3D Capsule Networks for Brain MRI Segmentation

Authors

Arman Avesta, MD,[1,2,3] Yongfeng Hui, BS, MPH,[2,3] Harlan Krumholz, MD, MS,[3,4] Sanjay Aneja, MD.[2,3,5]

[1] Department of Radiology and Biomedical Imaging, Yale School of Medicine, New Haven, CT 06510
[2] Department of Therapeutic Radiology, Yale School of Medicine, New Haven, CT 06510
[3] Center for Outcome Research and Evaluation, Yale School of Medicine, New Haven, CT 06510
[4] Division of Cardiovascular Medicine, Yale School of Medicine, New Haven, CT 06510
[5] Department of Statistics and Data Science, Yale University, New Haven, CT 06511

Abstract

INTRODUCTION: Segmenting the brain anatomy around a tumor on brain images is important for radiotherapy and surgical planning. Current auto-segmentation methods often fail to segment the brain anatomy when it is distorted by tumors.

OBJECTIVE: To develop and validate 3D capsule networks (CapsNets) that can segment brain structures with spatial features that were not represented in the training data.
Methods: We developed, trained, and tested 3D CapsNets using 3430 brain MRIs acquired in a multi-institutional study. We compared our CapsNets with U-Nets using a battery of performance measures, including accuracy in segmenting various brain structures, segmenting brain structures with spatial features not represented in the training data, performance when the models are trained using limited data, memory requirements, and computation times.

RESULTS: 3D CapsNets can segment third ventricle, thalamus, and hippocampus with Dice scores of 94%, 94%, and 91%, respectively. 3D CapsNets outperform 3D U-Nets in segmenting brain structures that were not represented in the training data, with Dice scores more than 30% higher. 3D CapsNets are also remarkably smaller models compared to 3D U-Nets, with 93% fewer trainable parameters. This led to faster convergence of 3D CapsNets during training, making them faster to train compared to U-Nets. The two models were equally fast during testing.

CONCLUSION: 3D CapsNets can segment brain structures with high accuracy, outperform U-Nets in segmenting brain structures with features that were not represented during training, and are remarkably more efficient compared to U-Nets, achieving similar results while their size is an order of magnitude smaller.

Introduction

Anatomical segmentation of diagnostic images is an important step in a variety of clinical workflows within radiology, radiation therapy, and surgery. Specifically, within radiation therapy of the brain, segmentation of critical organs at risk (such as hippocampus, brain stem, and optic nerves) is necessary to reduce toxicity in treatment planning. Within surgical planning, neuroanatomical segmentations aid in image-guided interventions. Manual segmentation is impractical because it requires radiologist-level expertise, is time-consuming, and is prone to inter- and intra-operator variability.[1] Current auto-segmentation methods are promising, but are limited when attempting to segment anatomy that is not well represented in the training data. With small, narrow training sets and with images that contain space-occupying lesions that distort the normal anatomy, these methods often underperform. <reference>

Auto-segmentation using capsule networks (CapsNets) represents a potential solution to this problem.[2,3] CapsNets are unique in deep learning architectures because, in addition to learning representative features of an image, they also encode spatial information (such as rotation, size, and shear) about the learned image features. If a structure rotates, changes in size, or undergoes other spatial changes, the capsule encoding that structure can still recognize it while encoding the changed spatial features.[2] CapsNets can achieve this level of knowledge generalization without data augmentation. CapsNets have been shown to outperform current segmentation methods in segmenting lungs on CT images, and muscle and fat tissues on MR images. However, CapsNets have not been well evaluated for segmenting neuroanatomic structures.

In this study, we developed and validated 3D capsule networks for volumetric segmentation of neuroanatomy on brain MRIs.[4] We trained and tested our model using a multi-institutional dataset of more than 3000 brain MRIs. We compared the performance of 3D CapsNets with 3D U-Nets across different neuroanatomic structures with varying levels of segmentation difficulty.

Methods

Dataset

The dataset used for this study included 3,430 T1-weighted brain MRI images, belonging to 841 patients from multiple institutions enrolled in the Alzheimer's Disease Neuroimaging Initiative (ADNI) study.[5] The participants in this study range from normal to mild cognitive impairment to Alzheimer's dementia. On average, each patient underwent four MRI acquisitions. Details of MRI acquisition parameters are provided in Appendix 5. We randomly split the patients into training (3,199 MRI volumes), validation (117 MRI volumes), and test (114 MRI volumes) sets.

Anatomic Segmentations

Three neuroanatomic structures were chosen for our analysis including third ventricle, thalamus, and hippocampus. These structures were chosen to represent neuroanatomic structures with varying degrees of segmentation difficulty. Segmentations for training and testing were obtained using FreeSurfer, which is a segmentation software with expert-level performance for non-distorted brain images (including in patients

with Alzheimer's dementia).[6] .[7–9] To ensure that segmentations were free from error, 120 randomly-selected MRIs from the training as well as all 114 MRIs in the test set were evaluated by a board-eligible radiologist for accuracy.

Image Pre-Processing

To make data loading faster, we converted the DICOMs of each brain MRI into a 3D NIfTi file.[10] MRI volumes were then corrected for intensity inhomogeneities, including B1-field variations.[7,11] Then, the skull, face, and neck tissues were removed, only leaving the brain.[12] The resultant 3D images were cropped around the extracted brain. To overcome memory limitations, we cropped 64×64×64-voxel boxes of the MRI volume that contained each segmentation target. We fixated the location of these boxes (in relation to the center of the cropped brain MRI) for all MR images.

3D CapsNet

We built on the 2D CapsNets introduced by LaLonde et al[4] to develop 3D CapsNets for volumetric segmentation. CapsNets are composed of three main ingredients: 1) capsules that each encode a structure together with the *pose* of that structure: the pose is an n-dimensional vector that learns to encode orientation, size, curvature, location, and other spatial information about the structure; 2) a supervised learning paradigm that learns the transforms between the poses of the parts (e.g. head and tail of hippocampus) and the pose of the whole (e.g. the entire hippocampus); and 3) a clustering paradigm that detects a whole if the poses of all parts (after getting transformed) vote for matching poses of the whole. Therefore, any CapsNet architecture requires procedures for: 1) creation of the first capsules from the input; 2) learning transforms between the poses of parts and wholes; and 3) clustering the votes of the parts to detect wholes.

Figure 1.A shows the architecture of our 3D CapsNet. The first layer, Conv1, performs 16 convolutions (5×5×5) on the input volume to generate 16 feature volumes, which are reshaped into 16D vectors at each voxel. The 16D vector at each voxel provides the first pose that can learn to encode spatial information at that voxel. The next layer, PrimaryCaps2, has two capsule channels that learn two 16D-to-16D convolutional transforms (5×5×5) from the poses of the previous layer to the poses of the next layer. Likewise, the next *convolutional* capsule layers (green layers in Figure 1.A) learn m-to-n-dimensional transforms between the poses of the previous layer and the poses of the next layer. The number of transforms at each layer matches the number of capsule channels (shown by stacks of capsules in Figure 1.A). Our CapsNet has downsampling and upsampling limbs. The downsampling limb learns *what* structure is present at each voxel, and the skip connections from downsampling to upsampling limbs preserve *where* each structure is on the image. Downsampling is done using 5×5×5 convolutional transforms with stride = 2. The poses in the deeper parts of the downsampling limb have more pose components (up to 64) to be able to encode more complex spatial information. Additionally, layers in the deeper parts of the model contain more capsule channels (up to 8) to be able to encode more structures at each voxel, since each voxel in these layers corresponds to multiple voxels in the input that can each represent a separate structure. Upsampling is done using 4×4×4 transposed convolutional transforms with stride = 2 (turquoise layers on Figure 1A). The final layer, FinalCaps13, contains one capsule channel that learns to activate capsules within the segmentation target and deactivate them outside the target. Activation of a capsule is determined

by the length of its pose vector, which is a number between 0 and 1. Further details about the activation of capsules are provided in the supplemental material.

To find clusters of the agreeing votes of the parts, we used the inner products between the poses of the parts and the aggregate pose of the whole.[3] We used Dice loss to train our models and to evaluate segmentation accuracy.[13] Further details about the clustering method, loss function calculation, and activation of each capsule are provided in the supplemental material.

3D U-Net

The 3D U-Net was used as a benchmark to compare the performance of the 3D CapsNets. The U-Net is considered among the highest-performing segmentation algorithms in diagnostic imaging. The U-Net has shown strong auto-segmentation accuracy across a variety of different image modalities and anatomic structures. Figure 1.B shows the architecture of our 3D U-Net. The input image undergoes 64 convolutions (3×3×3) to generate 64 feature maps. These volumes then undergo batch normalization and ReLU activation. Similar operations are carried out again, followed by downsampling using max-pooling (2×2×2). The downsampling and upsampling limbs each include four units. Upsampling is done using 2×2×2 transposed convolutions with stride = 2. The final layer carries out a 1×1×1 convolution to aggregate all 64 channels, followed by soft thresholding using the sigmoid function. The model learns to output a number close to 1 for each voxel inside the segmentation target, and a number close to 0 for each voxel outside the target.

Model Training

We used Dice loss for training our models. Adam optimizer was used with the following hyperparameters: $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$. Training was done using 50 epochs, each consisting of all 3,199 brain MRIs in the training set, and with the batch size of four. Because of the large epoch size, we split each epoch into mini-epochs that each comprised 30 batches (120 MRIs). After each mini-epoch during training, the Dice loss was computed for the validation set (117 MRIs).

We used dynamic paradigms for learning rate scheduling and for selecting the best models. The initial learning rate was set at 0.002. The validation set Dice loss was monitored after each mini-epoch, and if it did not decrease over 10 consecutive mini-epochs, the learning rate was decreased by half. The minimum learning rate was set at 0.0001. The model with the lowest Dice loss *over the validation set* was selected as the best model and was used for testing.

