Effects of reducing model parameters on VoxelMorph Image Registration

Alan Yao, Sai Bhargav Vuppala, Vincent Wang

ABSTRACT

In this study, we introduce an adaptation of the VoxelMorph framework for deformable, pairwise medical image registration based on the work done by the original VoxelMorph project [1], with a focus on volumetric data composed of image slices. Traditional registration methods often suffer from computational inefficiency when applied to large datasets or intricate deformation models. Departing from conventional approaches, we redefine registration as a function mapping input image pairs to a deformation field aligning the images. This function is parameterized by a reduced-size convolutional neural network (CNN), optimizing network parameters using a set of volumetric images sliced into two-dimensional inputs. For training, we propose strategies for utilizing slices in volumetric data, including a design for the allocation of static and moving frames. We investigate the impact of reducing the model size, specifically the encoder-decoder architecture (g_theta), to examine the performance achieved with fewer parameters. Our findings underscore the potential of a more streamlined architecture for efficient medical image registration. The choice of static and moving frames, as well as the use of volumetric slices, demonstrates versatility in the application of VoxelMorph to various medical imaging scenarios. Our methodology not only promises to accelerate medical image analysis and processing pipelines but also opens avenues for novel directions in learning-based registration and its diverse applications.

INTRODUCTION

Medical image registration, specifically deformable registration, plays a crucial role in various medical imaging studies, providing a means to establish dense, non-linear correspondences between pairs of images, such as 3D magnetic resonance (MR) brain scans. Traditional methods tackle this task by solving optimization problems for each image pair, aligning voxels based on similar appearances while adhering to constraints on the registration mapping. However, the computational intensity of pairwise optimization makes these methods time-consuming, particularly for large datasets or complex deformation models [2-4].

In response to these challenges, we introduce an adapted version of the VoxelMorph framework, a learning-based approach to deformable, pairwise medical image registration originally presented in [1]. In contrast to traditional optimization-centric methods, the VoxelMorph framework formulates registration as a parametrized function learned from a collection of volumes. The function, implemented using a convolutional neural network (CNN), maps two n-D input volumes to a deformation field aligning them. The CNN parameters are optimized using a training set of volumes, enabling the learning of a common representation for aligning new volume pairs. This global optimization during training replaces the need for costly pair-specific optimizations during test time, resulting in rapid registration, even on a CPU [2-5].

Our work focuses on leveraging slices in volumetric data as the training dataset, offering flexibility in the allocation of static and moving frames. We also explore the impact of reducing the model size, particularly the encoder-decoder architecture, to assess performance with fewer

parameters. In the original learning-based framework of VoxelMorph, it presents two differentiable objective functions: an unsupervised approach that maximizes image matching objective functions based on intensities, and a second approach that leverages anatomical segmentations during training. We only present the first differentiable objective function, the unsupervised approach without segmentation.

While our study centers on the registration of 3D MR brain scans, the adaptability of our method extends to various registration tasks within and beyond the medical imaging domain. We evaluate our framework on a diverse multi-study dataset, showcasing its ability to achieve comparable accuracy to state-of-the-art methods while operating significantly faster. The key contributions of our work include the exploration of novel training strategies, the utilization of volumetric slices, and the investigation of model size reduction for efficient medical image registration. This paper extends and refines the preliminary work presented at the 2018 International Conference on Computer Vision and Pattern Recognition [1], incorporating expanded analyses and introducing auxiliary learning models for improved registration training speed on new test image pairs. We emphasize the importance of further investigations to comprehensively understand the implications and potential advancements of our proposed adaptations.

