Examining the effect of input dimension on brain extraction

1045182

University of Oxford

Abstract. In this paper I examine the differing qualities of Machine Learning models for brain and substructure extraction, trained on varying dimensions of data and across the three anatomical planes, and see the effect of this choice on segmentation accuracy.
Code at https://tinyurl.com/DLHanon .

Keywords: MRI, Convolution, Machine Learning, Mini-Project

1 Introduction

Given the IXI Dataset, which consists of 3D MRI scans and volumetric labelling information, the natural question is "how effective are machine learning methods on the segmentation task?" We saw the basic UNet design in the course, which we trained to segment pixels in the brain in a 2D MRI from the rest of the image. This task is also affectionately known in the research community as "skull stripping", as our models learn to mask out pixels (or voxels) which aren't contained in the brain. This dataset is much larger, and more complex, than the toy dataset that we used in the practical. For starters, being composed of 3D tensors, the segmentation task is now extracting 3D volumes, rather than 2D as done previously. Secondly, as we are presented with 3D tensors, rather than, say, a series of 2D images, the dimensionality of the data that we feed to the model is now a hyperparameter of sorts - in order to recover a 3D segmentation mask, we can run the model on 2D slices individually and stack the outputs, or run a 3D convolution. Another choice in data preparation is how along which axis we slice the 3D tensors into 2D. The more complicated task, which we can run on the "subcortical" dataset, is more complex than brain extraction - rather than just stripping the skull, instead models are tasked with identifying specific regions of the brain. Previous attempts have been reasonably successful in the skull stripping task, but there is less research to be found on the subcortical one.

2 Related Works

The most important paper to cite here is of course [7], which introduces the UNet architecture off of which this entire work is based. The paper, unlike this work, is concerned solely with 2D image segmentation, specifically on cell images. The

objective in this case is also multi-class, whereas whole-brain segmentation maps only have labels in $\{0, 1\}$. The original paper is applicable to 3D segmentation, but we obviously need to adapt the method. Second to the original UNet paper is [2], which proposes SegNet. SegNet, like the models in this paper, is tasked with voxel-wise (3D) segmentation of brain scans, and does this using a hybrid approach, using a localised 3D filter and a broader 2D filter - the feature values after a layer of convolution are a function of that voxel, its immediate neighbours in 3D, and the 3 orthogonal 2D slices passing through that voxel. I won't implement SegNet, though it does align with the ideas presented in the SUNet later, of convolving over neighbours in both 2D and 3D space. I was unsurprised to read that someone before me had thought of doing 3D convolutions over the MRIs. [4] is an example of such a paper, which admittedly only runs the skull-stripping task, but achieves good results. They only use 125 training datapoints, consisting of 77 women and 48 men, who are all in the 21-45 age range. This paper and the original UNet paper differ in how many pooling layers are necessary - [7] notes that larger images require more pooling layers, and does 4 such operations on 572×572 images. If we look closely at the architecture in this paper, it only does two more pooling layers after the image has been transformed to 128×128. [4] only does 3 3D pooling operations. Following [4], and the intuition in [7], the models here will all contain 3 pooling layers, and 3 upsampling layers accordingly. [8] is much more related to how we choose the anatomical plane in our model. This paper talks about the observed (a)symmetry in the brain about the mid-sagittal plane, and how identifying this plane can improve downstream tasks, such as centring images for further analysis, as in [9], or scanning for malignancies, as in [1] which covers automatic segmentation of glioma, a type of brain tumour. This is done with axial slices, as, to paraphrase, "the fluid-filled space between the two hemispheres of the brain can be used to find a rough line of symmetry." This motivates the question of "which plane is best", in this case when taking a 2D or 2.5D approach to segmentation, but also in the field of MRI analysis generally. This paper, and its background area of research, would suggest that our model ought to look at images that have the sagittal plane "running down the middle" of them. We can encode this inductive bias by using models which read in axial or coronal slices, rather than sagittal.