Model Performance

In all following experiments, we trained CapsNets as well as U-Nets as competition, and then compared the performance of the two models.

Segmentation accuracy was compared, using Dice scores, for the third ventricle, thalamus, and hippocampus. These structures were chosen to represent neuroanatomic structures with varying degrees of segmentation difficulty. Third ventricle is an easy structure because it is a cerebrospinal fluid (CSF) filled

cavity with clear boundaries. Thalamus is a medium-difficulty structure because it is abutted by CSF on one side and brain parenchyma on the other side. Hippocampus is a difficult structure because it has a complex shape and is abutted by multiple brain structures with indistinct borders

To evaluate the performance of CapsNets on out-of-distribution images not represented in training data, we trained segmentation algorithms using only data from the right thalamus and right hippocampus. We subsequently measured the segmentation accuracy of each algorithm on test sets which only included the images of left thalamus and left hippocampus. Notably, we did not use data augmentation during training.

To assess the performance of CapsNets when training data is limited, we trained our models on smaller subsets of our training data comprised of 600, 240, and 120 MRI volumes. These smaller training sets were randomly selected. We then evaluated these models, trained on smaller training sets, on the full test set.

To evaluate the computational efficiency of CapsNets, we computed the model size using the number of trainable parameters and the memory required to run the model (in megabytes). We also measured the computational times required for training our models, as well as the time needed by the trained models to segment a brain MRI.

For all experiments, the mean segmentation accuracies over the test set were compared between CapsNets and U-Nets using paired-samples t-tests. The mean Dice scores together with their 95% confidence intervals were also tabulated for the two models and the three brain structures that were segmented. While our main measure of segmentation accuracy was Dice score, we also tested our final models using additional measures of segmentation accuracy. Details about these additional measures and the performance of our models using these measures are provided in the supplemental material.

Implementation

Image processing was done using FreeSurfer and Python. PyTorch was used for model development and testing. The SciPy package was used for statistical analyses. Training and testing of the models were run on AWS p2xlarge instances (4 vCPUs, 61 GB RAM, 12 GB GPU). The codes used to train and test our models is publicly available at: github.com/Aneja-Lab-Yale/Aneja-Lab-Public-CapsNet.

Results

This study included 3430 brain MRIs belonging to 841 patients across multiple institutions. [14] Patient demographics are provided in Table 1.

The accuracy of 3D CapsNets in segmenting various brain structures is above 90% and is within 1.5% of to the accuracy of U-Nets. Figure 1 shows the segmentation of various brain structures by both models in a patient. Table 2 compares the segmentation accuracy of the two models, measured by Dice scores. Supplemental Table 2 compares additional measures of segmentation accuracy between the two models.

The 3D CapsNets achieved better out-of-distribution segmentation accuracy compared to 3D U-Nets. When both models were trained to segment right-sided brain structures and tested on contralateral left left-sided brain structures, 3D CapsNets significantly outperformed 3D U-Nets with Dice scores more than 30% higher. Figure 3 illustrates segmentation of the contralateral left thalamus and hippocampus by both models in a patient. Table 2 compares out-of-distribution segmentation accuracy between the two models.

The 3D CapsNets and 3D U-Nets achieved comparable segmentation accuracy when trained on smaller datasets. When the size of the training set was decreased from 3199 to 600 brain MRIs, both CapsNet and U-Net were minimally affected. Further decrease in the size of the training set down to 120 brain MRIs caused a decrease in the accuracy of both models down to 85%. Figure 4 shows the performance of both models when trained on smaller datasets.

Our 3D CapsNets are remarkably smaller models compared to 3D U-Nets. The 3D CapsNet has 7.4 million trainable parameters, while the corresponding 3D U-Net has 90.3 million trainable parameters. In addition, the 3D CapsNet has fewer layers and fewer steps of image propagation in forward and backward passes, leading to a smaller cumulative size of the feature volumes in the entire model. The 3D CapsNet and 3D U-Net respectively hold 228 and 1364 megabytes of cumulative feature volumes in the entire model. Figure 5.A compares the size of 3D CapsNet and 3D U-Net models.

The 3D CapsNets train slightly faster compared to U-Nets. When we compared the training time between the two models (on an AWS p2xlarge instance with 12GB of GPU memory), our 3D CapsNets and 3D U-Nets respectively took about 1.5 and 2 seconds per example per epoch to train. The two models are equally fast during testing, taking 0.9 seconds to segment the MRI volume. Figure 5.B compares the training and testing times between the two models.

Discussion

In this study, we built on the previous work by Lalonde et al[4] to develop and validate 3D capsule networks for volumetric segmentation of brain MRIs. We also trained and tested 3D U-Nets as the main competitor. Our results showed that 3D CapsNets have high segmentation accuracy for segmenting various brain structures with Dice scores above 90%. While our CapsNets are one order of magnitude smaller than U-Nets, their segmentation accuracy is within 1.5% of U-Nets. In out-of-distribution segmentation, our CapsNets outperformed U-Nets with Dice scores more than 30% higher.

Our results corroborate previous studies that deep learning is effective in medical image segmentation.[15–19] This study also replicated the results of prior studies showing that U-Nets can segment brain images with high accuracy.[17,19] Our results also corroborate a previous study that showed the effectiveness of 2D CapsNets for segmenting biomedical images, outperforming other deep learning models including U-Nets.[4] A subsequent study showed that 2D CapsNets were less effective in segmenting heart and brain MRI slices.[24] As a result, 2.5D CapsNets were introduced that showed slightly improved performance in segmenting heart and brain MRIs, but with suboptimal segmentation accuracy.[24] Therefore, there was a need to develop 3D CapsNets for volumetric segmentation. This study achieved this goal by developing and validating 3D CapsNets for brain MRI segmentation.

CapsNets may represent better out-of-distribution modeling. Previous studies have already shown the generalizability of CapsNets to novel spatial features of inputs. In 2D object recognition, 2D CapsNets are outperformed other deep learning methods when the objects were imaged from viewpoints that were not represented during training.[2] In 2D image segmentation, 2D CapsNets were shown to outperform 2D U-Nets in segmenting rotated images.[4] Our study extends the literature by showing that 3D CapsNets outperform 3D U-Nets in segmenting mirror-image contralateral brain structures that were not represented during training. Notably, we did not use data augmentation during training. Therefore, this study provides further evidence that CapsNets have out-of-distribution segmentation capabilities. This study also corroborates previous results showing that CapsNet can model spatial features more efficiently, achieving higher or similar performance compared to other deep learning methods while having a smaller model size.[2–4] Our CapsNet is one order of magnitude smaller than U-Net while achieving similar segmentation results.

Our results corroborate with previous studies that show faster convergence of CapsNets during training, as compared to other deep learning methods.[2,4] Our results show that 3D CapsNets are slightly faster to train compared to 3D U-Nets. While clustering of pose vectors between capsule layers slows down CapsNets, they converge faster because they have 93% fewer parameters to train. The net effect of these opposing factors leads to slightly faster training of CapsNets. Our results also show that the two models are equally fast during testing. Given that the forward pass through the fixed, trained parameters during testing is faster compared to the forward *and* backward passes during training, the larger size of the 3D U-Net does not slow it down as much during testing as it does during training. At the same time, clustering between the capsule layers slows down the 3D CapsNet during testing to the same degree as during training. As a result, the two models end up being equally fast during testing.

To develop 3D CapsNets and make them work for volumetric brain MRI segmentation, we explored multiple design options, hyperparameters, loss functions, and implementation details to find optimal solutions. We used the validation set to explore these questions, and tested our final model on the test set only once. While our model performs well for volumetric segmentation of T1-weighted brain MRIs, we did not evaluate its performance for segmentation of other organs or other imaging modalities. We assume that our model would need modifications to perform well on other segmentation tasks or on brain MRIs that are pre-processed differently. We have described the experiments that helped us find optimal solutions for our design questions in the supplemental material, and we welcome further research to generalize our 3D CapsNet to the segmentation of other organs and to other imaging modalities.

Finally, some limitations of this study should be noted. First, the ground truth segmentations were elicited from FreeSurfer software package, since FreeSurfer is shown to have segmentation accuracy similar to human experts.[6,8,20,21] To ensure the accuracy of our segmentations, a radiologist checked and approved the segmentations of all MRIs in the test set as well as 120 randomly-selected MRIs in the training set. Second, we only validated our model for the segmentation of three brain structures of varying difficulty. Our model may not generalize to other organs or imaging modalities. Third, we only validated our model for segmenting brain MRIs without space-occupying lesions. Our model may not generalize to segmenting brain anatomy in the presence of lesions that distort the anatomy. However, this study showed out-of-

distribution segmentation capabilities of CapsNet, providing evidence that CapsNet may be the potential solution to the problem of neuroanatomic segmentation in the presence of space-occupying lesions.

Conclusion

In this study, we developed and validated 3D capsule networks for volumetric segmentation of brain MR images. While our capsule network is one order of magnitude smaller than the equivalent U-Net, it achieves comparable performance in segmenting brain structures of varying difficulty. Additionally, our capsule network showed out-of-distribution segmentation capabilities, making it a potential solution for neuroanatomical segmentation when the brain anatomy is distorted by space-occupying lesions.

References

1. Despotović I, Goossens B, Philips W. MRI Segmentation of the Human Brain: Challenges, Methods, and Applications. *Comput Math Methods Med* 2015;2015:e450341.

2. Hinton GE, Sabour S, Frosst N. Matrix capsules with EM routing. In: *International Conference on Learning Representations 2018*.

3. Sabour S, Frosst N, Hinton GE. Dynamic routing between capsules. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc.; 2017:3859–69.