METHODS

1. UNET

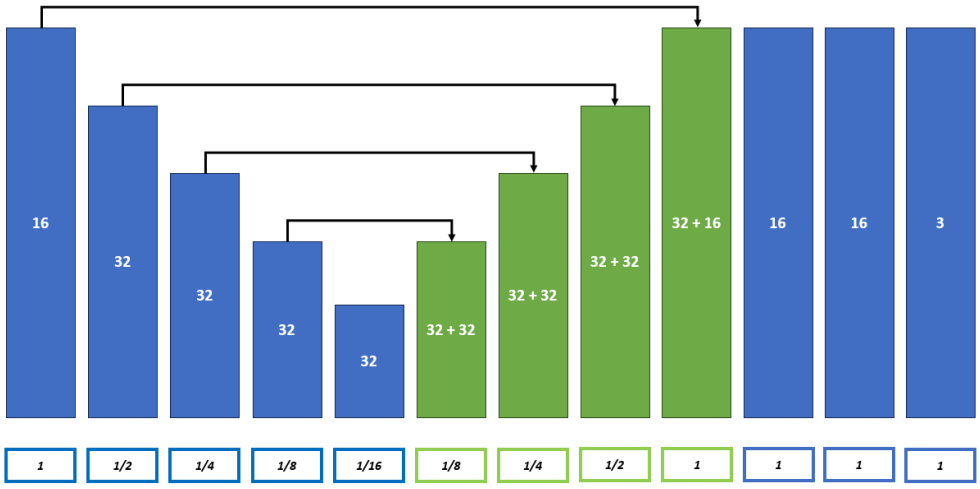

Figure 1: The UNET model architecture used as a baseline model (adapted from [1, Figure 3]). Arrows indicate forward-concatenated hidden layers. Upper boxes indicate the number of channels, while lower boxes indicate the per-dimension resolution ratio at each layer output. Blue layers represent an activation + downsampling convolutional output, and green layers represent a concatenation + upsample + activation + convolutional output.

For the initial architecture, the UNET model was replicated from [1] as shown above in Figure 1. The input of the model was a 2-channel 3D voxel image, with two channels representing the fixed and moving inputs, while the output was a 3-channel 3D deformation map, with cartesian displacements for each point. Initial convolutions in the encoder are done with kernel size 3 and stride 2 to halve the resolution at each layer, while subsequent convolutions in the decoder are done with kernel size 3 and stride 1, using upsampling layers instead to change resolution.

In order to test the effect of reducing the size of the model, two other architectures were considered (and a third consisting of a combination of the first two). The first sub-architecture involved the addition of an initial max-pooling layer, in addition to one additional final upsampling layer (Figure 2).

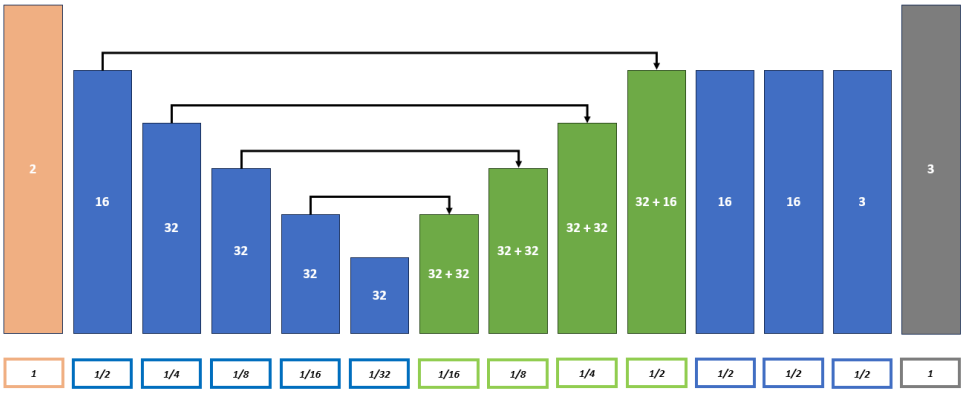

Figure 2: Modified VoxelMorph architecture #1. Note the addition of an initial max-pooling layer (orange) and an additional upsampling layer without convolution (gray).