3 Theoretical Considerations

3.1 2D processing

One way in which we can process the data is using 2D architectures - to turn the 3D tensors into 2D images, we simply slice the tensors in a given axis. Then, we can do extraction on each image slice using UNet, and then stack the slices together at the end. Loss is computed on a per-slice basis when training, but on whole brains during testing. We can either train a network on each slice dimension (say, 40 networks if we are slicing along the first axis in the data), or share weights between each slice. The former approach is extremely costly computationally, and also has practical drawbacks. Say, for example, that we

train a number of networks to segment the brain along each slice. If the 3D MRI passed to the model is not centered correctly, for example, if the subject leant to one side during the scan, then each network would be running on a slice different to those it was trained on - this would ruin the per-slice architecture trained on sagittal images. [6] in fact states that this (accidental leaning) is a common problem in MRI scans! A network trained on axial images, for example, could even be stumped by a patient who is particularly tall or short! I will illustrate the fragility of such a model by training a network on only one slice index, and then testing it on another. This reminds us of the idea of "overfitting" - a per-slice model will likely have better training accuracy, but will generalise poorly to real-world / unseen data.

Data in 2D As the network is weight-shared, and only deals with slices in a particular anatomical plane, we can sample data completely at random. For example, in the case of Guys' Hospital, in which the input tensor has shape $(321, 40, 128, 128)$, assuming we are taking slices in the sagittal plane (which corresponds to the second axis), we have $321 \times 40 = 12840$ training images, from which we can sample randomly. Each individual image, of size 128×128, is normalised to have pixel values s.t. $\mu = 0, \sigma = 1$ **per image**, or in PyTorch terms, we normalise along the two non-slice indices (in the sagittal case, these are the 3rd and 4th index). Overall, by an averaging property, the entire tensor will be normalised to $\mu = 0, \sigma = 1$, but this is unimportant as we aren't doing any operations on "the entire tensor at once".

3.2 2.5D

In the case of 2.5D, we pass in a series of 2D images, from which the model gets contextual information between images, but doesn't treat the data as a 3D tensor. Here I propose two new architectures: the RUNet, and the SUNet, which stand for **R**ecursive UNet and **S**andwich UNet respectively. I'll briefly outline the architectures, though they are both quite simple, and available in the code repo. I expect the 2.5D nets to work better than the 2D nets, for the simple reason that they consider context for each image. For example, the 2D scan networks have no way to align segments between images, which I imagine hurts accuracy somewhat. This is especially important for the subcortical structures, which are quite small - in this case, knowing how the structure extends on "either side" of the image should help the network.

RUNet & SUNet RUNet is a recurrent-type architecture, applied to the UNet. Informally, a forward pass of the network takes in a 3D tensor, but processes it as a sequence of 2D images. After a forward pass is done using a regular UNet, on a particular image, the output of the network is fed in as input along with the next image. My hope is that knowing how the previous image was segmented ought to improve segmentation of the next image. If a section of a subcortical structure, e.g. the amygdala, is found in a collection of pixels in an image, we

expect a similar label for these pixels in the adjacent images. My only worry is that this network is not truncated, and this can also harm training. To alleviate that, I also wrote the SUNet, which still receives sequences of images, but in a more local manner.

The SUNet, morally speaking, is just the 2D UNet, but with 3 input channels. Instead of feeding in the result of a previous segmentation as input to the UNet, we instead stack an image with its two neighbours as the input to the model. This way the convolution filters can "see" how structures in the image extend in the slice dimension, to better segment an image. However, we are only trying to segment the middle image - the other, sandwiching images are just to help the model, which will hopefully learn how to incorporate this information.

Learnable Tokens In both the SUNet and RUNet, we have the problem that edge slices don't have a "complete input". That is, there is no previous slice which we can feed into the model along with the slice we are trying to segment. Usually, in a forward pass of SUNet, we input to the model slices $(i-1, i, i+1)$ in order to do segmentation on slice i. Likewise in RUNet, on the forward pass for slice i, we feed in the result of the forward pass on slice $i-1$. To combat this, I tried two approaches. In the SUNet, I effectively pad the input with a copy of the slice. Concretely, when run on slices 0 and 39 (assuming we are working in the sagittal plane), we run the model on slices $(0, 0, 1)$ and $(38, 39, 39)$. In the case of RUNet, the model architecture contains a **learnable** token, of the same dimensionality as the input images, which the model can hopefully use when running the first forward pass.