4. LaLonde R, Xu Z, Irmakci I, et al. Capsules for biomedical image segmentation. *Med Image Anal* 2021;68:101889.

5. Crawford KL, Neu SC, Toga AW. The Image and Data Archive at the Laboratory of Neuro Imaging. *NeuroImage* 2016;124:1080–3.

6. Clerx L, Gronenschild EHBM, Echavarri C, et al. Can FreeSurfer Compete with Manual Volumetric Measurements in Alzheimer's Disease? *Curr Alzheimer Res* 2015;12:358–67.

7. Fischl B, Salat DH, Busa E, et al. Whole brain segmentation: automated labeling of neuroanatomical structures in the human brain. *Neuron* 2002;33:341–55.

8. Fischl B. FreeSurfer. *NeuroImage* 2012;62:774–81.

9. Fischl B, van der Kouwe A, Destrieux C, et al. Automatically parcellating the human cerebral cortex. *Cereb Cortex N Y N 1991* 2004;14:11–22.

10. Li X, Morgan PS, Ashburner J, et al. The first step for neuroimaging data analysis: DICOM to NIfTI conversion. *J Neurosci Methods* 2016;264:47–56.

11. Ganzetti M, Wenderoth N, Mantini D. Quantitative Evaluation of Intensity Inhomogeneity Correction Methods for Structural MR Brain Images. *Neuroinformatics* 2016;14:5–21.

12. Somasundaram K, Kalaiselvi T. Automatic brain extraction methods for T1 magnetic resonance images using region labeling and morphological operations. *Comput Biol Med* 2011;41:716–25.

13. Taha AA, Hanbury A. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC Med Imaging* 2015;15:29.

14. Weiner MW, Veitch DP, Aisen PS, et al. The Alzheimer's Disease Neuroimaging Initiative 3: Continued innovation for clinical trial improvement. *Alzheimers Dement J Alzheimers Assoc* 2017;13:561–71.

15. Rauschecker AM, Gleason TJ, Nedelec P, et al. Interinstitutional Portability of a Deep Learning Brain MRI Lesion Segmentation Algorithm. *Radiol Artif Intell* 2022;4:e200152.

16. Rudie JD, Weiss DA, Colby JB, et al. Three-dimensional U-Net Convolutional Neural Network for Detection and Segmentation of Intracranial Metastases. *Radiol Artif Intell* 2021;3:e200204.

17. Rudie JD, Weiss DA, Saluja R, et al. Multi-Disease Segmentation of Gliomas and White Matter Hyperintensities in the BraTS Data Using a 3D Convolutional Neural Network. *Front Comput Neurosci* 2019;13.

18. Duong MT, Rudie JD, Wang J, et al. Convolutional Neural Network for Automated FLAIR Lesion Segmentation on Clinical Brain MR Imaging. *Am J Neuroradiol* https://doi.org/10.3174/ajnr.A6138.

19. Weiss DA, Saluja R, Xie L, et al. Automated multiclass tissue segmentation of clinical brain MRIs with lesions. *NeuroImage Clin* 2021;31:102769.

20. Ochs AL, Ross DE, Zannoni MD, et al. Comparison of Automated Brain Volume Measures obtained with NeuroQuant and FreeSurfer. *J Neuroimaging Off J Am Soc Neuroimaging* 2015;25:721–7.

21. Yaakub SN, Heckemann RA, Keller SS, et al. On brain atlas choice and automatic segmentation methods: a comparison of MAPER & FreeSurfer using three atlas databases. *Sci Rep* 2020;10:2837.

Figure 1: CapsNet (A) and U-Net (B) architectures. Both models process 3D volumes in all layers, with dimensions shown on the left side. *D*, *H*, and *W* respectively represent the depth, height, and width of the image in each layer. In (A), the number over the Conv1 layer represents the number of channels. The numbers over the capsule layers (ConvCaps, DeconvCaps, and FinalCaps) represent the number of pose components. The stacked layers represent capsule channels. In (B), the numbers over each layer represent the number of channels. In the 3D U-Net, the convolutions have stride=1 and the transposed convolutions have stride = 2. Please note that the numbers over capsule layers show the number of pose components, while the numbers over non-capsule layers show the number of channels.

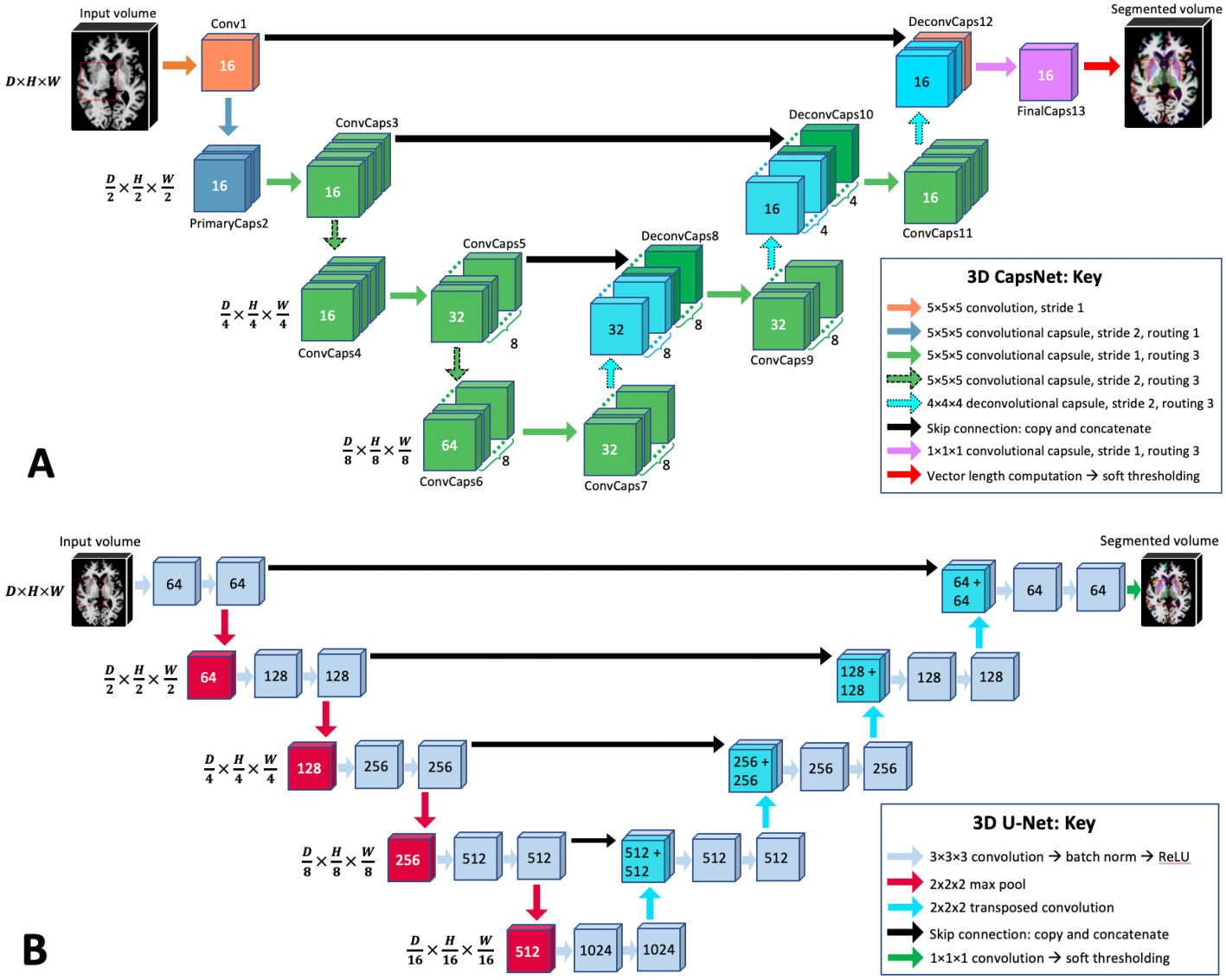


Table 1: Study participants. These accuracies were computed on the test set (114 brain MRIs). The 3rd ventricle, thalamus, and hippocampus respectively represent easy, medium, and hard structures to segment.

Data Partitions	Number of MRI volumes	Number of patients	Age mean ± SD	Gender % female	Ethnicity
Training set	3199	841	? ± ?	?%	?
Validation set	117	30	? ± ?	?%	?
Test set	114	30	? ± ?	?%	?

Figure 2: CapsNet vs U-Net in segmenting brain structures that were represented in the training data. Segmentations for three structures are shown: 3rd ventricle, thalamus, and hippocampus. Target segmentations and model predictions are respectively shown in white and red. Dice scores are provided for the entire volume of the segmented structure *in this case* (this case was randomly chosen from the test set).