Using this architecture, the overall size of layer operations performed on each input is reduced by a factor of ⅛ (½ per dimension, for three dimensions), improving the speed at which the model can be trained. In addition, the amount of memory required improves as well since the size of layers that UNET needs to save for concatenation is also reduced by a factor of ⅛, allowing for an increase in the batch size during training, further improving speeds.
The second modified architecture removed four intermediate layers within the UNET architecture (Figure 3).

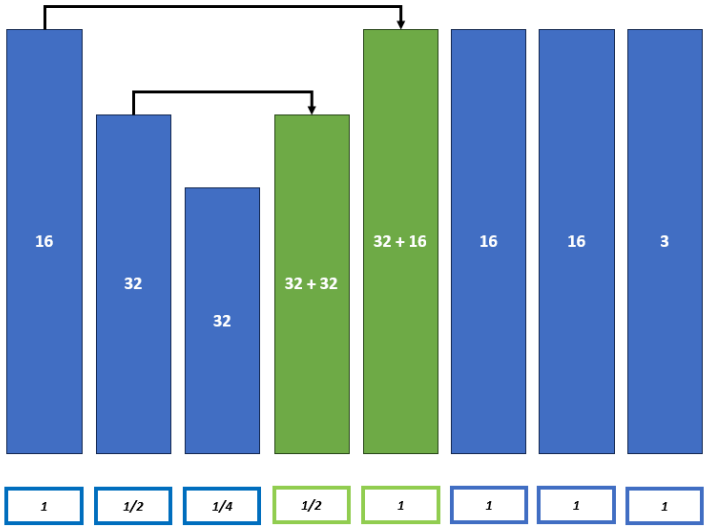

Figure 3: Modified VoxelMorph architecture #2. Four layers have been removed in the middle of the network–two downsampling convolutional layers, and two upsampling convolutional layers.

Similar to the first modified VoxelMorph architecture, this reduced model has the benefit of fewer computations (improving speed), as well as a smaller memory requirement (fewer layers required to be stored in memory for future concatenation during decoding).
Finally, a third modified architecture was also tested as a combination of the first and second architectures–i.e. the addition of a max-pool and additional upsampling layer, alongside removing four intermediate layers within the UNET model.

2. VxmDense
   VxmDense is a class that extends UNET functionality. It concatenates the input moving and reference images and feeds them into the UNET. VxmDense also processes the UNET output to generate a three-channel flow field (the deformation/registration field) using trainable parameters, and then warps the moving image with the flow field by calling the SpatialTransformer. The final output of VxmDense consists of the warped image and the flow field.

3. SpatialTransformer
   The SpatialTransformer was replicated from [1]. It takes a 3-channel flow field and a 3D image as inputs. It initializes a grid matching the shape of the source image. The flow field, which stores the new x, y, and z voxel locations in the source image, is added to the grid to create a displacement map each time the function is called. Every pixel/voxel in the source image is sampled based on the transformed grid to form the warped image. Because the displacement map might contain sub-pixel locations (non-integer coordinates), a bilinear interpolation was performed based on the eight neighboring voxels.

4. Train loader
   The dataset used in this study is the OASIS dataset, which was loaded using the neurite-OASIS from Adalca. The dataset has 414 T1 MRIs from the OASIS dataset, processed using FreeSurfer and SAMSEG. To create a data loader as an input for our model, we define a custom dataset class, CustomDataset, for handling medical imaging data in the NIfTI format using PyTorch. The dataset is designed to load pairs of aligned images, 'aligned_orig.nii.gz' and 'aligned_norm.nii.gz,' from a specified root directory. It utilizes the Torchvision library for data transformations and PyTorch's DataLoader for efficient batch loading. The dataset can be instantiated with the root directory containing patient subdirectories, and a DataLoader is created for iterating through the dataset in batches. The example usage demonstrates loading the dataset and iterating through the DataLoader to obtain batches of stacked images. Each batch contains two images (orig and norm) stacked along a new axis, and the shape of the resulting batch is printed, confirming the expected format.