Data in 2.5D As the data are still processed "per image", we still normalise the data in a per-image fashion, along the index from which we are taking slices. The manner in which we load the data is also model specific. For RUNet, we load in an entire tensor at once, and the model will iterate over slices. In the SUNet, we load triple slices into the model. This means that the RUNet has one datum per patient, while for SUNet there is the same multiplier as in 2D.

3.3 3D

The final, obvious way in which we can do this task is in 3D. In this case, we adapt the UNet architecture, but replace 2D operations with 3D operations - we go from 2D convolution filters and transposed convolutions to their 3D equivalents. In terms of practical applications, 3D convolutions are a much more "expensive" operation computationally, but for the relatively small data tensors, this doesn't seem to be a problem. There isn't any notion of "choosing a plane" in the 3D task, as the 3D convolutional filters move in all 3 dimensions when sliding over the image.

Data in 3D Preparing the 3D data is slightly different than the previous sections. As we are dealing with each 3D tensor "in one go", it makes sense to do

intensity normalisation over the entire tensor, rather than per slice. In numerical terms, we normalise over all indices except the first at once. There is a slight difference between this per-patient normalisation and the per-slice normalisation used in the previous sections, though it is largely imperceptible to the eye. The difference "from above" can be seen in 1.

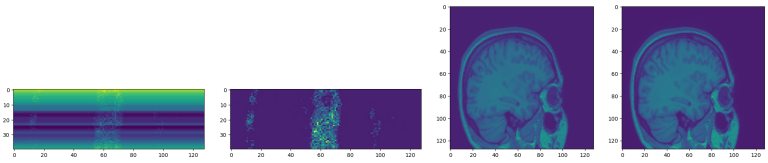

Fig. 1: Left: An axial slice, showing sagittal slice-wise normalisation (L) and tensor-wise normalisation (R). The differing colour at index 0 in the sagittal plane correspond to a values of -0.913 and -0.986 for the "background" colour of the two sagital slices. The contrast in these images has been made very high. Right: Images corresponding to sagittal slices 0 in the above figures. Here, the images (which have been normalised using the same boundaries in imshow) are indiscernable to the naked eye. I direct the confused reader to the MPL [3] docs.

Convolution vs FC FC results are uninterestingly bad on even the 2D task, and that isn't really the focus of this work. I do, however, include in the results, a comparison of generalisation power of FC vs UNet architecture (trained on one slice and tested on another).

Anatomical plane As mentioned in the related work section, I am particularly interested in how well models trained on different anatomical planes do. One reason why I initially thought one plane might be easier that the other is that the cerebral cortex is wrinkled, and this wrinkling may not be uniform across dimensions. It could be the case that in a sagittal image, the cortex looks extremely wrinkled, but each axial slice has a smooth boundary between the cortex and its surroundings (the meninges). Upon a closer inspection, slices in all three dimensions have wrinkly borders. However, the wrinkling is more complex in the sagittal images, due to the truncation of the MRIs.

Whole-brain vs Subcortical As mentioned previously, I will attempt all tasks on both the brain extraction task and the subcortical structure labelling task. However, I won't be training the network on all 5 labels in the subcortical dataset - converting the labels into a one-hot form requires us to initialise a tensor of shape $(321, 5, 40, 128, 128)$, which repeatedly crashed Google Colab, or ate up all available memory on my local machine. Instead, I will train on identifying segmenting individual structures - in the end I just went with the thalamus (label 1).

Data & Preparation I only use the GUYS dataset in this task, but I can justify this. For one, as mentioned in the related works, other papers working on a similar task, e.g. [4], have used the same, or a smaller number of patients and still found good results. Secondly, for the tasks in which we don't process each tensor as a whole, we effectively multiply the amount of training datapoints by either 40 or 128, to go from a number of 3D tensors to a number of 2D slices. Most compellingly, we can view convolutional architecture as a form of regularisation. The model is forced to share weights between convolutions on all pixels / voxels, which makes the model far less able to overfit to the training data. In terms of preparing the target dataset, I convert the labels to one-hot encodings, and have the models output vectors of the same shape.