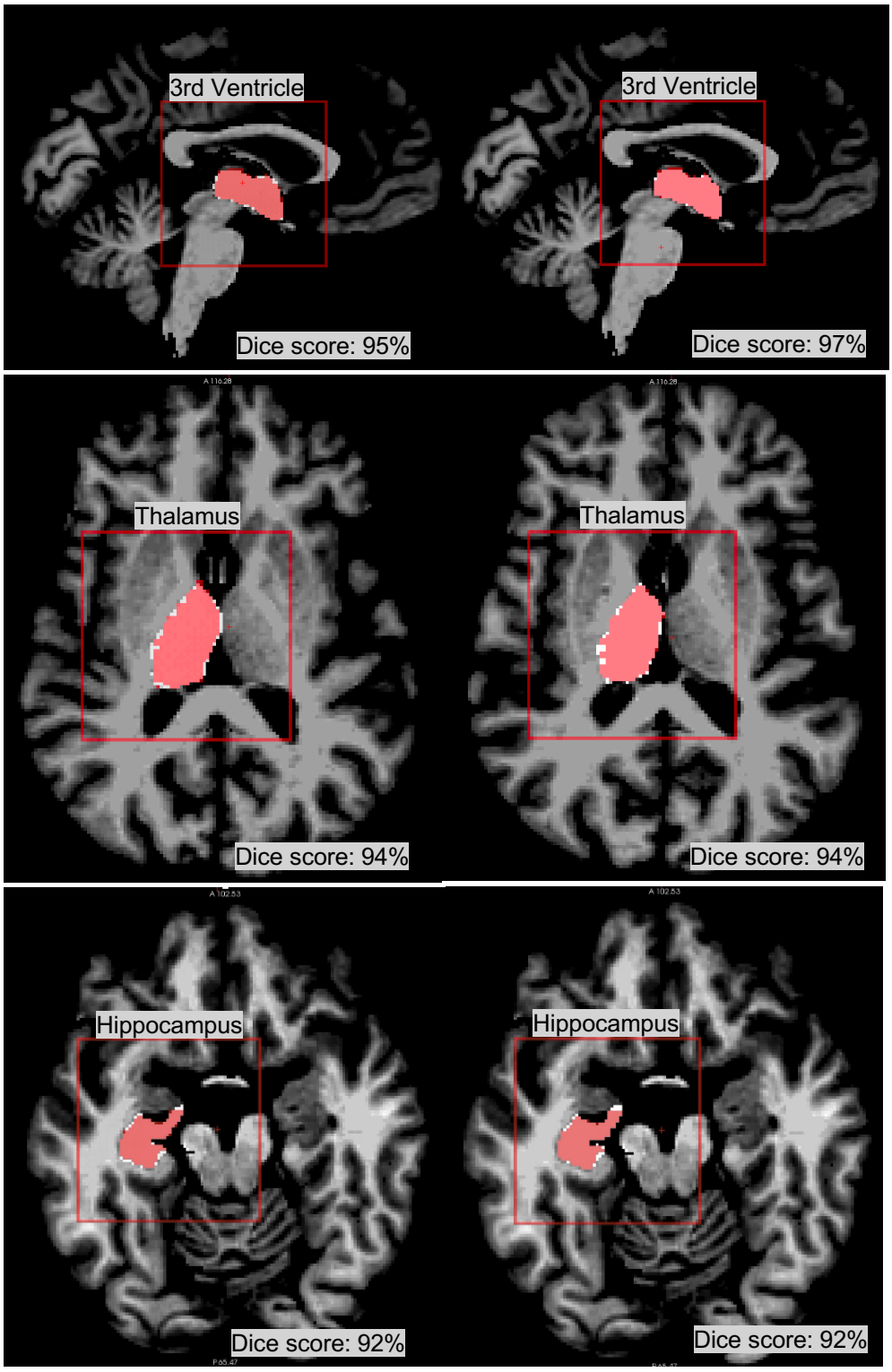


Table 2: CapsNet vs U-Net in segmenting brain structures that were represented in the training data. The segmentation accuracy was quantified using Dice scores on the test (114 brain MRIs). The 3rd ventricle, thalamus, and hippocampus respectively represent easy, medium, and difficult structures to segment.

Brain structure	CapsNet Dice score (95% CI)	U-Net Dice score (95% CI)	P-value†
3rd ventricle	93.6 (93.2 to 94.0) %	95.3 (95.0 to 95.6) %	< 0.01
Thalamus	93.6 (93.4 to 93.8) %	94.4 (94.3 to 94.6) %	< 0.01
Hippocampus	91.0 (90.7 to 91.3) %	92.5 (92.1 to 92.9) %	< 0.01

† Paired-samples t-test, degrees of freedom = 114 - 1 = 113

Figure 3: CapsNet outperforms U-Net in out-of-distribution segmentation. Both models were trained to segment right-sided brain structures, and were tested to segment contralateral left-sided brain structures. Target segmentations and model predictions are respectively shown in white and red. Dice scores are provided for the entire volume of the segmented structure *in this case*. While CapsNet partially segmented the contralateral thalamus and hippocampus, U-Net poorly segmented thalamus and entirely missed the hippocampus.

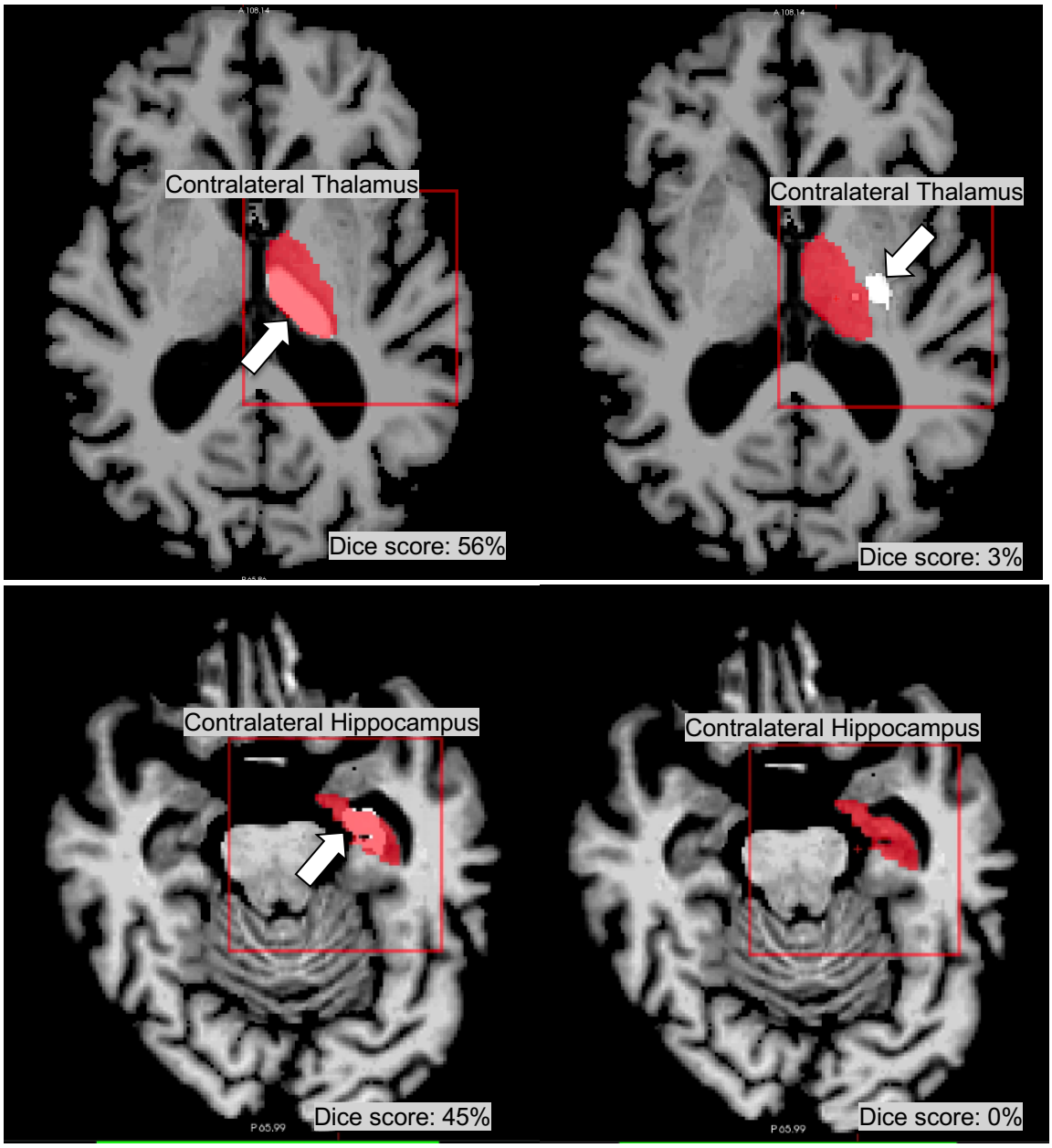


Table 3: CapsNet vs U-Net out-of-distribution segmentation accuracy. Both models were trained to segment the right thalamus and hippocampus. Then, they were tested on segmenting the contralateral left thalamus and hippocampus.

Brain structure	CapsNet Dice score (95% CI)	U-Net Dice score (95% CI)	P-value†
Thalamus	52 (46 to 58) %	16 (11 to 21) %	< 0.01
Hippocampus	43 (38 to 48) %	10 (6 to 14) %	< 0.01

† Paired-samples t-test, degrees of freedom = 114 - 1 = 113

Figure 4: CapsNet vs U-Net segmentation accuracy as a measure of training set size. When the size of the training set was decreased from 3199 to 600 brain MRIs, both models maintained their segmentation accuracy above 90%. Further decrease in the size of the training set down to 120 MRIs led to worsening of their segmentation accuracy down to 85% (measured by Dice scores).

