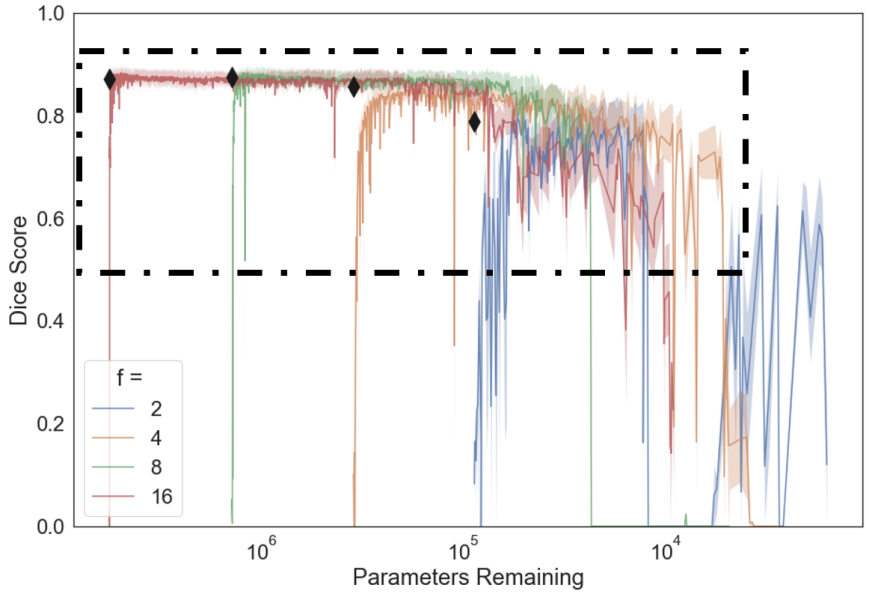
3.1.2. Varying Numbers of Filters f

Having validated the use of the L2 norm as the pruning metric, the *STAMP+* algorithm was tested on the HarP data for models with varying numbers of filters: $f = [2, 4, 8, 16]$. Each model is simultaneously pruned and trained until the model is unable to be pruned any further, leaving only one filter remaining in each layer. Figure 6a shows the performance of the model: the performance is evaluated on the test set for each iteration of the pruned model; the mean value (the solid line) and the interquartile range bounds (shaded) are shown. The result is shown for four different initial sizes of model, plotted against the number of parameters remaining in the model. For comparison, the correspondingly sized *Standard UNet* models were also trained until convergence.

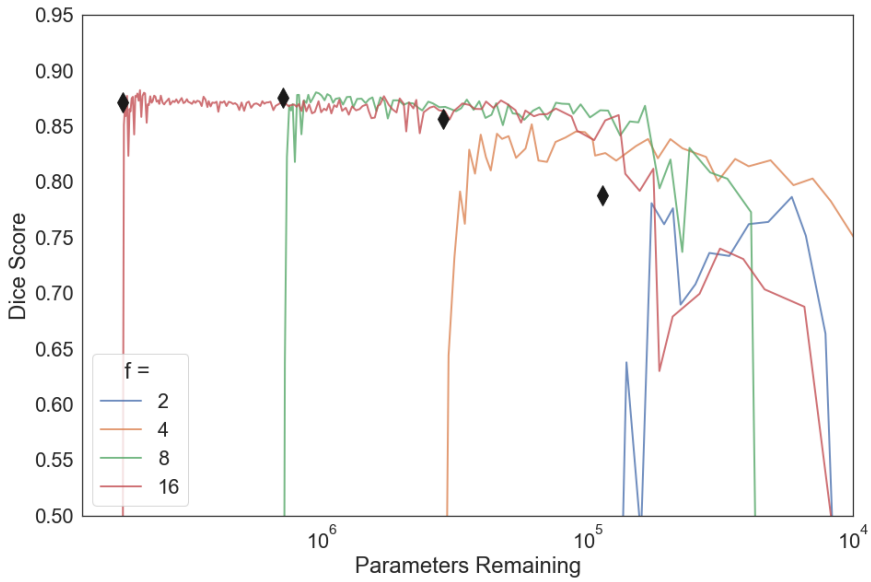
First, it can be seen that the model can be pruned and trained simultaneously, such that the same network performance is reached as the model trained to convergence; therefore, the pruned network is sufficiently powerful to be able to represent the variation in the data. It is also evident from Fig. 6 that it is possible to prune the models to a fraction of their original size without reducing the network performance. We also found that the smallest model, when $f = 2$, is not able to perform the segmentation well (maximum Dice score $= 0.788 \pm 0.052$ for the standard UNet and 0.799 ± 0.050 for the pruned model); however the larger networks can be pruned to the same number of parameters as the $f = 2$ model (and smaller) and still perform well on the segmentation task. This indicates that by pruning a larger model we are able to learn a better arrangement of parameters than when naïvely creating a model of that size.

Figure 7 shows the distribution of filters of the differently sized networks both at initialisation and at the point in pruning where they have the same number of parameters as the $f = 2$ model. The number of filters in these pruned models correspond to the $f = 2$ model trained to convergence having a Dice score of 0.788 ± 0.052 compared to the pruned models' performances of 0.833 ± 0.041, 0.859 ± 0.047, and 0.861 ± 0.028 for $f = 4, 8, 16$ respectively. This means that all of the larger pruned models perform significantly better than the *Standard UNet* model of that size (the best performing model was selected using the validation data). It can be seen that the filter distributions learned through pruning the models are very different to the standard UNet architecture of doubling filters, instead having a far flatter distribution, with a similar number of features at each depth being maintained.

Results for training with an initially shallower UNet can be found in the supplementary material, where it can be seen that the same patterns of results are found as with the original depth considered here.



(a)



(b)

Figure 6: a) Results for pruning models of varying model sizes – using different values of f – while training to segment the hippocampus on the HarP data. The black diamond corresponds to a model of the same size being trained to convergence on the same data, with all hyper-parameters held the same except for the absence of pruning the network. b) The same result, zoomed in and subsampled (only every 10th data point shown, only the mean, with no interquartile range shown) to allow the result to be seen more clearly. The box on a) indicates the zoomed in region.

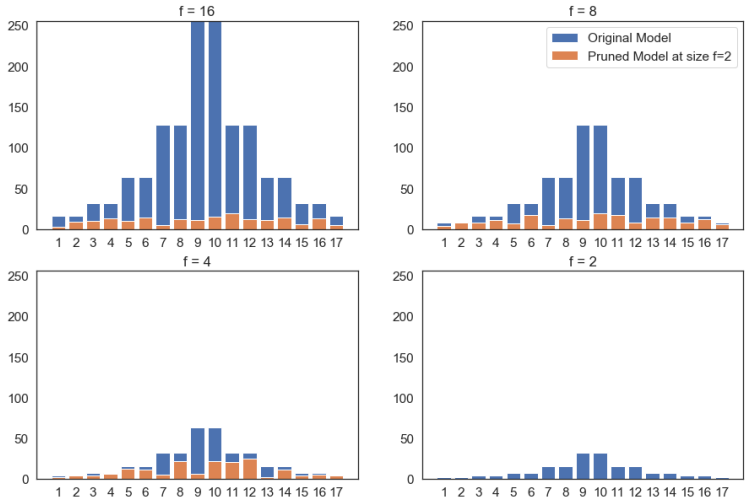


Figure 7: The number of filters in each layer for the models trained in figure 6. The distribution of the number of filters shown in blue corresponds to the model's original filter distribution, and the one shown in orange is the distribution when the model has the same total number of parameters as the original model with $f = 2$.

3.1.3. Method Comparison

In Fig. 8 the results achieved using different pruning methods are compared: STAMP+, *PruneFinetune*, and STAMP. First, it can be seen, by comparing STAMP with and without targeted dropout, that the targeted dropout both improves the model's performance (from 0.844 ± 0.03 to $0.879 \pm 0.02, p < 0.001$) and additionally makes the training more stable. This shows that the targeted dropout successfully makes the model more robust to being pruned, even with this relatively simple task.

Comparing STAMP to *PruneFinetune*, we can see that pruning the already converged model leads to more stable training than STAMP (Fig. 8). However, the removal of a single filter at a time is very conservative compared to the normal approaches taken in the literature, where a percentage of filters, or all filters under a threshold value are removed [14], which we would then expect to be less stable. Furthermore, the recovery time between filter prunings is much longer, as the model is allowed to recover entirely before additional pruning is performed (representative training graphs can be found in the supplementary material). Finally, the original had to be trained to convergence before the network could be pruned, and so *PruneFinetune* has a much larger computational cost while performing less well than the proposed method of STAMP+$(0.860 \pm 0.03, 0.879 \pm 0.02, p < 0.001)$. Therefore, the proposed method allows better performance to be achieved without having to train the original model.

3.1.4. Filter Magnitudes

Remembering that the UNet architecture conventionally has doubling features with increasing model depth and halving resolution, Fig. 9 shows the adaptive targeted dropout probability values at each layer depth through training, for a model trained with $f = 8$,

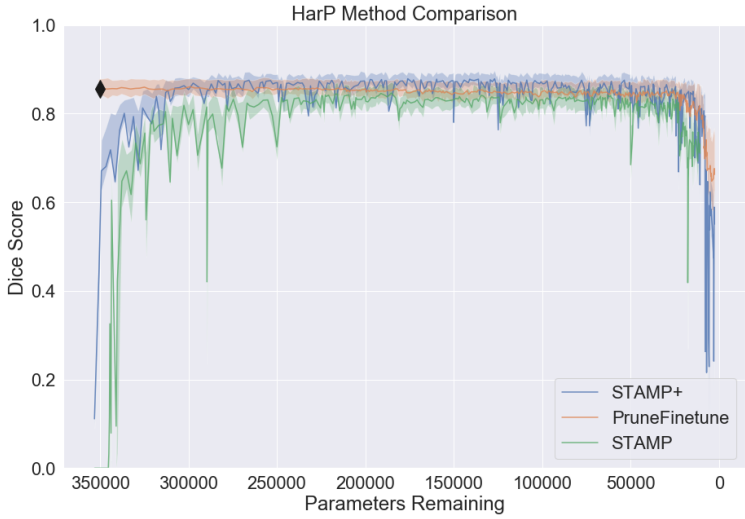


Figure 8: Comparison of pruning the HarP data with $f = 4$ for our proposed method, STAMP+, *Prune-Finetune* and STAMP (i.e. without targeted dropout - green blocks from Fig. 1 removed). The black diamond indicates the performance of the *Standard UNet* trained to convergence without any pruning – $f = 4$.

with every 30 iterations being reported. The x value corresponds with the location model, as indicated by the model architecture (Fig. 3). It can be seen that until the model becomes very small (highly pruned, iterations higher than 600) the distribution of the calculated values is very similar between iterations, with a tendency for the bottleneck of the model to have the highest dropout value. This is to be expected, as this is where most filters in the model are located and, thus, where there is likely to be the greatest level of redundancy. It can also be seen that towards the end of training the dropout values for some of the depths are 0. These correspond to the depth where only a single filter remains, thus to prune (or to apply dropout) would break the model and information would be unable to flow.

3.1.5. Filter Survival

Figure 10a shows the distribution of the number of filters at each depth as the network is pruned. The darker the section of the bar, representing the number of filters at a given layer, the longer the filters remained in the model architecture. As would be expected, because they are based on the activation magnitude values, the filters in the bottleneck of the network are pruned aggressively first, corresponding to the high dropout values. It can also be seen across the network that the first layer within a pair of layers, at a given depth, is pruned more quickly than the second, for both the encoder and the decoder.