5. Loss Functions
   Three loss functions were implemented in our code as replicated from [1]. The first loss function is the mean-squared-error (MSE) between the warped image and the reference (fixed) image. The second loss function is the negative normalized cross-correlation (NCC) of those two images. The local cross-correlation of reference image f and moving image m with displacement map ϕ can be expressed as: $CC(f, m\circ\phi) = \sum_{p\in\Omega} \frac{(\sum_{p_i}(f(p_i)-\hat{f}(p_i))([m\circ\phi](p_i)-[\widehat{m\circ\phi}](p)))^2}{((\sum_{p_i}(f(p_i)-\hat{f}(p_i))^2)(\sum_{p_i}([m\circ\phi](p_i)-[\widehat{m\circ\phi}](p))^2))}$, where $\hat{f}(p)$ and $[\widehat{m\circ\phi}](p)$ denote the local mean intensity: $\hat{f}(p) = \sum_{p_i} f(p_i)/n^3$. $m\circ\phi$ is the warped image. The third loss function is a regularization term to promote the smoothness of the warped image. It computes the L2-norm of the x, y, and z gradients of the warped image. The gradient is calculated

using the finite difference method. The final loss function combines either MSE or NCC with this gradient regularization term.

6. Plotting
   This code utilizes the Matplotlib library to generate a 3x4 grid of subplots, creating a visual representation of brain scans reconstructed through the VoxelMorph model. The subplots are organized into three rows, each corresponding to a different type of brain slice: the fixed slice, the deformed moving slice, and the original moving slice. The code iterates over a data loader, processing each batch using a trained VoxelMorph model. For each iteration, the code extracts fixed and moving slices, performs image registration using the model, and then visualizes the results by displaying the fixed slice, the deformed moving slice, and the original moving slice in separate subplots. The loop is limited to processing the first four batches to create a concise visual summary. This code provides a snapshot of the model's performance by showcasing registered brain slices in a structured grid, aiding in the evaluation of the VoxelMorph model's efficacy in reconstructing brain scans.

RESULTS

1. Loss Function
   In this study, we conducted a comparative analysis of two distinct loss functions, namely Mean Squared Error (MSE) and Normalized Cross-Correlation (NCC), integrated with a smooth loss framework for image reconstruction. The primary focus was to evaluate the efficacy of these loss functions in achieving optimal reconstruction performance.
   The application of MSE as the loss function for image reconstruction yielded suboptimal results. Despite multiple iterations and the inclusion of smooth loss, the MSE-based reconstruction exhibited challenges in converging to zero effectively. The overall loss trajectory indicated a slow convergence, and when visualized, the reconstructed images did not exhibit satisfactory fidelity. The inadequacies in the MSE-based reconstruction suggested limitations in capturing the intricacies of the underlying image features.
   Contrastingly, the utilization of Normalized Cross-Correlation (NCC) as the loss function demonstrated promising outcomes for image reconstruction. The NCC reconstruction exhibited visually pleasing results, with the loss consistently decreasing and reaching negative values. Although the initial progress was slow, a notable improvement was observed when the size of the UNET architecture was reduced. This adjustment contributed to a more effective optimization process, leading to enhanced reconstruction quality.

2. Permutations of the model architecture
   For the training that was done with a combination of both size-reduction methods (max-pooling and removal of convolutional layers), the total time until completion was 1hr 11min 22s.
   It produced a minimum loss of -.49 with the loss for each training step shown in Figure 4.