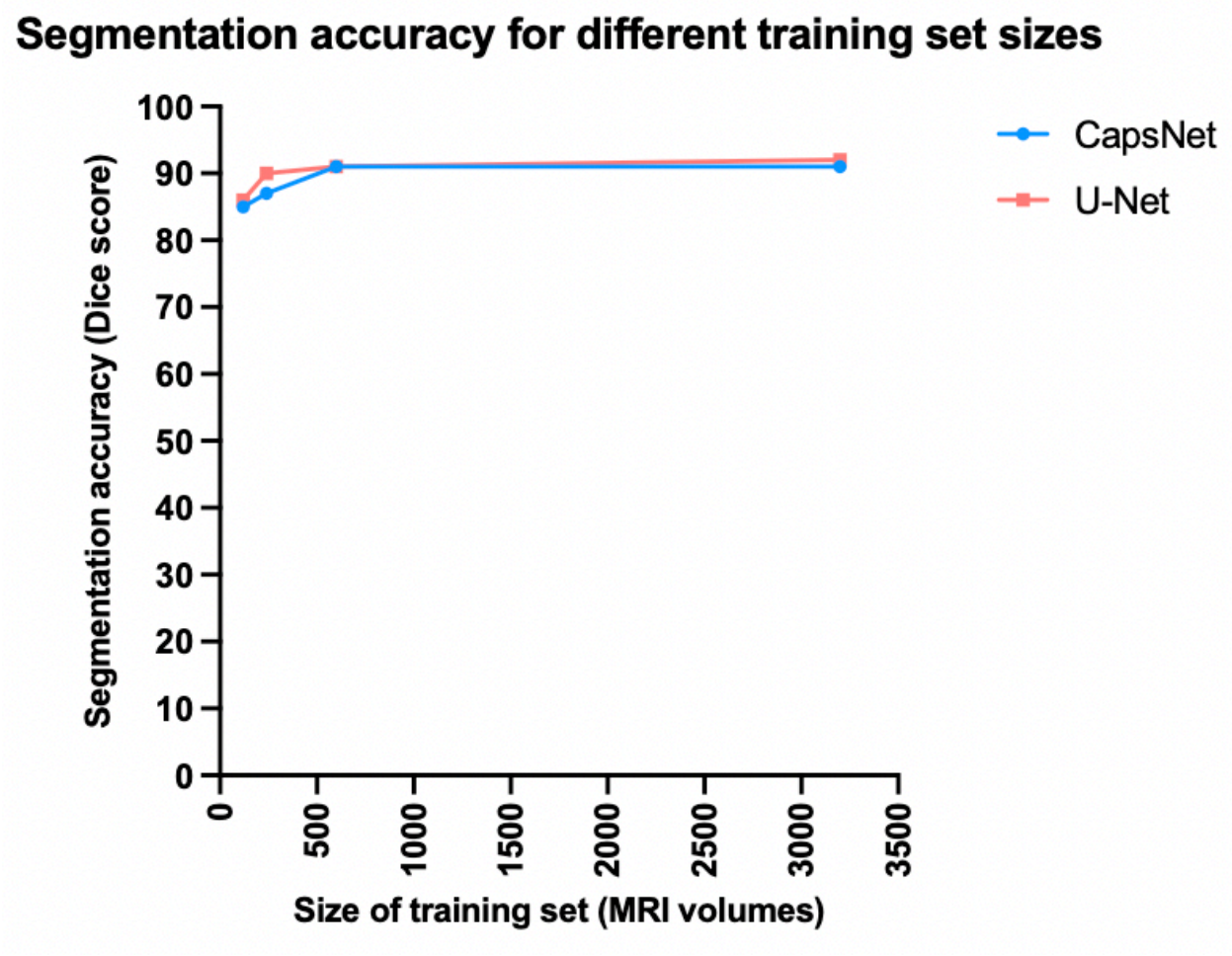
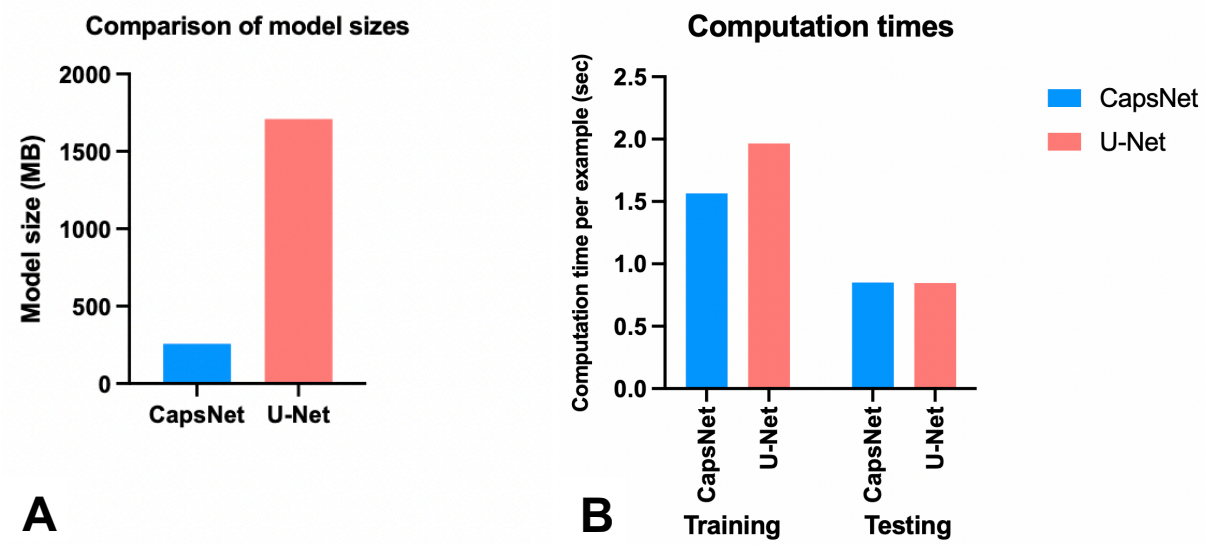


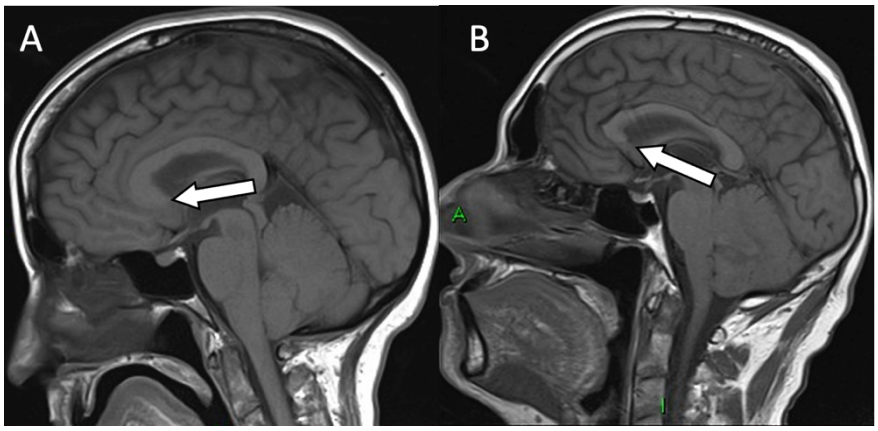
Figure 5: Model size (A) and computation times (B) compared between CapsNet and U-Net. The model size bars in (A) represent parameter size (28 and 345 MB for CapsNet and U-Net, respectively) plus the cumulative size of the forward and backward pass feature volumes (228 and 1364 for CapsNet and U-Net, respectively). The CapsNets train train slightly faster (B), given that they have 93% fewer trainable parameters. However, clustering between the capsule layers slows down CapsNets, making them only slightly faster than U-Nets during training. The two models are equally fast during testing.



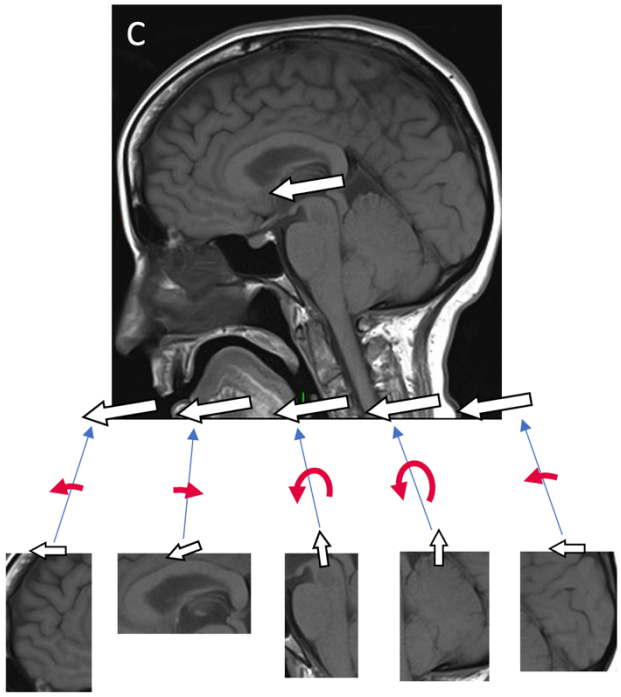
3D Capsule Networks for Brain MRI Segmentation

SUPPLEMENTAL MATERIAL

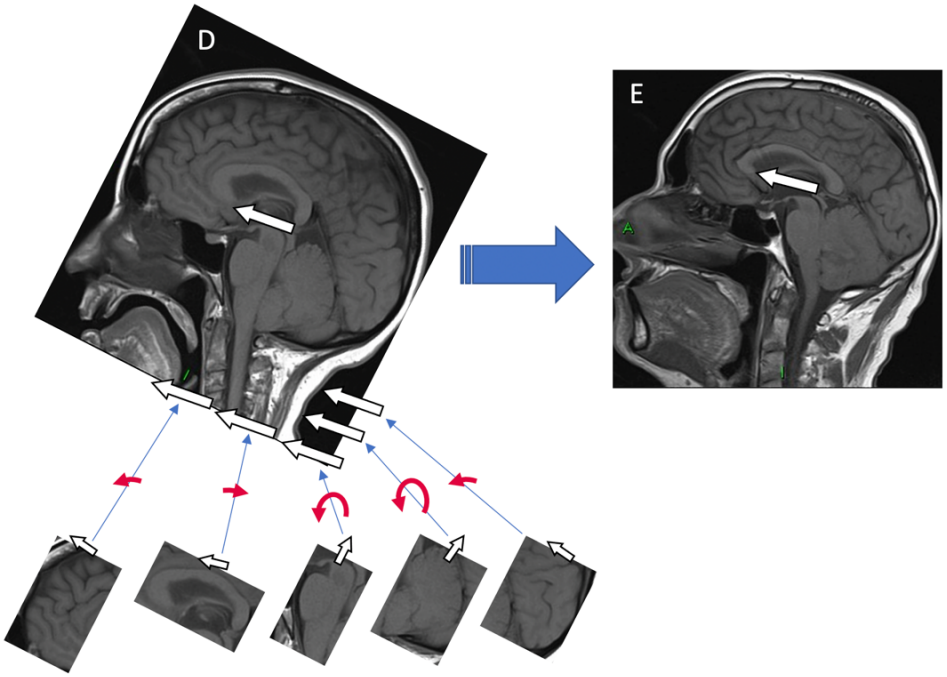
Appendix 1: Capsule Networks



The idea behind capsule networks (CapsNets) is to be able to detect an object even if its spatial features change. (A) shows the sagittal T1-weighted brain MRI of a patient with a forward head tilt, and (B) shows the MRI of another patient with a backward head tilt. White arrows (connecting the posterior commissure to the anterior commissure) demonstrate the orientation of the brain. Let's assume that we have a CapsNet that is trained to segment the entire brain. Let's additionally assume that the training set only contains patients with forward head tilt (like in A). An ideal CapsNet should generalize to segment the brain in patients with a backward tilt (like in B). To achieve this goal, CapsNets encode the spatial features of each structure that they detect. Here, the spatial features of each brain are encoded in a *pose* vector, which also contains the orientation of the brain (white arrow). Here, we want to illustrate how CapsNets detect a whole (the brain) when parts (frontal pole, corpus callosum, brainstem, cerebellum, occipital pole, etc.) all vote for the same spatial features of the whole.