Figure 10b shows the magnitudes of the filters: $\hat{\Theta}_{L_2}(\boldsymbol{z}_l^{(k)})$ averaged across the testing dataset as the model is gradually pruned. It can be seen that as the model is gradually pruned, the average value of the remaining filters increases and the high magnitude filters remain throughout the pruning. The first dashed vertical line at iteration 182 represents the point in the pruning where the model had the best performance on the testing data (49% of the filters of the original model). It can be seen that the majority of the low-magnitude

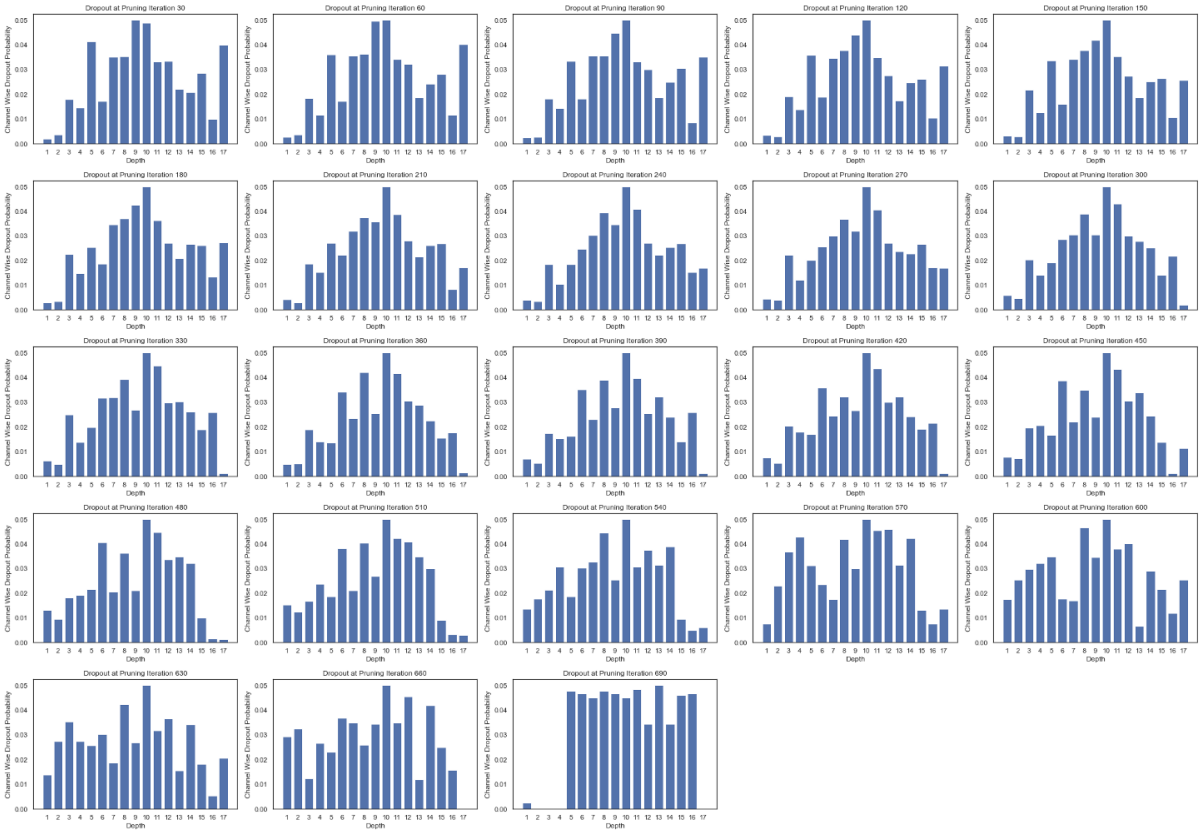


Figure 9: The adaptive targeted dropout probability values at each layer (depth) for every 30 epochs of training on the HarP data – $b_{drop} = 0.05$ – for $f = 8$. It can be seen that the dropout values are quite consistent across pruning iterations until the network becomes very small in size. When a layer has a single filter at that depth, the dropout value becomes 0, guaranteeing that information can be propagated through the model.

filters (assumed to be uninformative) have been removed, showing that more than half of the filters in the model can be removed without having a negative impact on the segmentation. The second dashed line represents the smallest network (20% of the filters of the original model) for which the performance was not significantly worse than the best performing model. This indicates that the network can be reduced to a small number of filters, where all of the activations are playing a more important role in producing the outputs.

These results clearly demonstrate that the proposed method can be utilised to simultaneously train and prune a UNet, while working in a low-data regime, as is common in medical imaging. The results also indicate the benefit of the addition of the targeted dropout, showing an increased performance, even on this relatively easy task.

3.1.6. Base Dropout Value b_{drop}

The addition of the targeted dropout does, however, introduce a hyperparameter: the base dropout value b_{drop}. Figure 11 explores the effect of changing the value of b_{drop} on the proposed STAMP+ method. Across the experiments, a base value of 0.05 was used, meaning

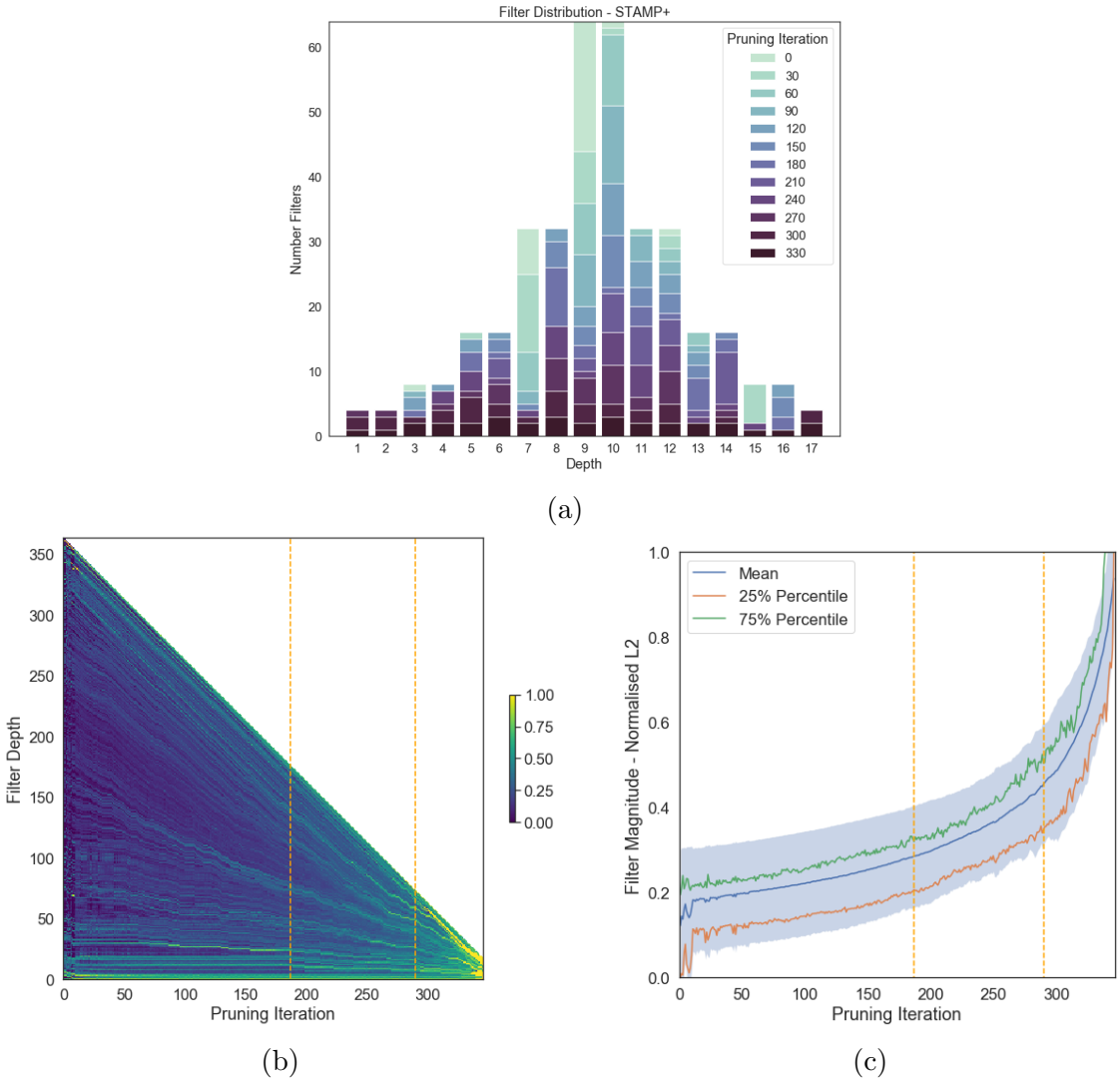


Figure 10: a) Shows the distribution of the filters with network depth as the model is pruned, meaning that the darker the shade of the filter block, the longer the filters at that depth were that were maintained in the models. b) Shows the magnitudes of the activations, averaged across the training data, maintained in the model as it is pruned. It can be seen that the lower magnitude activations are pruned first and the average value of the activation increases as the model is pruned. The first vertical dashed line at 182 filters corresponds to the distribution of filters which gave the best performance on the testing data, and the second dashed line corresponds to the distribution of the filters for the smallest model that was able to complete the segmentation successfully, with no substantial difference in performance from the best performing model. c) Shows the average filter magnitude, and the lower and upper quartile bounds with pruning iteration. It can be seen that the average value increases consistently with pruning iteration.