4 Method

To test the effect of differing data-dimensionalities, I will train the 2D, 2.5D, and 3D models, and report their test loss & accuracy. Additionally, to test the effect of anatomical plane choice, I will train & run the 2D and 2.5D models on all 3 planes separately, and see whether there is much difference. I will also contrast the effectiveness of the 2D, 2.5D, and 3D architectures on both the skull-stripping, and subcortical tasks.

My hypotheses are as follows: the skull stripping task is easy enough that there won't be much difference between all modes of data preparation. However, the subcortical task requires interaction between slices, as the structures are small and local. In this case, I expect the 2.5D and 3D models to outperform the 2D models. I can't tell a priori which anatomical plane will do best on these tasks - I think that in the 2D case, sagittal will do a lot worse, as it won't be able to line up slices of small structures between images. I also expect the 3D architecture to struggle more on skull shaving than subcortical - the 3D filters need to be quite complex to model 3D wrinkling, while the subcortical volumes are quite uniform.

All models are trained using an Adam optimiser, with $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-9}$, as recommended in the original paper [5]. I use the soft version of Dice Loss, as seen in practicals as the loss function, though final accuracies are reported after an $argmax$ function is applied. Models were trained for 20 epochs, though the validation loss typically stabilised after ≈10 epochs in the 2D and 3D cases, ≈5 in the 2.5D.

Other hyperparameters include the MaxPool kernel size, which I set to 2 for both 2D and 3D models, the convolution kernel size, which I set to 3, and the batch size of 16. The convolution kernel of size 3 is typical in such architectures, as it is large enough to capture local information, but of a minimal size to reduce model complexity, and the maxpool of size 2 loses minimal information at each layer. I varied the batch size, and smaller values ($\leq$4) gave unstable training. I used 16 for all recorded results.

Table 1: Test Loss, Thalamus Extraction (above), and Test Loss & Accuracy, Whole-Brain Extraction (below). Accuracy left out due to label imbalance.

Model	Sagittal	Axial	Coronal
2D Unet	(**0.1123**)	(0.3942)	(0.4206)
SUNet	(**0.1167**)	(0.4084)	(0.4743)
RUNet	(**0.1542**)	(0.5838)	(0.5933)
3D Unet	(0.0489)		
2D Unet	(**0.0191**, 0.9831)	(0.0933, 0.9777)	(0.1630, 0.9714)
SUNet	(**0.0171**, 0.9848)	(0.1070, 0.9718)	(0.3118, 0.8554)
RUNet	(**0.0224**, 0.9816)	(0.4077, 0.6188)	(0.4296, 0.5943)
3D Unet	(0.0236, 0.9819)		

Table 2: Fragility Experiment (Axial). UNet generalised much better to unseen slices than FCNet. Validation loss on slice index 40, test stats on slice 20.

Model	Val Loss	Test Loss	Test Acc
FCNet	0.0021	0.4	0.4791
2D UNet	0.0175	0.1021	0.9006

RUNet training I tried a pre-training approach on the RUNet. Along with training the network from scratch on the input, I also trained a truncated model on a hybrid input, of an image, and the ground truth labels of the previous image. This way we don't get long term gradient updates, as we can train this network on individual slices. The network may also learn better how to attend over the previous outputs. We can then use this pre-trained network as part of the larger RUNet architecture, and learn the token in a second round of training.

5 Results

Which plane is best? Pretty resoundingly, the answer is sagittal! This runs in completely the opposite direction to my hypotheses. I was particularly surprised on the skull shaving, as the images look easier in the axial and sagittal planes, due to the truncation of the images(in the sides of the head).

Architectures In another surprise, the 2.5D architectures did worse than the 2D architectures in the subcortical task. One reason for this may be that the thalamus isn't long and thin (like the caudate, for example), so the 2.5D models don't gain much from tracking its position between layers. I am surprised, however, that the 2.5D architectures were considerably worse. Having looked

through the test results, I can say that the low dice score in the subcortical task by the SUNet and RUNet is due to a very high false positive rate. The performance of the 3D architectures is unsurprisingly good, and as suspected, showed the biggest improvements against the other architectures in the subcortical task.