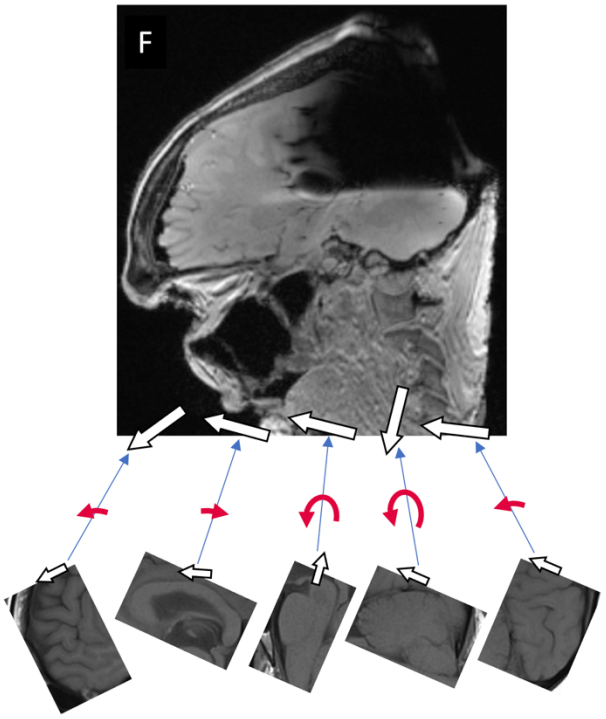


CapsNets are composed of three main ingredients: 1) capsules that each encode a structure together with the *pose* of that structure: the pose is an n-dimensional vector that learns to encode orientation, size, curvature, location, and other spatial information about the structure; 2) a supervised learning paradigm that learns the transforms between the poses of the parts (e.g. corpus callosum, brainstem) and the pose of the whole (e.g. the entire brain); and 3) a clustering paradigm that detects a whole if the poses of all parts (after getting transformed) vote for matching poses of the whole. Therefore, any CapsNet architecture requires procedures for: 1) creation of the first capsules from the input; 2) learning transforms between the poses of parts and wholes; and 3) clustering the votes of the parts to detect wholes.



(C) shows a CapsNet that has already detected parts of the brain and has encoded their spatial features (demonstrated by the smaller white arrow over each part). The red curved arrows demonstrate the transforms between the poses of the part and the pose of the whole. After transformation, each part votes for a candidate pose of the whole. If all these votes match, the

whole is present. Please note that we are only showing the orientations here for simplicity, but the pose vectors encode more complex spatial features.



In (E), We want the CapsNet to detect the backward-tilted brain while the model is only trained on forward-tilted brain images (such as in C). We can imagine that (E) is just the rotated version of (C), as demonstrated in (D). The parts are all rotated clockwise (compared to the poses of the parts in C). However, the same transforms (red curved arrows) can still transform the poses of the parts into the candidate poses of the whole. The candidate poses of the whole still match, and therefore the whole is detected. This process does not need any data augmentation: an ideal CapsNet can detect objects when they are rotated or have undergone other spatial changes, without the need for any data augmentation. This is because the transforms between the parts and the wholes (red curved arrows) will not change under these circumstances. Therefore, a change in the poses of the parts will cause an equivalent change in the pose of the whole. This is a powerful capability that makes CapsNets *equivariant* to the changes in the inputs: spatial change in the inputs will cause an equivalent spatial change in the pose of the detected objects. Such CapsNets can still detect the changed objects and will encode these changes in the pose of the detected objects. As a result, a CapsNet that is trained on forward-tilted braines (in C) can detect backward-tilted brains (in E) without the need for any data augmentation.

This approach is fundamentally different from other machine learning methods such as U-Nets, which do not have equivariance capabilities. Instead, the max-pooling layers in U-Nets try to kill information about the changes in the inputs to make the model *invariant* to the changed inputs. In essence, CapsNets use equivariance to *encode and model the spatial changes* in the inputs, making CapsNets more efficient in handling different variations of the same object. On the other hand, U-Nets use information killing (in max-pooling layers) to make the model *invariant to the spatial changes* in the inputs. Therefore, U-Nets cannot efficiently detect different variations of the same object.

(F) demonstrates why CapsNets are less susceptible to adversarial attacks compared to U-Nets. Here, this adversarial image contains all parts of the brain but with orientations that do not make sense, not making a whole. When the poses of the parts are transformed into the candidate poses of the whole (using the same transforms as in C), the candidate poses of the whole do not match. Therefore, the CapsNet would not detect a brain because of the mismatch between the candidate poses of the brain. On the other hand, a U-Net that is trained using augmented data would detect the parts. Such a U-Net has no mechanism to encode the orientation and other spatial features of each part. In the U-Net feature space, each part is either present or absent. Since all parts are present on this adversarial image, the U-Net can be fooled to detect the entire brain.

We can indeed use data augmentation to train U-Nets to detect objects with changed spatial features. We can also use adversarial training to prevent U-Nets from detecting adversarial images. But these inefficiencies lead to the need for a larger U-Net model. On the other hand, CapsNets handle the changed spatial features in a smarter way. This allows CapsNets, with one order of magnitude smaller size compared to U-Nets, to achieve similar results.

Appendix 2: Findings Agreeing Pose Vectors

Let's assume the previous capsule layers has six capsule channels, each outputting the vote vector of a part (v_1 to v_6). To find the vote vectors that agree, we first compute the vector summation of all vote vectors (v):

$$v = \sum_i v_i$$

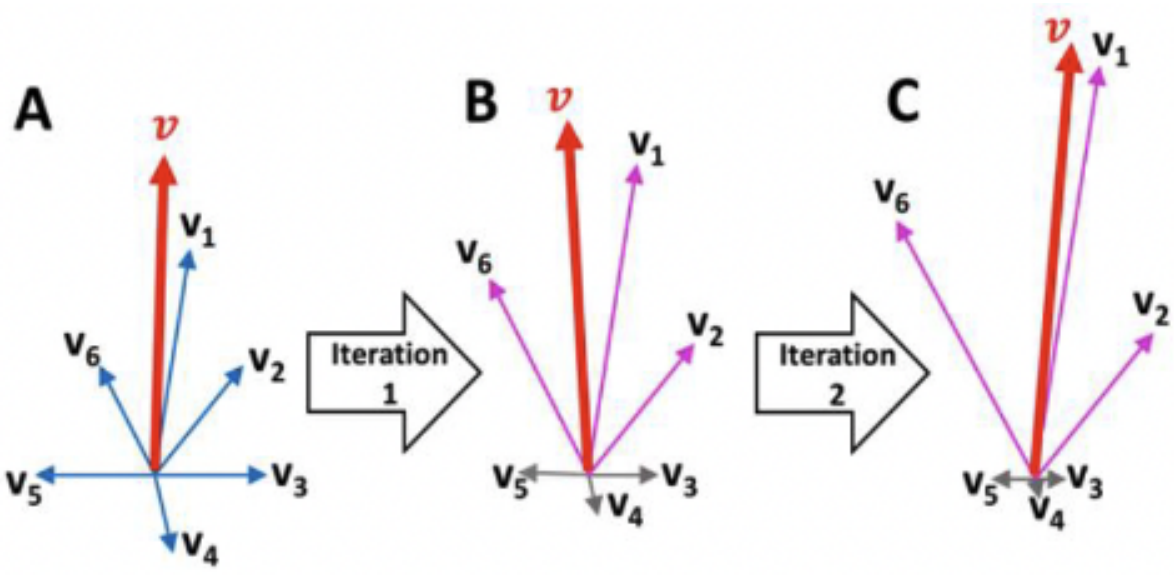
Then, we compute the inner products between each vote vector v_i and the sum v, yielding weights for each vote vector w_i:

$$w_i = v_i . v$$

Please note that w_i are scalar. Next, we re-compute the vector sum v using the weighted average of the vote vectors using weights w_i computed in the previous step :

$$v = \sum_i w_i \, v_i$$

This process is often repeated for three iterations. The number of iterations is a hyperparameter that should be set between capsule layers. This whole process increases the weights of the vectors that align with the sum (v_1, v_2, and v_6 in this example) and decreases the weights of the vectors that do not align with the sum (v_3, v_4, and v_5 in this example).