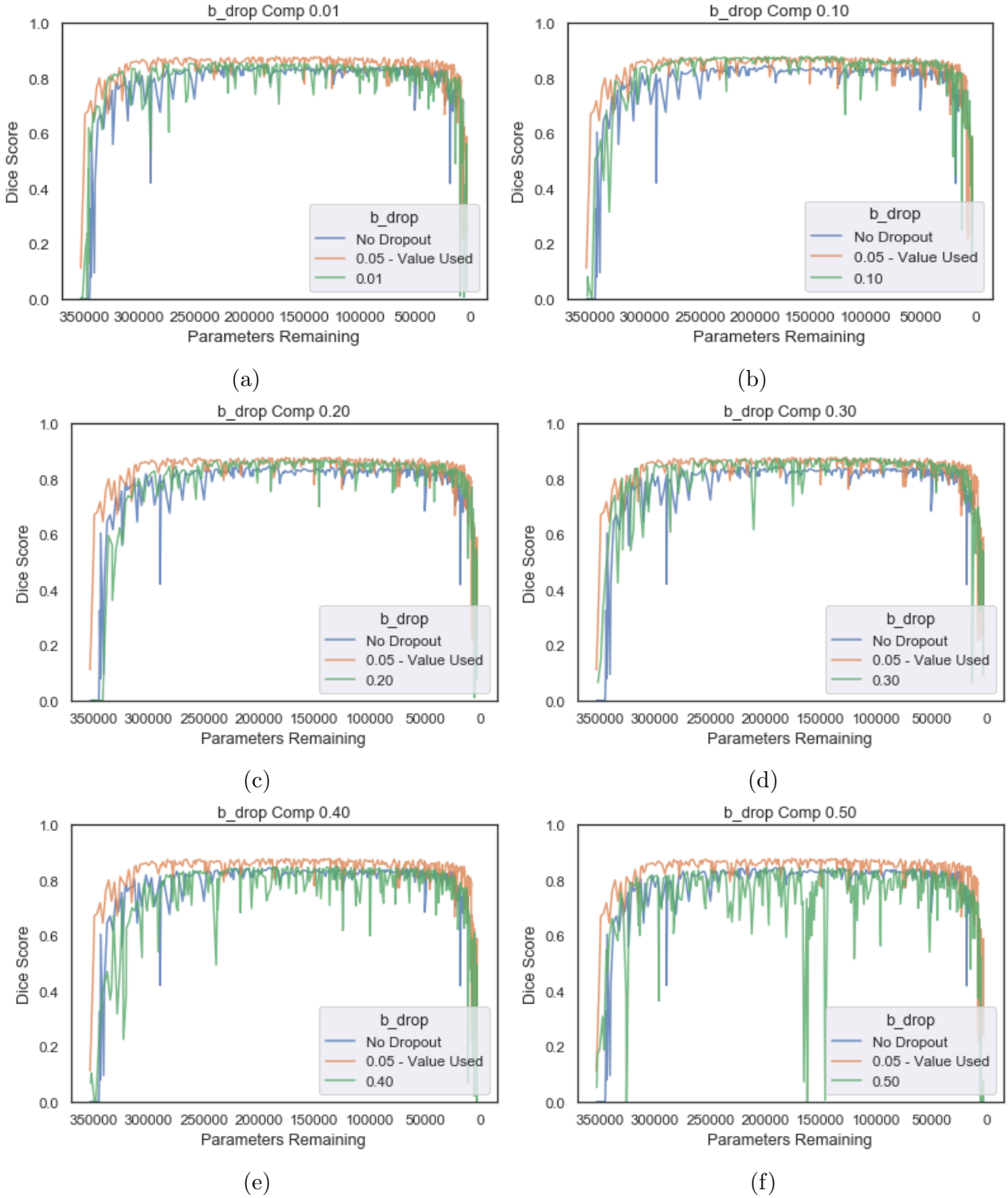


Figure 11: Here, the effect of changing the value of b_{drop} is explored. For clarity, three versions are compared in each panel: STAMP (no dropout) (blue), the base dropout value we used across the experiments (0.05 - in orange), and the base dropout value of interest for that panel (green).

that the maximum dropout probability that can be applied to a layer is 0.05. This is a lot lower than the dropout probability used with parameterwise dropout [43] where values as high as 0.5 can be used; however, spatial or channelwise dropout is much more aggressive and, as shown in Fig. 8, the addition of targeted dropout with $b_{drop} = 0.05$ made a significant ($p < 0.001$) improvement to the model performance. It can be seen in Fig. 11 that very low values (for instance 0.01 shown in Fig. 11a) are too low to give the benefit of the addition of the targeted dropout. Very similar performance was observed for a large range of dropout values $0.05 - 0.30$, showing that the method is not too dependent on the value chosen. As the b_{drop} value is increased to very high values, $0.40 - 0.50$, it can be seen, as would be expected, that the model becomes unstable and the performance degrades. Therefore, the results are sensitive to the choice of the b_{drop} value, but there is a large range of values for which good performance is achieved and there is clear benefit from its addition. The value 0.05 is used across the experiments shown in this chapter; it was found that this produced good results in all cases.

3.1.7. Number of Recovery Epochs

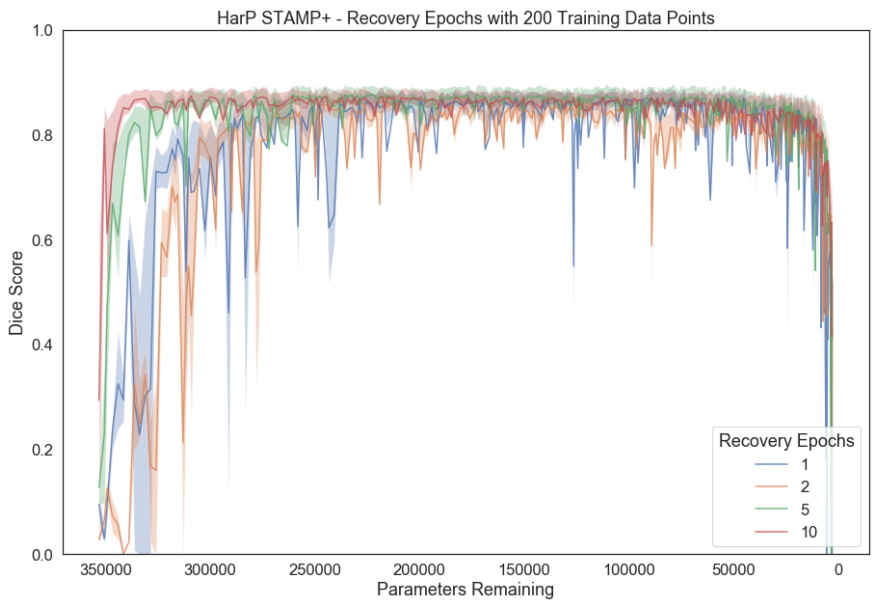
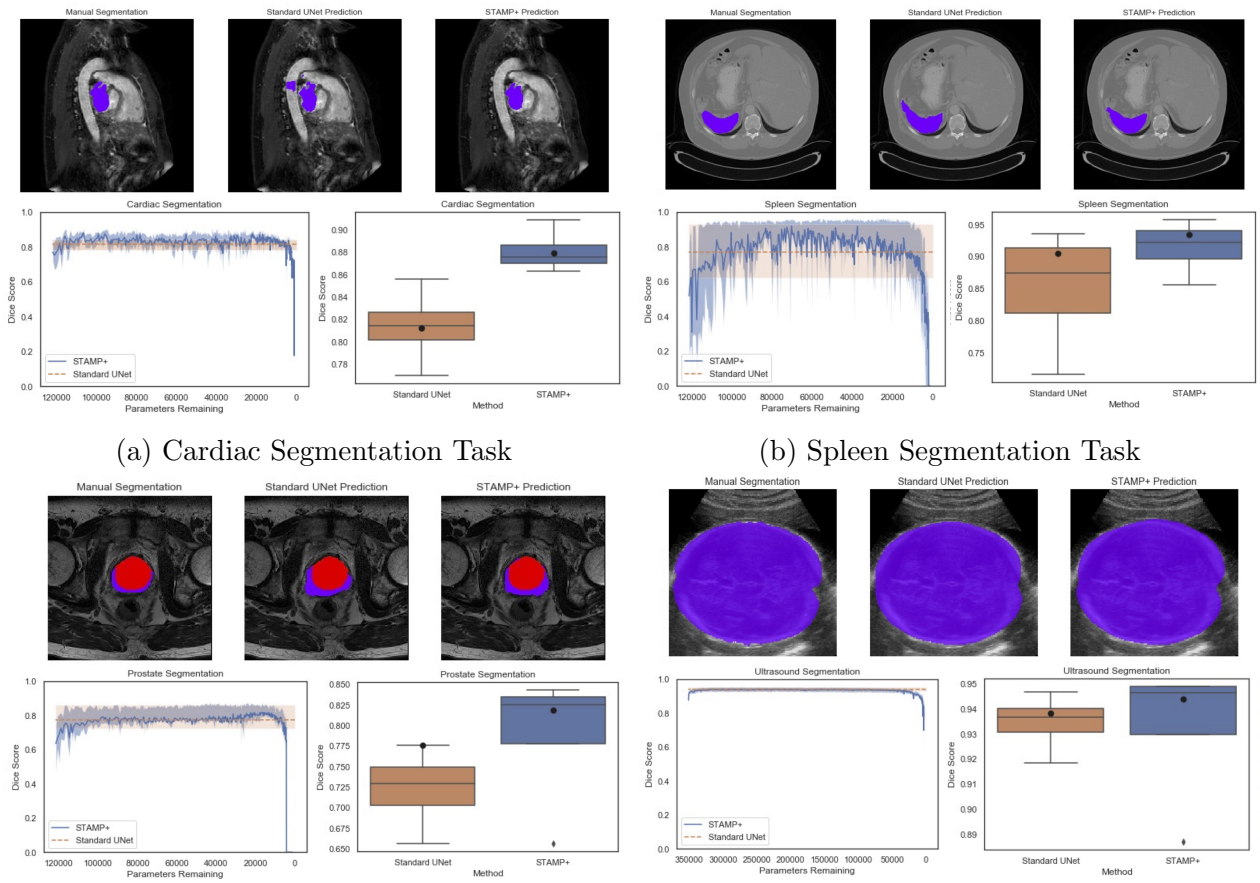


Figure 12: Performance of the STAMP+ framework with increasing numbers of recovery epochs between filter prunings.

STAMP+ requires iterating between training the network and pruning filters one at a time. We therefore explore the effect of the number of recovery epochs between filter prunings, when the network is allowed to train without pruning. Figure 12 shows the performance of the network with pruning for increasing numbers of recovery epochs. It can be seen that the maximum performance reached was very similar in all cases ($1ep = 0.870, 2ep = 0.862, 5ep = 0.876, 10ep = 0.874$, with the best performing model selected using the validation data). The biggest difference between runs as the recovery epochs is changed

is the stability of the training, with the stability in performance between pruning iterations increasing with the number of recovery epochs. Therefore, the setting of the recovery epochs is a trade-off between the stability and the duration of the training.

3.2. Application to Other Datasets



(a) Cardiac Segmentation Task

(b) Spleen Segmentation Task

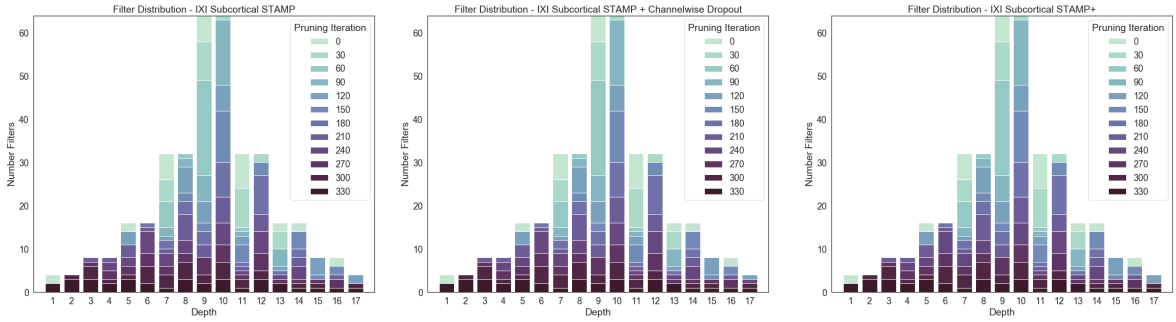
(c) Prostate Segmentation Task

(d) Ultrasound Segmentation Task

Figure 13: Example segmentation and comparison of the performance of the *Standard UNet* and the proposed STAMP+ for four different datasets. The segmentation shown is indicated by the filled circle on the boxplot.

To test the robustness of the chosen hyperparameter values, the same settings were applied to a range of datasets and tasks. Therefore, the following results were all run with the hyperparameters set as: number of recovery epochs $= 2$, $b_{drop} = 0.05$, learning rate $= 1 \times 10^{-2}$, $f = 4$.