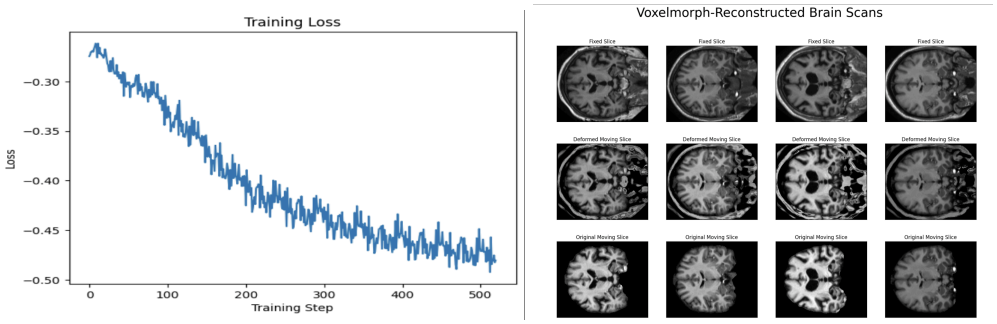

Figure 4: For the case when both max pooling is increased and UNet has a reduced size, the relationship between training step and loss during training of the model seen on the left and 4 random slices of the fixed, deformed, and original moved scan seen on the right.

For training done only max-pooling reduction, the results are as follows. The total time was 1 hr 11 min and 22s with a final loss of -0.48.

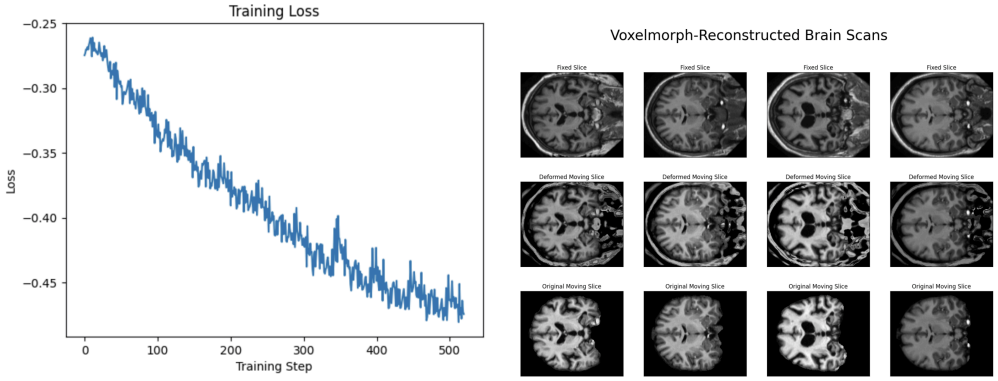

Figure 5: For the case when only max pooling is increased, the relationship between training step and loss during training of the model seen on the left and 4 random slices of the fixed, deformed, and original moved scan seen on the right.

For the default VoxelMorph model with no reduction techniques applied, it took 1 hr 40 min to conduct 10 epochs and almost had a final loss reaching -0.6.

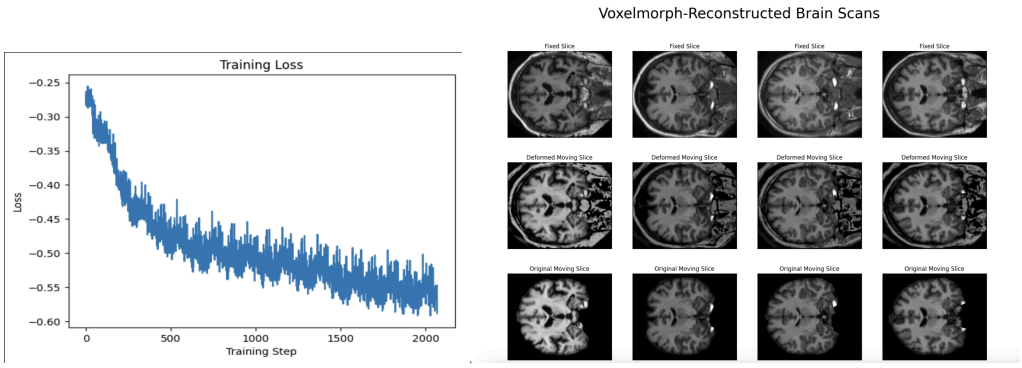

Figure 6: For the case when neither max pooling is increased nor UNet has a reduced size, the relationship between training step and loss during training of the model seen on the left and 4 random slices of the fixed, deformed, and original moved scan seen on the right.