Appendix 3: Converting Final Layer Poses into Segmentations

The final layer of the 3D CapsNet is composed of one capsule channel that learns to activate capsules within the segmentation target and deactivate them outside the target. Activation of a capsule is determined by the length of its pose vector, which is a number between 0 and 1. The ground truth segmentations are coded similarly: pixels outside and inside the segmentation target are respectively coded by 0 and 1.

During testing, the length of the final layer's pose vectors is thresholded at T:

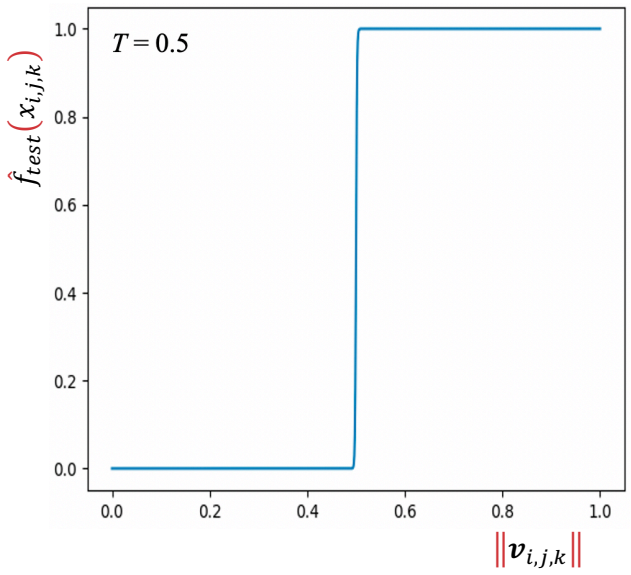
$$\hat{f}_{test}(x_{i,j,k}) = \begin{cases} 0, & \|\boldsymbol{v}_{i,j,k}\| < T \\ 1, & \|\boldsymbol{v}_{i,j,k}\| \geq T \end{cases} \quad (1)$$

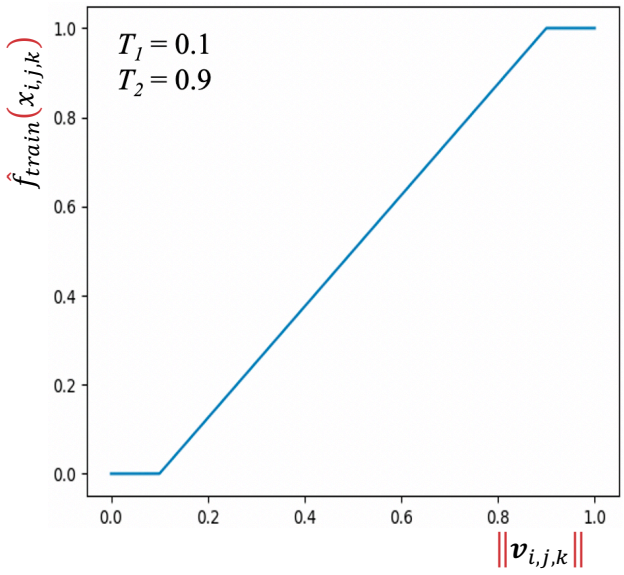
where $\hat{f}(x_{i,j,k})$ is the prediction of the CapsNet for the input voxel $x_{i,j,k}$ and $\|\boldsymbol{v}_{i,j,k}\|$ is the length of the final layer's pose vector $\boldsymbol{v}_{i,j,k}$ at the location (i,j,k) of the MRI volume (please note that $\boldsymbol{v}_{i,j,k}$ is itself a function of $x_{i,j,k}$, the function being the entire CapsNet that takes $x_{i,j,k}$ is the input and gives $\boldsymbol{v}_{i,j,k}$ as the output).

During training, the length of the final layer's pose vectors undergoes a piecewise transform as follows:

$$\hat{f}_{train}(x_{i,j,k}) = \begin{cases} 0 & , \quad \|\boldsymbol{v}_{i,j,k}\| < T_1 \\ \dfrac{\|\boldsymbol{v}_{i,j,k}\| - T_1}{T_2 - T_1} & , \quad T_1 \leq \|\boldsymbol{v}_{i,j,k}\| < T_2 \\ 1 & , \quad \|\boldsymbol{v}_{i,j,k}\| \geq T2 \end{cases} \quad (2)$$

If we set $T = 0.5$ $T_1 = 0.1$ and $T_2 = 0.9$, we get the following diagrams for $\hat{f}_{test}(x_{i,j,k})$ and $\hat{f}_{train}(x_{i,j,k})$ as functions of $\|\boldsymbol{v}_{i,j,k}\|$:





During training, the piecewise conversion (formula 2) enables a *forgiving paradigm* for the length of the final layer's pose vectors: if the length of the vector is more than 0.9 for a voxel inside the segmentation target, the loss for that voxel would be zero. Intuitively, a pose vector with a length more than 0.9 for a voxel inside the segmentation target is considered "good enough", so the training algorithm does not try to perfect the length of this vector to 1. Similarly, a pose vector with a length less than 0.1 is considered good enough for a voxel outside the segmentation target, so the training algorithm does not try to perfect the length of this vector to 0. This forgiving training paradigm makes the training of CapsNet stable because this

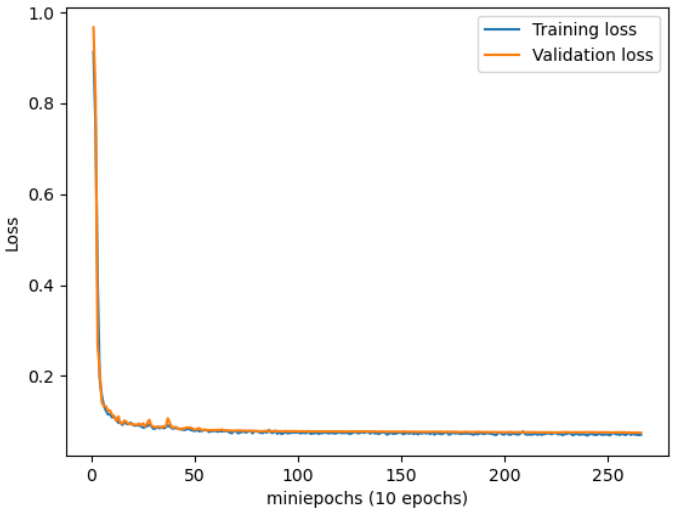
paradigm does not try to perfect the length of the pose vectors of the final layer to 0's and 1's. When a training paradigm tries to perfect the length of the pose vectors to 0's and 1's, that training paradigm becomes unstable because the pose vectors can assume values close to 0 and 1, but not exactly 0 or 1. Remember that the pose vectors are generated by the *squash function*, which cannot generate vectors with a length equal to 0 or 1:

$$\boldsymbol{v}_{i,j,k} = squash(\boldsymbol{s}_{i,j,k}) = \frac{\boldsymbol{s}_{i,j,k}}{\|\boldsymbol{s}_{i,j,k}\|} \cdot \frac{\|\boldsymbol{s}_{i,j,k}\|^2}{1 + \|\boldsymbol{s}_{i,j,k}\|^2} \tag{3}$$

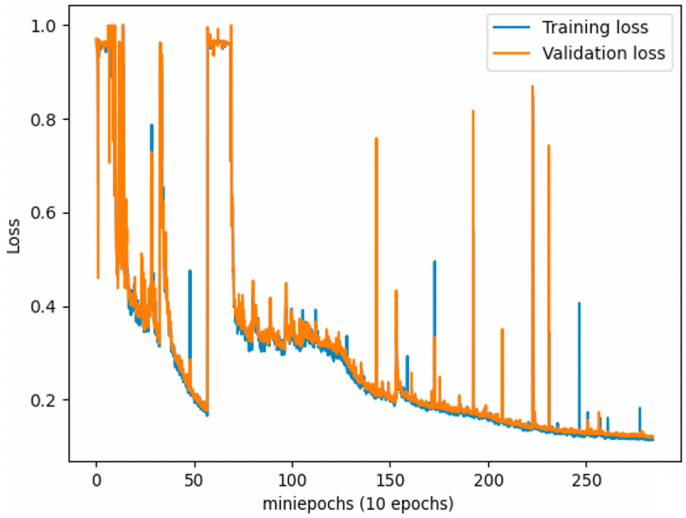
where $\boldsymbol{s}_{i,j,k}$ is the total input to the final layer capsule at the location (i,j,k), and $\boldsymbol{v}_{i,j,k}$ is the pose vector of the final layer capsule at that location.

Our experiments show that training with the forgiving paradigm is more stable and leads to faster convergence. When we did not convert the length of the pose vector $\|\boldsymbol{v}_{i,j,k}\|$ using the conversion function (formula 2), CapsNet training became unstable. Here, we show the evolution of the training set and the validation set losses during 10 epochs of training.