The results can be seen in Fig. 13, where the ground-truth segmentations and predictions, and the segmentation performance with pruning for each task can be seen. The results for the datasets from the Medical Decathlon are the values calculated for the reconstructed 3D volumes, so that they are comparable with the results reported in the Medical Decathlon Challenge. The predictions from STAMP+ are the results on the held out test data from the



(a) (b) (c)

Figure 14: (a) *STAMP*, (b) *STAMP + Channelwise dropout out* (c) *STAMP+* for subcortical segmentation on the IXI data. The darker the filter, the longer it was maintained in the model. It can be seen that all the methods led to a very similar pattern of pruning of the filters, showing that the adaptive dropout does not drive which filters are pruned, merely makes the model more robust to being pruned.

best model, selected using the performance on the validation data. It can be seen that despite the hyperparameter values not being optimised for these tasks, the proposed approach is able to successfully train and prune the network simultaneously, producing networks that match, or out-perform, the *Standard UNet*. This shows that the chosen hyperparameters are robust across tasks, dataset size, and image modality, and therefore indicates that the approach should be applicable for a wide range of medical imaging tasks.

3.3. Application to IXI Subcortical Segmentation - Ablation Study

The task of segmenting three deep grey matter structures was also considered: the caudate, putamen and thalamus. Without data augmentation, it was not possible to train a *Standard UNet* to successfully segment all three regions; however, it was possible to train the STAMP+ method to segment all three regions. Therefore, an ablation study was performed to explore the effect of the pruning and the targeted dropout on the performance of the model. Therefore, the following methods were tested: *Standard UNet*, *Standard UNet* with channelwise dropout, *Standard UNet* with targeted dropout, STAMP, STAMP with channelwise dropout (STAMP+D), and STAMP+. For the models with spatial dropout, the dropout probability was set to the average value of the targeted dropout values.

First, the training distributions for the three pruning studies can be seen in Fig. 14. This shows that the filter distributions as the models are pruned are very similar. This demonstrates that the targeted dropout is not driving the pruning process, with all methods showing the same pattern, and so the pruning pattern is being driven by the data. The improvement seen through the addition of the targeted dropout is due to the dropout making the model more robust to the removal of filters.

Figures 15 and 16 show the results of the ablation study, with Fig. 15 showing a representative subject from the test set of the Guy's dataset, and Fig. 16 showing the box plots for each site, averaged over the three subcortical regions. It can be seen from the example segmentations that the *Standard UNet* fails to segment all of the regions and even the addition of the targeted dropout, whilst it leads to a small amount of the third region

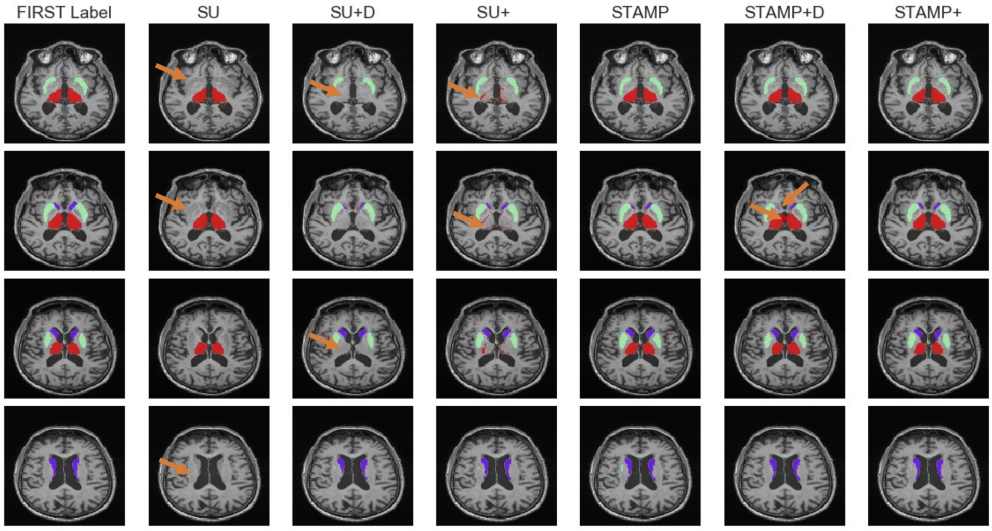


Figure 15: Example segmentations for the caudate, thalamus and putamen trained using the Guy's data from the IXI dataset for, left to right, starting at the second column: *Standard UNet* (SU), *Standard UNet* with channelwise dropout (SU + D), *Standard UNet* with targeted dropout (SU+), STAMP, STAMP with channelwise dropout (STAMP+D), and STAMP+. The labels generated using the FSL FIRST tool (shown in the first column) are used as a proxy for manual labels. Arrows indicate regions with clear differences between approaches.

being segmented, does not lead to a substantial improvement in the performance of the network. Therefore, in this low-data regime and for a relatively difficult task, where the *Standard UNet* fails to complete the segmentation to a standard that would be acceptable, the pruning process alone creates a network that is significantly better at segmenting the three regions. Across all folds, the *Standard UNet* found a local minimum which led to one of the regions being segmented very well at the cost of another not being segmented at all. This is also the mode of failure of the pruned network when the amount of training data is reduced to a very low number of subjects, as is explored in the following section. The region that is not segmented does, however, vary between folds and is not necessarily the smallest region.

It can also be seen that while the improvement provided by the targeted dropout is relatively small, it is significant ($p = 0.0007$) and, in addition, increases the stability of the STAMP+ procedure. There is also a significant improvement compared to using a fixed amount of spatial dropout across the networks ($p < 0.001$): this might be expected as this strategy does not encode information about which filters are the most likely to be removed, so there is a chance that filters that are very important to the performance would be dropped, especially as the pruning progresses, hindering the model training. The value of spatial dropout used was very low (0.037, calculated as the average p value from STAMP+ and with a higher value the performance of the network was hindered further.

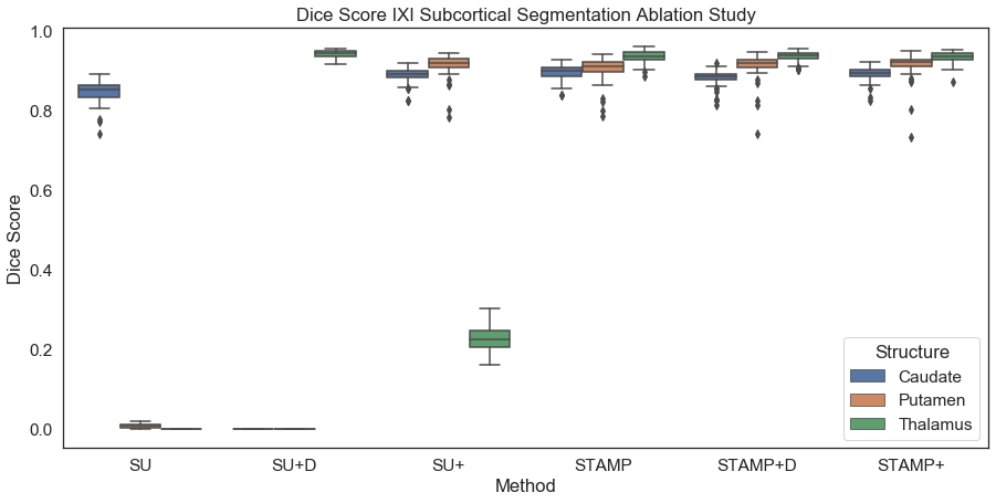


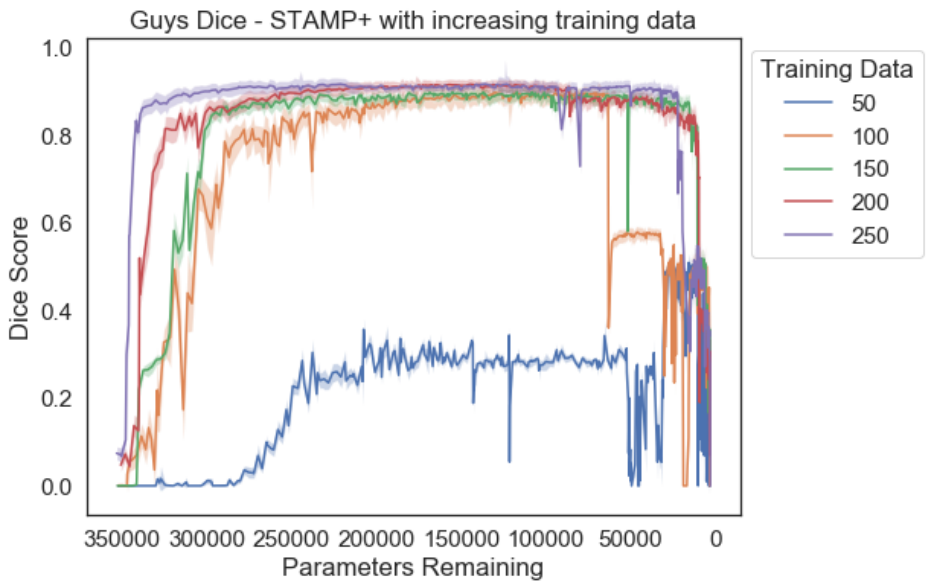
Figure 16: Dice results for the IXI subcortical segmentation ablation study, comparing *Standard UNet* (SU) , *Standard UNet* with channelwise dropout (SU+D), *Standard UNet* with targeted dropout (SU+TD), STAMP, STAMP with channelwise dropout (STAMP+D) and STAMP+. It can be seen that even the simplest pruned model, STAMP, outperforms all the models trained without pruning, which were unable to segment all three labels, as seen in Fig. 15. STAMP+ slightly but significantly out performed the other approaches for the Dice score averaged over the three regions.

3.3.1. Amount of Training Data

The poor performance of the *Standard UNet* on this task is almost certainly due to the shortage of training data available for the task. Access to more data for training is not an option in this instance, and so how much more data is required to improve the performance cannot be explored. However, reducing the amount of training data to explore the performance of the STAMP+ method can be explored. Figure 17 shows the Dice scores on the testing set, averaged across the three subcortical regions. The results broken down by subcortical region for the *Standard UNet* and STAMP+ can be found in the supplementary material. Both sets of results show that the pruned network can be successfully trained with substantially less data than the *Standard UNet* and, in all cases, provides a better performance. This is not just a function of the reduced number of parameters, as we were unable to train a *Standard UNet* with $f = 2$ on the data. Some of the increase in performance might be due to the network learning coadapted relationships between layers and overfitting less to the very small amount of training data available. Therefore, STAMP+ clearly provides advantages for segmentation tasks when working in low-data regimes, as is very common in medical imaging applications.

3.4. Training in Low Data Regimes

Having shown that the STAMP+ algorithm has worked in a low data regime when the *Standard UNet* was not able to create acceptable segmentations, training in low data regimes was now more systematically explored. Clearly, as would be expected, the performance of the method improves with increasing amounts of training data, as shown in Fig. 18, with the maximum Dice score increasing and the stability of the score between pruning iterations



(a)

Figure 17: Dice scores averaged across the three subcortical regions for increasing numbers of available training images for the IXI data. 250 represents the full training set.

also increasing. The network is able to produce Dice scores of over 0.82 even when presented with only 25 training examples.

3.4.1. Comparison to Standard UNet and PruneFinetune

Figure 19 then compares STAMP+ to the *Standard UNet* for increasing numbers of training subjects, for models with $f = \{2, 4, 8\}$. It can be seen that for all sizes of model, and for all amounts of data, the STAMP+ model still outperforms the *Standard UNet*, demonstrating its utility in low data regimes.

STAMP+ is then compared to *PruneFinetune* for increasing numbers of training subjects. In Fig. 20 the comparison can be seen for $25, 50$ and 100 training subjects. We found that when the number of training subjects is very low (for instance, consider 25 training subjects) the original *Standard UNet* model performed badly on the data. As this model is then the initialisation for the *PruneFinetune* pruning and there are still very few subjects available for training, the model struggles to perform well from the poor starting point. The STAMP+ model has noisier performance between pruning iterations, but substantially outperforms the *PruneFinetune* approach. Even as the number of training subjects increases, STAMP+ outperforms *PruneFinetune*.