RUNet Pre-training The RUNet trained from scratch actually did **better** (on Dice Loss) than the one using a pre-trained sublayer - on sagittal slices, the basic RUNet achieved average test loss = 0.0683, test accuracy = 0.9983, while the pre-trained one achieved average test loss = 0.0182, test accuracy = 0.9840. Due to this minimal difference, and the evidence in 2, which suggests that the model learned a more informative token from scratch, I didn't push this further.

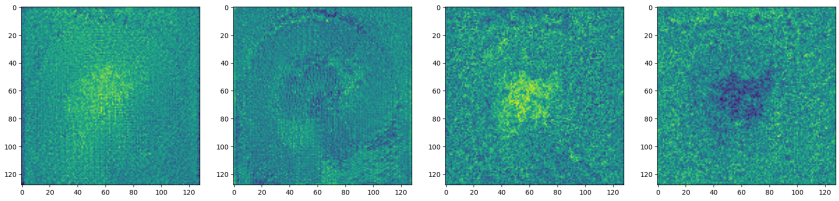

Fig. 2: The learned tokens from the RUNet, on sagittal whole-brain extraction. Left: learned using the original, non-truncated learning, trained "from scratch". Right: those derived in the second step of the pre-training procedure outlined earlier. There are two tokens in both cases, as the model **outputs** 2-channels.

6 Discussion / Conclusion

I think I've explored the question of data preparation pretty exhaustively, and with a reasonable amount of theoretical investigation. To explore further the effect of plane choice, it would be interesting to do this study on non-truncated images, where the model can't trivially learn to segment the straight edges of the non-sagittal images in the brain extraction task. Related works mention that finding the mid-sagittal plane is useful, but our models don't actually have an inductive bias that makes use of this. It would therefore be interesting to test models which explicitly use this technique, e.g. a model with some symmetry constraint. While I varied the form of **input** to the model, I didn't really explore the effect of changing the kernel size and shape. The models I tested are all rather straightforward, either using out-of-the-box 2D or 3D convolutions. Papers like [2] are a lot more imaginative, and it would be interesting to explore that further as well. I am somewhat disappointed that the pre-trained RUNet models didn't do better, both in terms of final accuracy, and in learning the token. The poor performance of all subcortical models suggests that it would be good to run these experiments over all subcortical labels at once, as their positions relative to one another may be more learnable.

References

1. Barzegar, Z., Jamzad, M.: Fully automated glioma tumour segmentation using anatomical symmetry plane detection in multimodal brain mri. IET Computer Vision **15**(7), 463–473 (2021). https://doi.org/https://doi.org/10.1049/cvi2.12035, https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cvi2.12035
2. de Brebisson, A., Montana, G.: Deep neural networks for anatomical brain segmentation (2015)
3. Hunter, J.D.: Matplotlib: A 2d graphics environment. Computing in Science & Engineering **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
4. Hwang, H., Rehman, H.Z.U., Lee, S.: 3d u-net for skull stripping in brain mri. Applied Sciences **9**(3) (2019). https://doi.org/10.3390/app9030569, https://www.mdpi.com/2076-3417/9/3/569
5. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)
6. Liu, Y., Collins, R., Rothfus, W.: Robust midsagittal plane extraction from normal and pathological 3-d neuroradiology images. IEEE Transactions on Medical Imaging **20**(3), 175–192 (2001). https://doi.org/10.1109/42.918469
7. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation (2015)
8. Tuzikov, A.V., Colliot, O., Bloch, I.: Evaluation of the symmetry plane in 3d mr brain images. Pattern Recognition Letters **24**(14), 2219–2233 (2003). https://doi.org/https://doi.org/10.1016/S0167-8655(03)00049-7, https://www.sciencedirect.com/science/article/pii/S0167865503000497
9. Zhang, R., Sato, T., Arisawa, H.: Symmetry recognition using mid-sagittal plane extraction and tilt correction in 3d head images. In: The SICE Annual Conference 2013. pp. 761–766 (2013)