Training with the forgiving paradigm (with conversion)



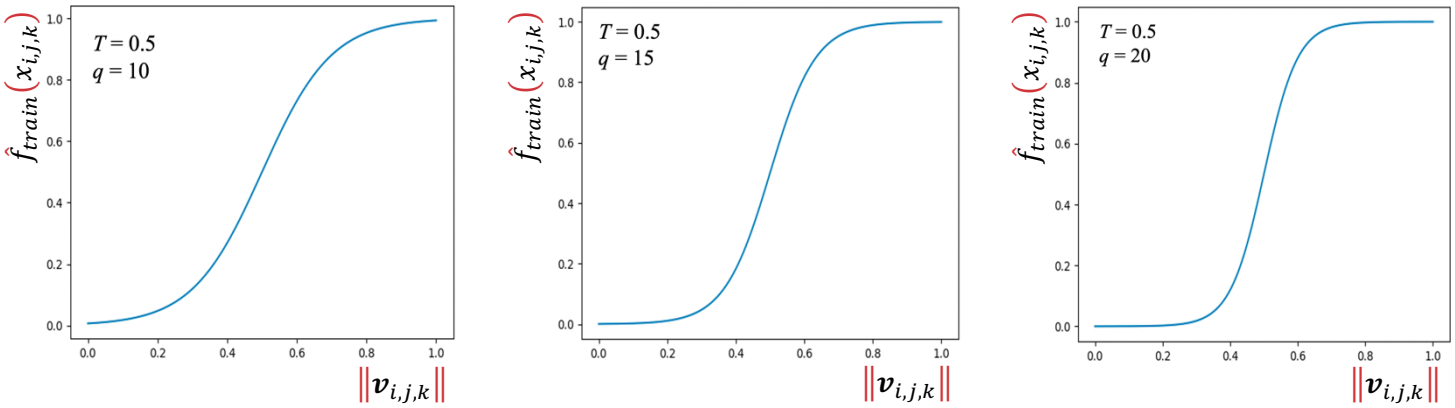
Training without the forgiving paradigm (without conversion)



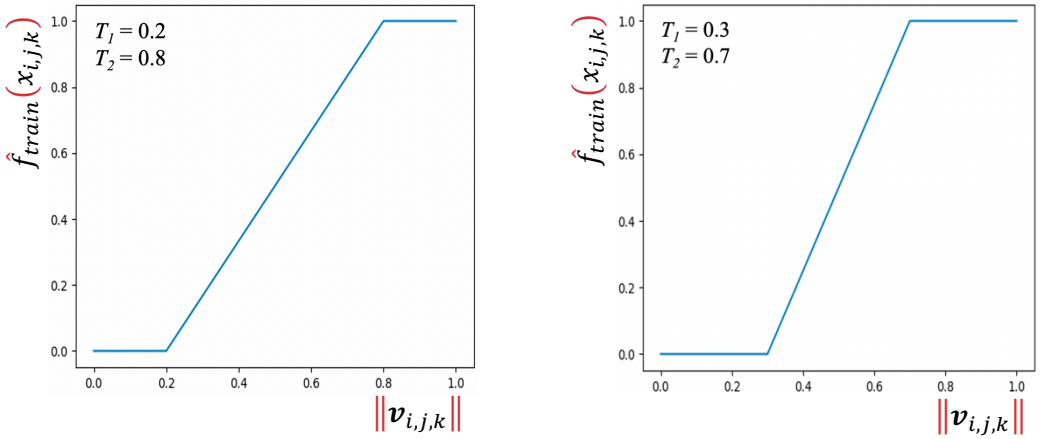
Additionally, we examined other conversion functions. None of these functions lead to better stability or faster convergence compared to the piecewise function (formula 2). We describe these functions here (together with their plots) so that other groups would be aware of other conversion functions that we think are suboptimal for this task:

$$\hat{f}_{train}(x_{i,j,k}) = sigmoid\left(q.\left(\|\boldsymbol{v}_{i,j,k}\| - T\right)\right) = \frac{1}{1 + e^{-q.(\|\boldsymbol{v}_{i,j,k}\| - T)}} \tag{4}$$

We set $T = 0.5$ and tried different values for q equal to 10, 15, and 20:



We also examined the piecewise conversion function (formula 2), but with different values for T_1 and T_2:



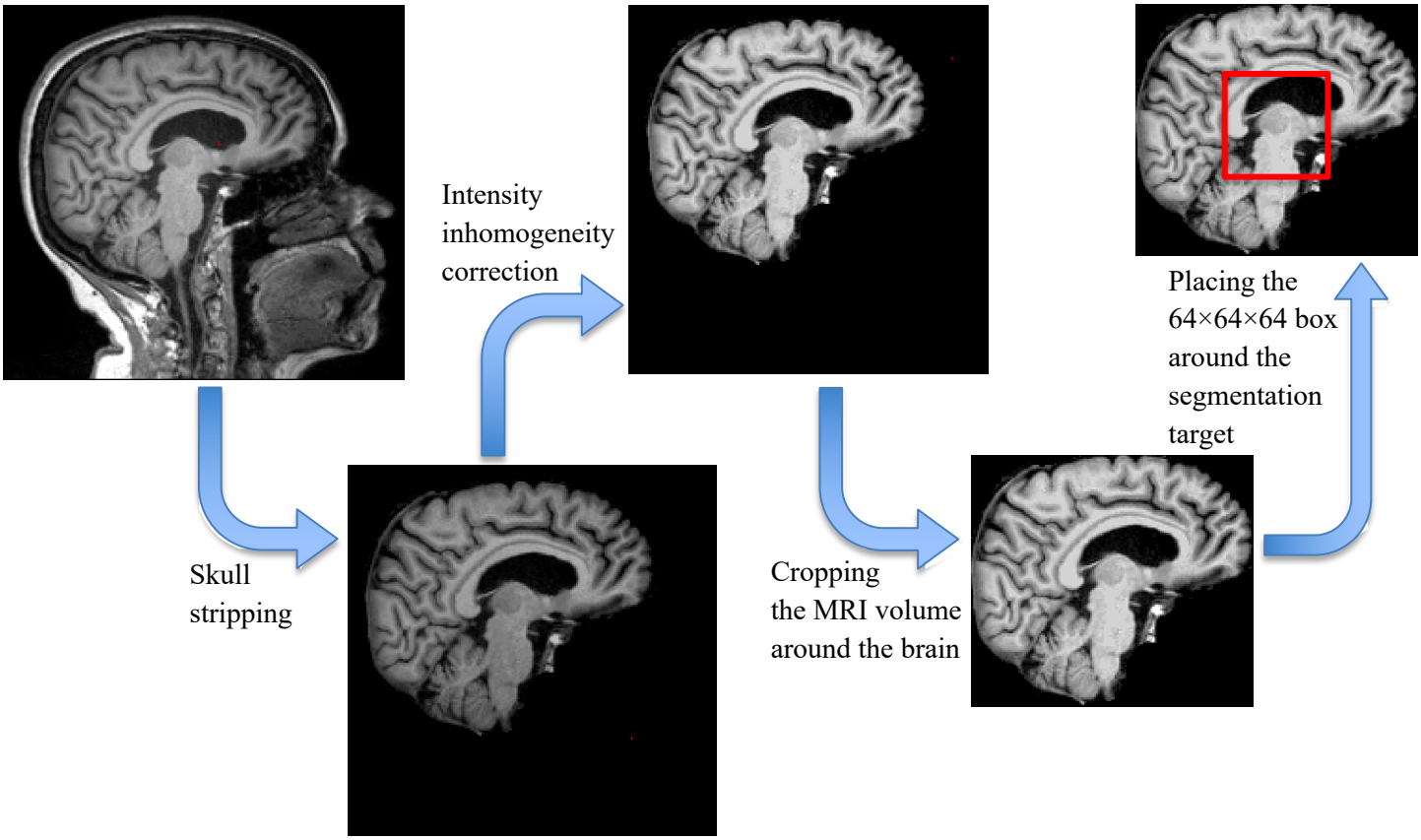
None of these conversion functions was as effective as the piecewise function with $T_1 = 0.1$ and $T_2 = 0.9$ in improving the stability and convergence of CapsNet training.

Appendix 4: Pre-Processing

We used FreeSurfer to correct for intensity inhomogeneities including B1-field variations. FreeSurfer first registers the brain to MNI305 atlas. Then, pixel intensities are used to roughly segment the white matter. The variations in the pixel intensities in the white matter are then used to estimate the B1 field map Finally, B1 bias field correction is done by dividing the pixel intensities by the estimated bias field.[1]

We used FreeSurfer[2] for skull stripping. Skull stripping includes removal of the skull, face, and neck, only leaving the brain. FreeSurfer uses a hybrid method of skull stripping that combines a watershed algorithm and a deformable surface model.[3] This method first roughly segments the white-matter based on pixel intensities. Then, watershed algorithms are used to find the gray-white matter junction and the brain surface. Next, a deformable surface model is used to model the brain surface. The curvature of the brain surface at each point is computed, and these curvatures are used to register the brain surface onto an atlas. The atlas is formed by computing the curvatures of the brain sulci and gyri in several subjects. Notably, the sulci and gyri of the brain surface are constant among all humans, constituting positive and negative curvatures that are present in any brain image (unless the brain surface is markedly distorted by space-occupying lesions). The reconstructed brain surface, registered to the atlas, is then corrected in case the curvatures in a particular region of the surface do not make sense. The resulting corrected brain surface model is used for skull stripping.[3]

To overcome memory limitations, we cropped 64×64×64-voxel boxes of the MRI volume that contained each segmentation target. The position of each box (e.g. for segmenting the right hippocampus) was determined by visually inspecting 20 brain MRI volumes, randomly selected from the training set. The visual inspection included moving the "crop" box (64×64×64 voxels) over the MRI volumes to find the optimal position of the crop box. The position of this box was then fixated (with regard to the center of the skull-stripped brain volume) for each brain structure and for all subjects in the training, validation, and test sets. This task was done by the first author (board-eligible radiologist with 9 years of experience in neuroimaging research).



Appendix 5: MRI parameters

Field strength = 3.0 tesla
Coil = 8HR Brain

Weighting = T1
Flip angle=8.0 degree
TR = 6.6 ms
TE = 2.8 ms
TI = 900.0 ms

Acquisition type = 3D
Acquisition plane = Sagittal
Matrix size = 256×256×166 pixels (X×Y×Z)
Pixel size = 1×1×1.2 mm (X×Y×Z)
Pixel spacing: along X direction = 1 mm; along Y direction= 1 mm

Appendix 6: Hyperparameters

References:

1. Dale AM, Fischl B, Sereno MI. Cortical surface-based analysis: segmentation and surface reconstruction.

2. Fischl B. FreeSurfer. *NeuroImage* 2012;62:774–81.

3. Ségonne F, Dale AM, Busa E, et al. A Hybrid Approach to the Skull Stripping Problem in MRI.