3.4.2. Number of Recovery Epochs

It was hypothesised that STAMP+ might be more sensitive to the number of recovery epochs between pruning iterations when working in a low data regime. Therefore, the experiment exploring the effect of the number of recovery epochs was repeated for 25 and 50 training subjects.

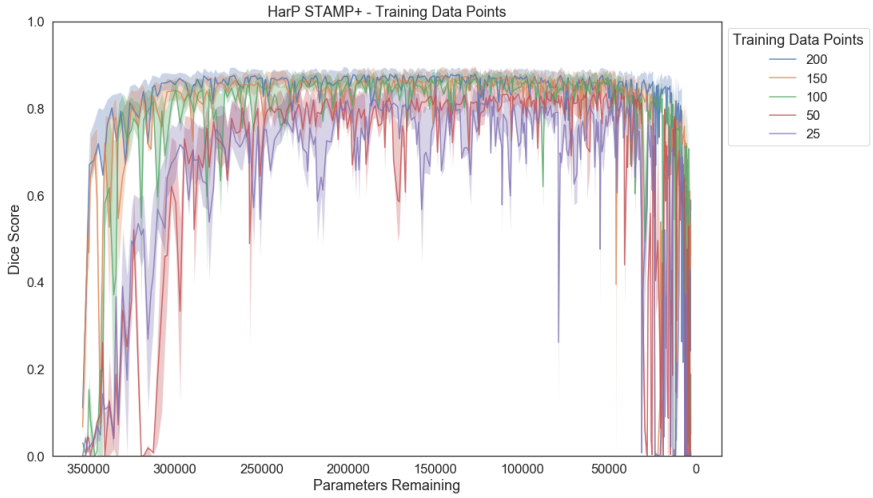


Figure 18: Dice scores for the HarP hippocampal segmentation for increasing amounts of training subjects.

The results can be seen in Fig. 21. It can be seen that when working in a low data regime, the number of recovery epochs between pruning iterations needs to be increased to allow the model to reach good segmentation performance. Therefore, the number of recovery epochs needs to be increased at a cost to the amount of time required to train the model. Despite this, training of the STAMP+ method still takes many fewer training epochs than *PruneFinetune*.

3.4.3. Training in Low Data Regimes – Application to Other Datasets

Finally, training in low data regimes was explored for other datasets. Based on the previous experiment, the number of recovery epochs was set to 5 and the other hyperparameters remained the same as in previous experiments. The results can be seen in Fig. 22. It can be seen that for all the tasks, STAMP+ outperforms the *Standard UNet* when there were low numbers of training subjects, showing the power of the pruning method for working in low data regimes.

4. Discussion

We have presented a method that performs simultaneous training and pruning of a UNet architecture for medical image segmentation. While pruning is well explored for large classification datasets, to the best of our knowledge it has not been explored for medical image segmentation, where different architectures and generally small training datasets create a different challenge for pruning algorithms. Therefore, the STAMP+ method and the addition of Targeted Channelwise Dropout has been explored for a range of segmentation datasets and tasks: we have shown that not only is it possible to create smaller models with as good or increased performance on the task of interest, but also that, through pruning, better performing models can be produced in low data regimes.

Using the HarP data, the method has been validated on data with manual segmentation labels and the effects of the design decisions have been explored. The results indicate that

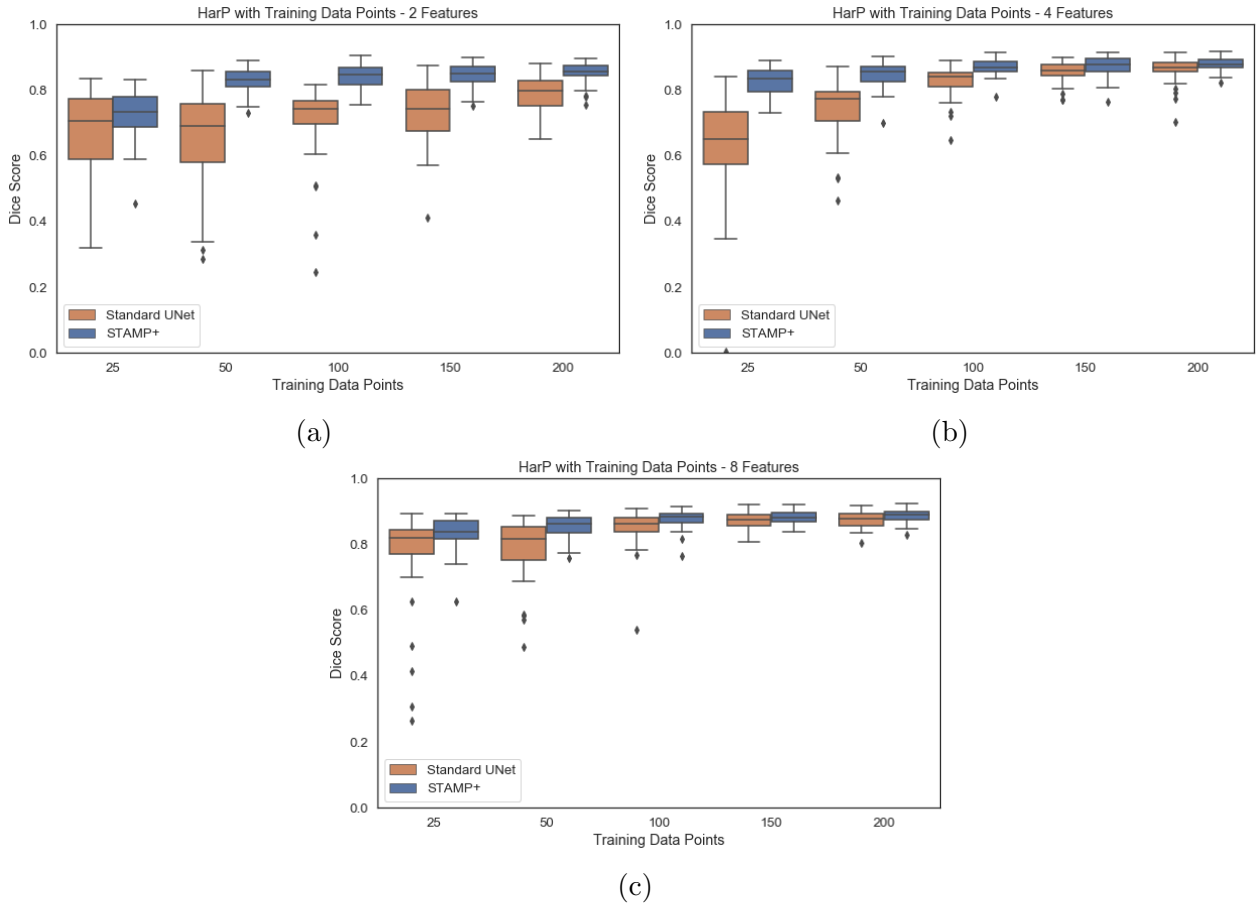


Figure 19: Dice scores for the HarP hippocampal segmentation for increasing amounts of training subjects, comparing STAMP+ and the *Standard UNet*, for $f = \{2, 4, 8\}$.

better performance is achieved by initially choosing a large model and pruning the model to be smaller, rather than beginning with a small model, and that different distributions of the filters are kept by the pruning process than might be naïvely chosen. The experiments also show that the addition of the Targeted Dropout leads to an improvement in performance and that this improvement is present for a relatively wide range of b_{drop} values (0.05 to 0.30).

It has been demonstrated that the hyperparameters used are robust across a range of tasks and imaging modalities, for both 3D and 2D problems, showing that the method is applicable across a range of medical imaging tasks, and may be used in many cases without the need for extensive parameter searching or optimisation. That is, the default parameters led to the same or improved performance compared to the *Standard UNet* for all cases. Furthermore, the increase in performance in low data regimes has been demonstrated across these different datasets and tasks. The ablation study on the IXI dataset indicates that the improvement is not just a function of the pruning or the targeted dropout, but a combination of the two aspects.

Whilst care has been taken to test the method across modalities and segmentation tasks,

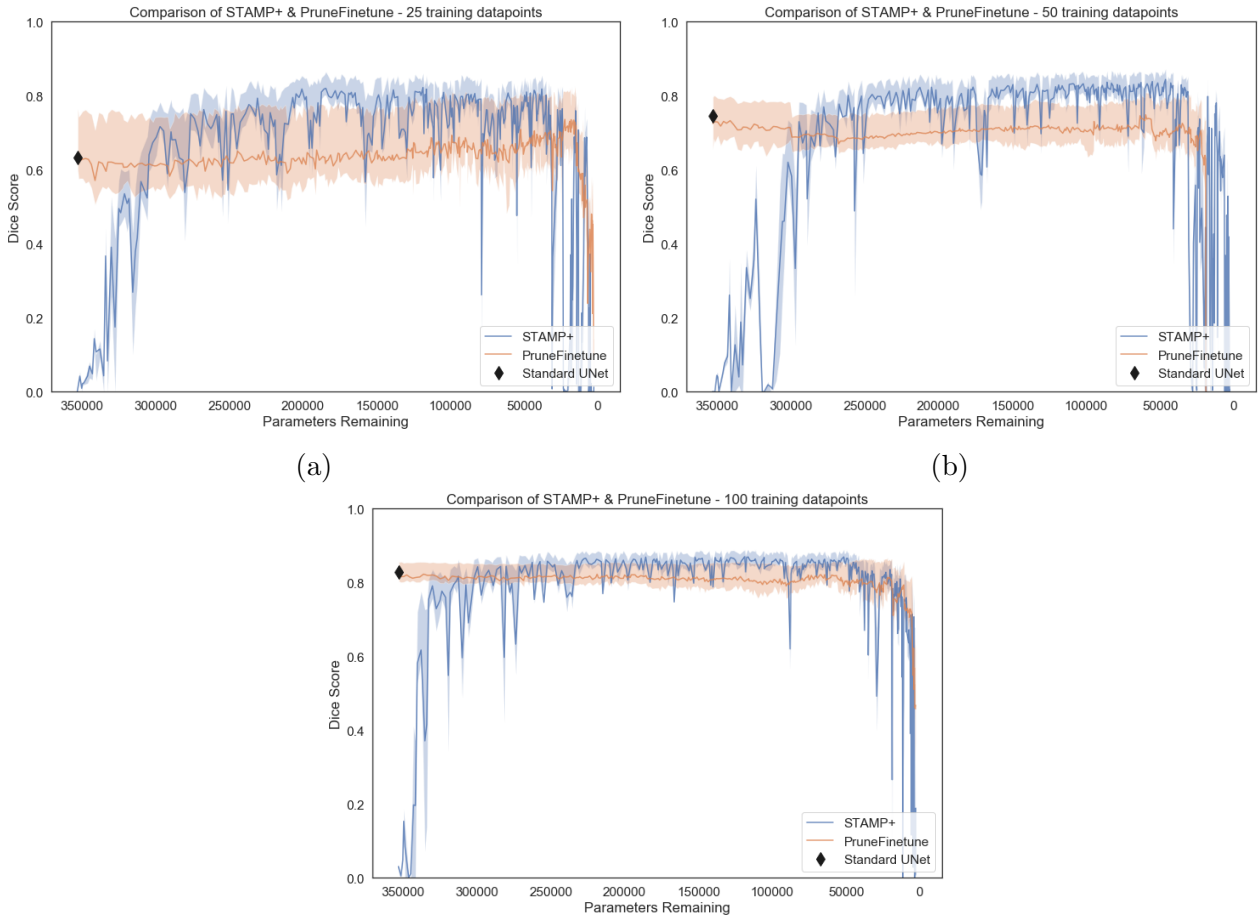


Figure 20: Dice scores for the HarP hippocampal segmentation comparing STAMP+ and *PruneFinetune* for a) 25, b) 50 and c) 100 training subjects.

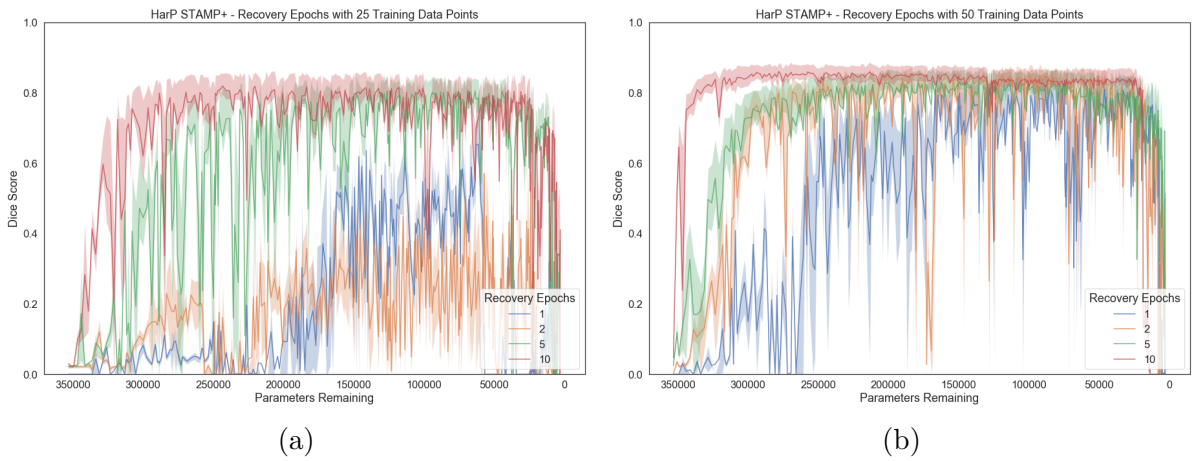
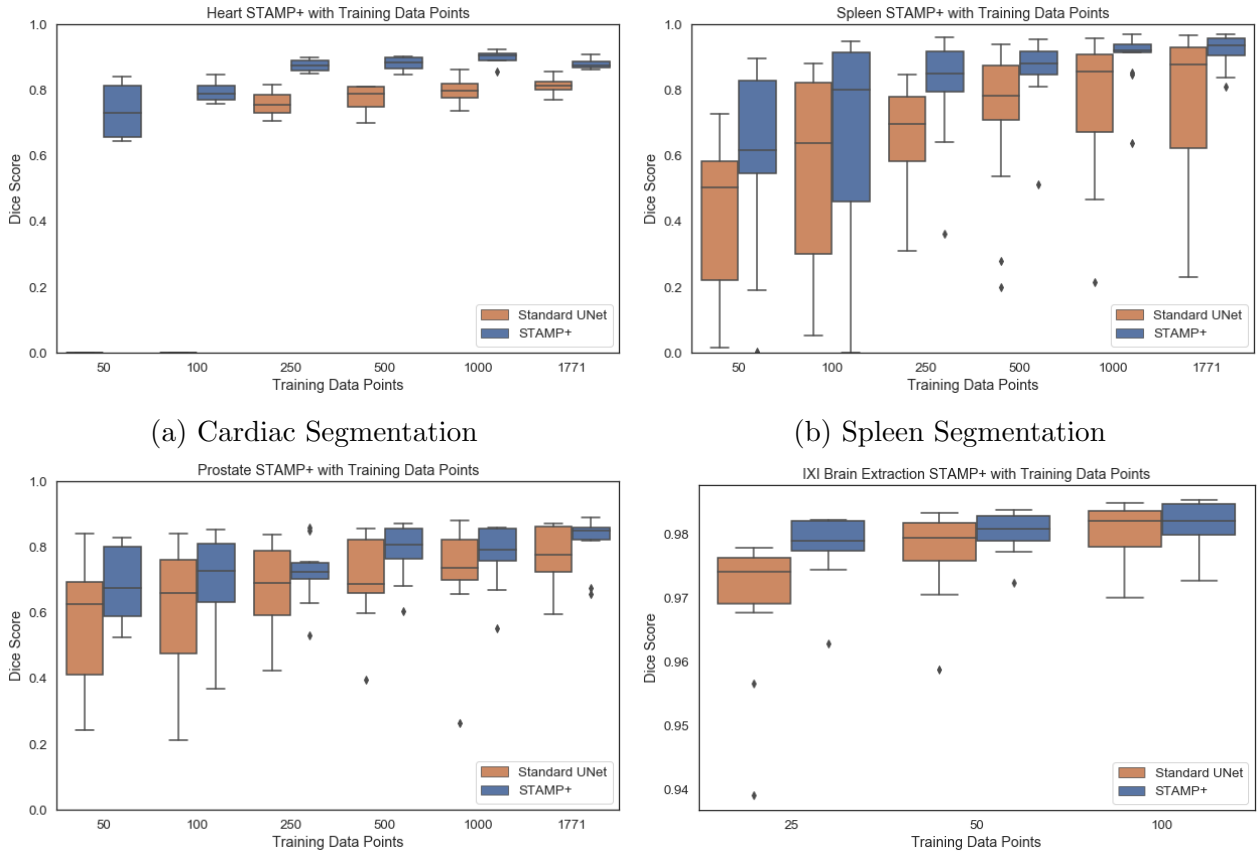


Figure 21: Dice scores for the HarP hippocampal segmentation when exploring the effect of the number of recovery epochs for a) 25 and b) 50 training subjects.



(a) Cardiac Segmentation

(b) Spleen Segmentation

(c) Prostate Segmentation

(d) Brain Extraction

Figure 22: Dice scores for training low numbers of subjects comparing STAMP+ and the *Standard UNet* across segmentation tasks and modalities. Note that the *Standard UNet* was unable to complete the segmentation for the two lowest numbers of data points with the cardiac dataset.

the labels used for the IXI dataset were generated using automated tools. The two tools used – FSL FIRST and ANAT – employ differing model assumptions so the results should be valid for comparison between methods; however, it does potentially limit the maximum achievable performance due to the imperfections in the labels.

The method has also only been explored for the standard UNet architecture. This decision was made as the UNet is the most popular architecture for medical image segmentation, and the majority of methods either use the UNet architecture or derivatives thereof. It is expected that the results would generalise to other similar networks, but this has not been explored explicitly within this work.

Finally, a potential limitation of the method is that the performance between pruning iterations is unstable, especially compared to the *PruneFinetune* method. Across this work, the best model was evaluated using the validation data, and this has corresponded well with good performance on the testing data – although it does not always corresponding to the highest performing iteration on the testing data. This has been true across the datasets

explored here but would not necessarily be true if the method were to be applied to other datasets. The use of the targeted dropout helps to reduce this by increasing the stability between iterations, but if this were not seen to be the case for a given dataset, it may be necessary to increase the number of recovery epochs to increase the stability of the model training. It is probable that a lot of the instability is due to working in the low data regime, as the stability visibly increases as the number of training subjects is increased. In practice a large number of recovery epochs could be used, and gradually reduced to smaller numbers if the training were stable, and could be simply automated.

5. Conclusion

We have developed and demonstrated a method to allow simultaneous training and pruning of a UNet architecture. We have shown that in low data regimes this outperforms the equivalent *Standard UNet* models and the standard pruning method due to not needing to train the original full-sized model. The channelwise targeted dropout assists the pruning, by making the model more robust to being pruned. The method has been demonstrated across different organs, segmentation tasks and imaging modalities and it is expected that the results should generalise to other networks architectures.

6. Data and Code Availability Statement

The data used in these experiments are available from the relevant studies. The code is available at `github.com/nkdinsdale/STAMP` (released on acceptance). Weights from training are available on request through emailing the corresponding author.

7. Acknowledgements

ND is supported by the Engineering and Physical Sciences Research Council (EPSRC) and Medical Research Council (MRC) [grant number EP/L016052/1]. MJ is supported by the National Institute for Health Research (NIHR), Oxford Biomedical Research Centre (BRC), and this research was funded by the Wellcome Trust [215573/Z/19/Z]. The Wellcome Centre for Integrative Neuroimaging is supported by core funding from the Wellcome Trust [203139/Z/16/Z]. AN is grateful for support from the UK Royal Academy of Engineering under the Engineering for Development Research Fellowships scheme.

The computational aspects of this research were supported by the Wellcome Trust Core Award [Grant Number 203141/Z/16/Z] and the NIHR Oxford BRC. The views expressed are those of the author(s) and not necessarily those of the NHS, the NIHR or the Department of Health.

8. References

[1] Robin Flores, Renaud La Joie, and Gaël Chételat. Structural imaging of hippocampal subfields in healthy aging and Alzheimer's disease. *Neuroscience*, 309, 08 2015. doi: 10.1016/j.neuroscience.2015.08.033.

[2] O. Ronneberger, P.Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015.

[3] Ozgun Cicek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3D U-Net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2016*, pages 424 – 432, 2016. doi: 10.1007/978.3.319.46723.8.49.

[4] F. Milletari, N. Navab, and S. Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 565–571, 2016. doi: 10.1109/3DV.2016.79.

[5] Abhijit Guha Roy, Sailesh Conjeti, Nassir Navab, and Christian Wachinger. QuickNAT: A fully convolutional network for quick and accurate segmentation of neuroanatomy. *NeuroImage*, 186, 11 2018. doi: 10.1016/j.neuroimage.2018.11.042.

[6] G. Balakrishnan, Amy Zhao, M. Sabuncu, J. Guttag, and Adrian V. Dalca. VoxelMorph: A learning framework for deformable medical image registration. *IEEE Transactions on Medical Imaging*, 38: 1788–1800, 2019.

[7] Fabian Isensee, Paul Jaeger, Simon Kohl, Jens Petersen, and Klaus Maier-Hein. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods*, 18:1–9, 02 2021. doi: 10.1038/s41592-020-01008-z.

[8] Nicola K. Dinsdale, Mark Jenkinson, and Ana I. L. Namburete. Spatial warping network for 3d segmentation of the hippocampus in mr images. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, pages 284–291, 2019. doi: 10.1007/978-3-030-32248-9.32.

[9] Jo Schlemper, Ozan Oktay, Michiel Schaap, Mattias Heinrich, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention gated networks: Learning to leverage salient regions in medical images. *Medical Image Analysis*, 53:197–207, 2019. doi: https://doi.org/10.1016/j.media.2019.01.012.

[10] M. Yahyatabar, P. Jouvet, and F. Cheriet. Dense-Unet: a light model for lung fields segmentation in chest X-ray images. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, pages 1242–1245, 2020. doi: 10.1109/EMBC44109.2020.9176033.

[11] S. Vaze, W. Xie, and A. I. L. Namburete. Low-memory CNNs enabling real-time ultrasound segmentation towards mobile deployment. *IEEE Journal of Biomedical and Health Informatics*, 24(4):1059–1069, 2020. doi: 10.1109/JBHI.2019.2961264.

[12] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning with applications to medical imaging. *33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*, 02 2019.

[13] Jakub Nalepa, Michal Marcinkiewicz, and Michal Kawulok. Data augmentation for brain-tumor segmentation: A review. *Frontiers in Computational Neuroscience*, 13:83, 2019. doi: 10.3389/fncom.2019.00083. URL https://www.frontiersin.org/article/10.3389/fncom.2019.00083.

[14] Yann Le Cun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems 2*, page 598–605, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. ISBN 1558601007.

[15] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. In *International Conference on Learning Representations*, 11 2016.

[16] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, page 1135–1143, 2015. doi: 10.5555/2969239.2969366.

[17] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 11 2019.

[18] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *ArXiv*, 07 2016.

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and H.P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, 08 2017.

[20] Suraj Srinivas and R. Venkatesh Babu. Data-free parameter pruning for Deep Neural Networks. *Proceedings of BMVC*, July 2015. doi: dx.doi.org/10.5244/C.29.31.

[21] Guillaume Bellec, David Kappel, Wolfgang Maass, and Robert Legenstein. Deep rewiring: Training very sparse deep networks. 11 2017.

[22] Shibo Shen, Rongpeng Li, Zhifeng Zhao, Honggang Zhang, and Yugeng Zhou. Learning to prune in training via dynamic channel propagation. *ArXiv*, 2020. doi: arXiv:2007.01486.

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS 2014 Deep Learning Workshop*, 03 2014. doi: arXiv.

[24] Andrew Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, 04 2017. doi: arXiv:1704.04861.

[25] Joseph Cohen, Henry Lo, and Wei Ding. Randomout: Using a convolutional gradient norm to win the filter lottery. In *ICLR Workshop*, 02 2016.

[26] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

[27] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). *Technical Report*, 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

[28] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[29] Aidan Gomez, Ivan Zhang, Kevin Swersky, Yarin Gal, and Geoffery Hinton. Targeted dropout. *Advances in Neural Information Processing Systems 32 (NIPS 208)*, 2018.

[30] Saihui Hou and Zilei Wang. Weighted channel dropout for regularization of deep convolutional neural network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:8425–8432, 07 2019. doi: 10.1609/aaai.v33i01.33018425.

[31] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. JMLR Workshop and Conference Proceedings, 13–15 May 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.

[32] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 648–656, 2015. doi: 10.1109/CVPR.2015.7298664.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456. PMLR, 07–09 Jul 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

[34] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32 (NIPS 2019)*, 2019.

[36] Giovanni Frisoni and Clifford Jack. Harp: The eadc-adni harmonized protocol for manual hippocampal segmentation. a standard of reference from a global working group. *Alzheimer's and Dementia*, 11, 02 2015. doi: 10.1016/j.jalz.2014.05.1761.

[37] C. Tobon-Gomez, A. J. Geers, J. Peters, J. Weese, K. Pinto, R. Karim, M. Ammar, A. Daoudi, J. Margeta, Z. Sandoval, B. Stender, Y. Zheng, M. A. Zuluaga, J. Betancur, N. Ayache, M. A. Chikh, J. L. Dillenseger, B. M. Kelm, S. Mahmoudi, S. Ourselin, A. Schlaefer, T. Schaeffter, R. Razavi, and K. S. Rhode. Benchmark for algorithms segmenting the left atrium from 3d ct and mri datasets. *IEEE Transactions on Medical Imaging*, 34(7):1460–1473, 2015. doi: 10.1109/TMI.2015.2398818.

[38] Amber Simpson, Julie Leal, Amudhan Pugalenthi, Peter Allen, Ronald DeMatteo, Y. Fong, Mithat Gönen, William Jarnagin, T Kingham, Michael Miga, Jinru Shia, Martin Weiser, and Michael D'Angelica. Chemotherapy-induced splenic volume increase is independently associated with major complications after hepatic resection for metastatic colorectal cancer. *Journal of the American College of Surgeons*, 220, 12 2014. doi: 10.1016/j.jamcollsurg.2014.12.008.

[39] Geert Litjens, Oscar Debats, Wendy van de Ven, Nico Karssemeijer, and Henkjan Huisman. A pattern recognition approach to zonal segmentation of the prostate on mri. volume 15, pages 413–20, 10 2012. ISBN 978-3-642-33417-7. doi: 10.1007/978-3-642-33418-4_51.

[40] A.T. Papageorghiou, Eric Ohuma, Douglas Altman, Tullia Todros, Leila Ismail, Ann Lambert, Yasmin Jaffer, Enrico Bertino, Michael Gravett, Manorama Purwar, Julia Noble, Ruyan Pang, Cesar Victora, Fernando Colugnati, Maria Carvalho, Laurent Salomon, Zulfiqar Bhutta, Stephen Kennedy, and José Villar. International standards for fetal growth based on serial ultrasound measurements: The fetal growth longitudinal study of the intergrowth-21st project. *The Lancet*, 384:869–879, 09 2014. doi: 10.1016/S0140-6736(14)61490-2.

[41] Brian Patenaude, Stephen Smith, David Kennedy, and Mark Jenkinson. A bayesian model of shape and appearance for subcortical brain segmentation. *NeuroImage*, 56:907–22, 02 2011. doi: 10.1016/j.neuroimage.2011.02.046.

[42] Ali Gholipour, Caitlin Rollins, Clemente Velasco-Annis, Abdelhakim Ouaalam, Alireza Akhondi-Asl, Onur Afacan, Cynthia Ortinau, Sean Clancy, Catherine Limperopoulos, Edward Yang, Judy Estroff, and Simon Warfield. A normative spatiotemporal mri atlas of the fetal brain for automatic segmentation and analysis of early brain growth. *Scientific Reports*, 7, 12 2017. doi: 10.1038/s41598-017-00525-w.

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. doi: 10.5555/2627435.2670313.

9. Supplementary Material

9.1. Shallower Network

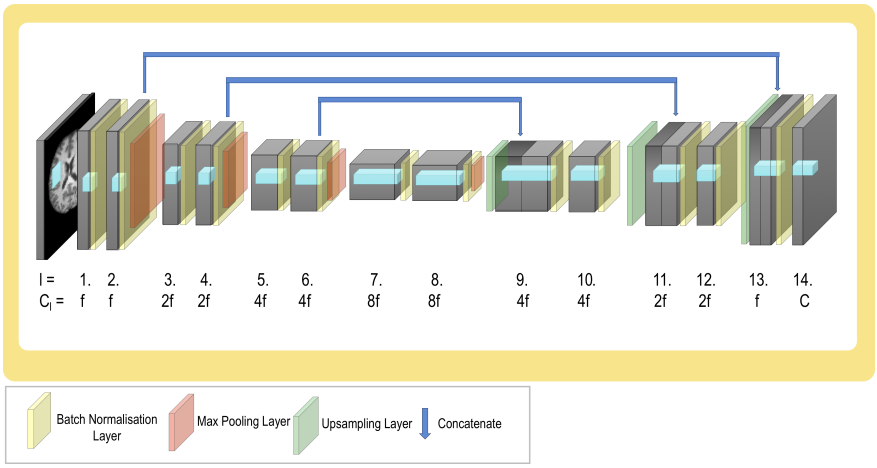


Figure 23: Shallower UNet model architecture - it follows the standard pattern of halving resolution and doubling filters at each depth but has less depth than the UNet architecture previously considered. l corresponds to the layer depth, C_l is the number of channels in that layer and C is the number of classes in the output segmentation.

To allow further exploration of the effect of the initial size of the model, pruning a shallower original model is also considered. A downsampling and an upsampling layer are removed, so the original model is smaller, with fewer parameters, but still having the ability to learn the same number of features at the layers of abstraction that remain. A shallower network is considered, rather than a deeper one, as it allows the exploration of another way in which the model could be smaller in terms of parameters. This architecture can be seen in Fig. 23. All other parameters are held the same as in the experiments with the full-sized model and the value of f will again be varied.

We first investigate results obtained using the HarP data, for different values of $f = [2, 4, 8, 16]$. Figure 24 shows the results, where it can be seen that the results with the shallower UNet follow the same pattern as was seen with the *Standard UNet* (Fig. 6). Again, it is clearly visible that a better performance on the segmentation task is achieved if we begin with a larger model and prune it to be the same size as the smaller model, rather than originally training the smaller model to convergence. The pruning was, however, less stable, showing that the model training was less robust to filters being removed than when the model was deeper.

Fig. 25 compares the results from the shallower UNet to the original UNet when the original models were matched in terms of parameters, where Fig. 25a compares an original depth UNet with $f = 4$ to a shallower UNet with $f = 8$, and Fig. 25b compares an original depth UNet with $f = 8$ to a shallower UNet with $f = 16$. The black diamonds represent the original *Standard UNets* trained to convergence; they perform very similarly on the data. As they are pruned, the two similarly sized networks have very similar performance on the

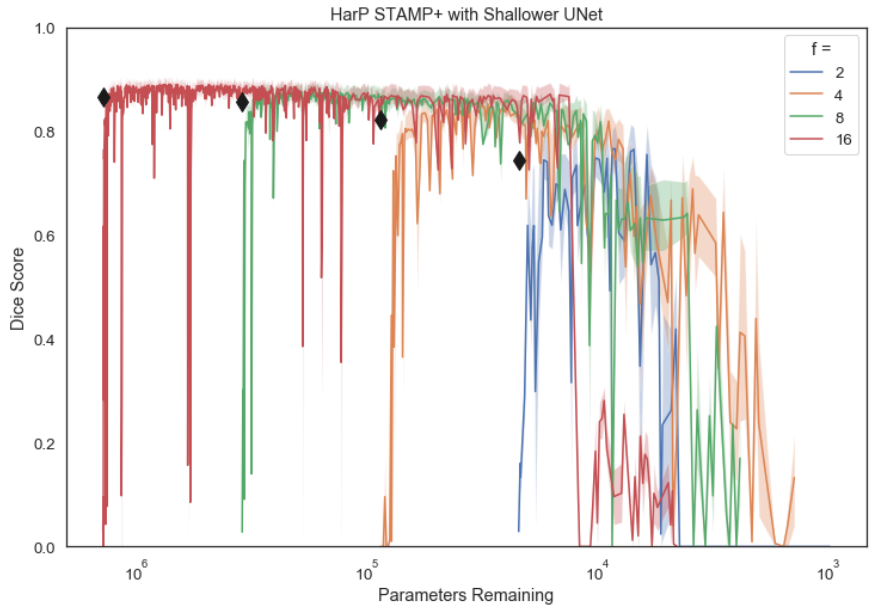
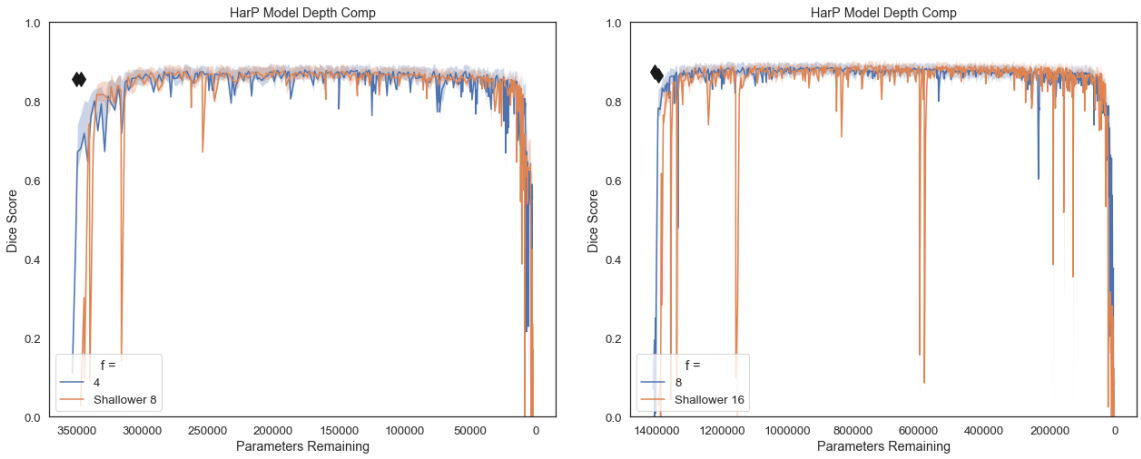


Figure 24: Dice scores on the HarP data using the shallower UNet architecture for $f = [2, 4, 8, 16]$, with interquartile bounds shown, using STAMP+. Diamonds represent a *Standard UNet* with that number of parameters trained to convergence.



(a) (b)

Figure 25: Direct comparison of pruning the original UNet considered – Fig. 3 – and the shallower UNet – Fig. 23 – for the case where the two models have similar numbers of parameters originally: a) compares the UNet with $f = 4$ to the shallower UNet with $f = 8$. b) compares the UNet with $f = 8$ to the shallower UNet with $f = 16$. The black diamonds represent the original *Standard UNets* trained to convergence.

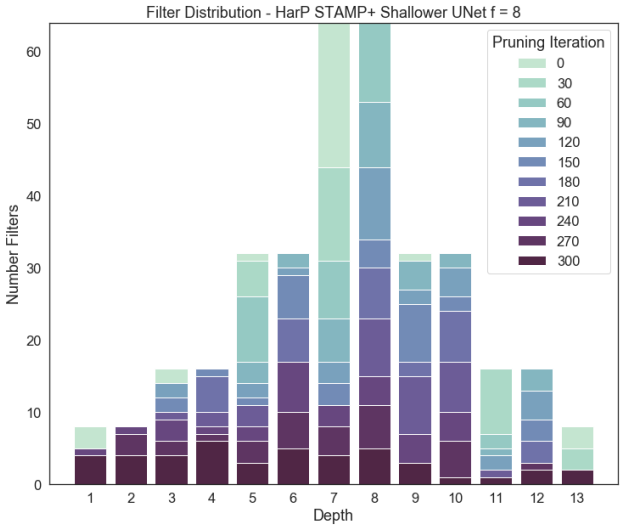
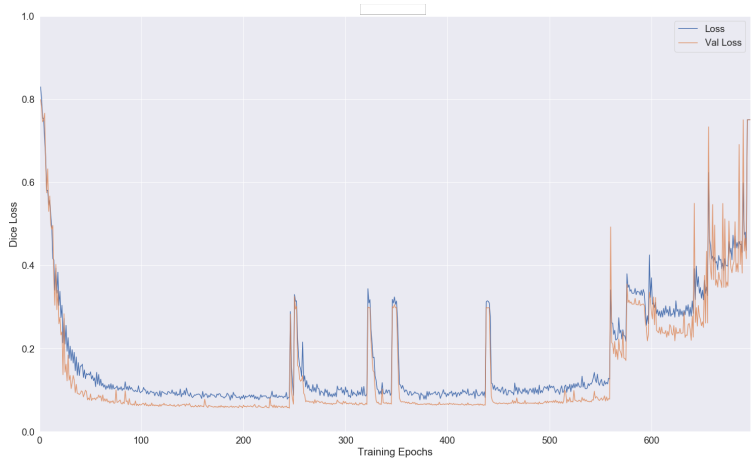


Figure 26: The distribution of the filters with network depth as the shallower model is pruned for hippocampal segmentation on the HarP data, where the darker the shade of the filter block, the longer the filters at that depth were maintained in the model.

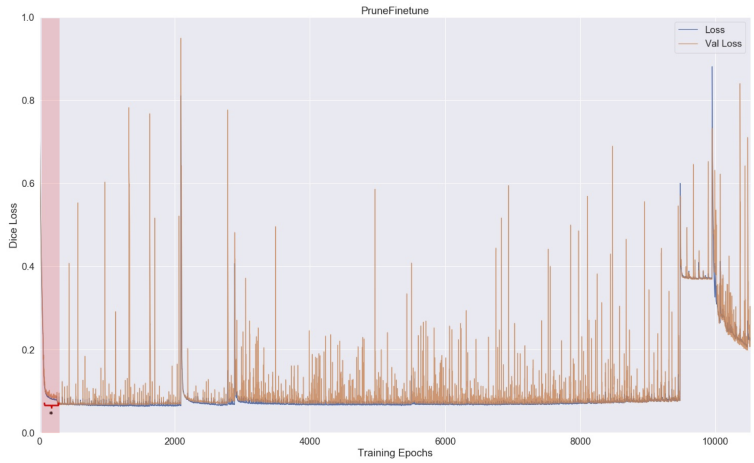
main task, indicating that the exact arrangement of the filters is not vital for the model's performance. This therefore suggests that the algorithm is reasonably robust to the initial model configuration and the results are not due to the initial choice of model architecture. This is also demonstrated by Fig. 26, which shows the distribution of filters during training for the shallower UNet when training with the HarP data. It can be seen that, as is evident in the other examples shown, the bottleneck of the model is pruned aggressively early in the pruning, and the first layer at each depth is pruned first – the result shown is for $f = 8$ but the pattern was the same for all values of f considered.

9.2. Training Graphs

Figure 27 shows representative training curves for the two apporaches to pruning, with a) showing STAMP+ during training and b) *PruneFinetune*. The region indicated in red shows the epochs used to train the initial *Standard UNet* model which was used as the initialisation for the pruning process. It can clearly be seen that STAMP+ requires a fraction of the epochs to train and that the validation performance is much more stable throughout the process.



(a) STAMP+



(b) *PruneFinetune*

Figure 27: Training and validation loss curves for a) STAMP+ and b) *PruneFinetune* for the HarP data. The region indicated in red for *PruneFinetune* indicates the epochs for training the initial UNet model.

9.3. IXI Subcortical Segmentation

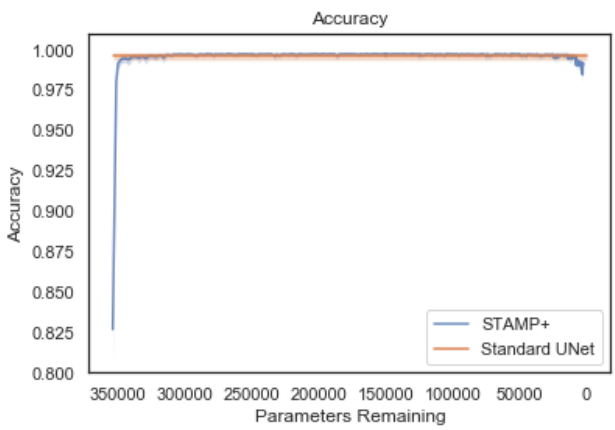
Table 2 shows the results for the IXI subcortical segmentation with increasing amounts of training data, broken down by subcortical region being segmented. It can be seen that when there is insufficient training data to train the STAMP+ method, the failure mode is the same as the *Standard UNet*, with the network failing to segment one of the subcortical regions.

Number of Training Images	Standard UNet			STAMP+		
	Caudate	Putamen	Thalamus	Caudate	Putamen	Thalamus
50	0.28 ± 0.07	0.19 ± 0.03	0.14 ± 0.04	0.78 ± 0.04	0.75 ± 0.04	0.0 ± 0.0
100	0.54 ± 0.12	0.65 ± 0.08	0.75 ± 0.06	0.86 ± 0.02	0.89 ± 0.04	0.93 ± 0.02
150	0.46 ± 0.06	0.78 ± 0.06	0.21 ± 0.07	0.88 ± 0.02	0.90 ± 0.03	0.92 ± 0.02
200	0.032 ± 0.01	0.0 ± 0.0	0.82 ± 0.02	0.90 ± 0.02	0.91 ± 0.03	0.94 ± 0.01
250	0.90 ± 0.03	0.080 ± 0.05	0.0 ± 0.0	0.89 ± 0.02	0.92 ± 0.03	0.94 ± 0.01

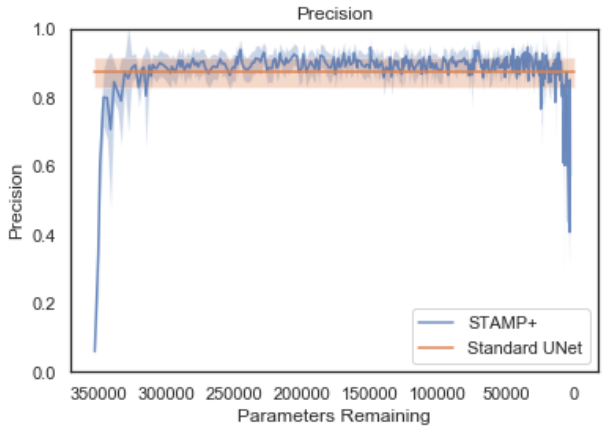
Table 2: Table showing the Dice scores of the different subcortical regions with different amounts of training data for the testing data coming from the Guy's site, comparing *Standard UNet* and STAMP+. The pattern of results was the same for the other two sites.

9.4. Evaluation Metric

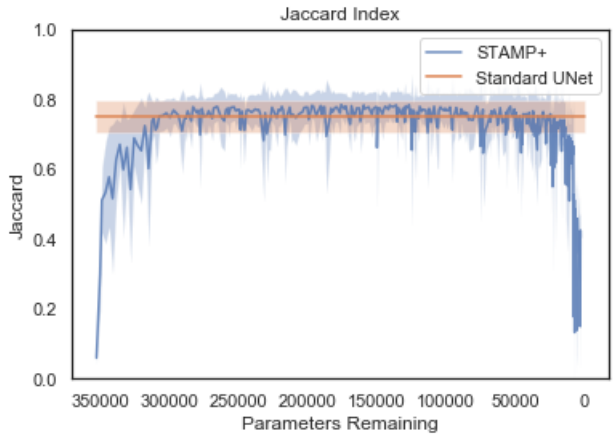
Dice scores were reported throughout this work. Here the results for comparing the *Standard UNet* and STAMP+ are evaluated using different metrics for segmentation performance, and shown in Fig. 28 for the hippocampal segmentation task utilising the HarP data.



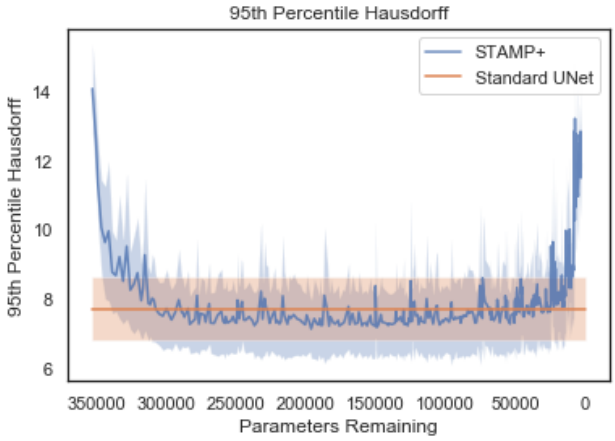
(a) Accuracy



(b) Precision



(c) Jaccard



(d) 95th Percentile Hausdorff

Figure 28: Results with different evaluation metrics, comparing *Standard UNet* and STAMP+.