The final permutation was to only reduce the size of Unet which resulted in an overall time of 1 hr 36 min to conduct 10 epochs with a final loss of approximately -0.5 as seen in Figure 7.

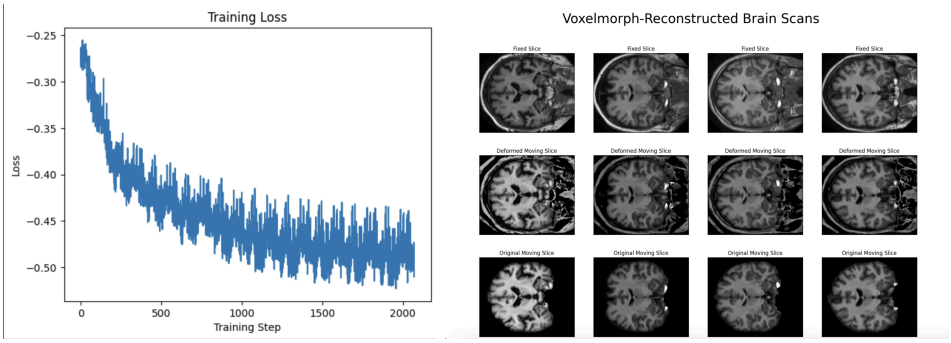

Figure 7: For the case when only UNet has a reduced size, the relationship between training step and loss during training of the model seen on the left and 4 random slices of the fixed, deformed, and original moved scan seen on the right.

DISCUSSION

1. Loss

   The comparative analysis between MSE and NCC underscored the superior performance of NCC in the context of image reconstruction. The negative loss values in NCC, indicative of improved alignment between the reconstructed and target images, contrasted with the less satisfactory convergence and visual quality observed in MSE-based reconstruction. These findings suggest that NCC, when combined with a smooth loss framework, holds promise for achieving more accurate and visually appealing image reconstructions.

   The observed improvement in NCC performance following the reduction in UNET size implies that careful consideration of network architecture plays a crucial role in the effectiveness of loss functions. Fine-tuning such parameters can significantly impact convergence rates and overall reconstruction quality. In summary, our comparative analysis highlights the limitations of MSE for image reconstruction, while showcasing the potential of NCC when integrated with a smooth loss framework. These findings provide valuable insights for researchers and practitioners seeking optimal strategies for image reconstruction tasks, emphasizing the importance of selecting appropriate loss functions and optimizing network architecture.

   When talking about overall loss for each permutation of the model architecture, the loss did not converge completely for the first two permutations, when the max pooling was increased and when Unet size was reduced. This could be solved with a couple more epochs but it seems that reducing the size of the architecture requires more epochs to converge the model. This does not seem to affect the output image though as shown in Figures 4-7, all of the slices for all of the permutations look similar.

2. Changes from Original Code

   The goal of this project was to decrease the parameters (g_theta) of the model and figure out if it was possible to replicate the results of the original as well as decrease the time it takes to train the model. As mentioned above when discussing the UNET architecture, there are 4 permutations of the model that are being tested for evaluation. The first permutation tested was when the maxpool was increased and the UNET structure was reduced. As mentioned in the results, the loss value for this permutation was less than the other models although it had a significantly reduced time to run 10 epochs. When comparing the loss values of all four models, it can be seen that the higher the

loss after the 10 epochs, the more intensive the model was. The highest loss value achieved was approximately -0.6 which was done for the model that did not have any modifications from the original. The lowest loss value was the model that had both of the changes done to it. The loss value differs by about 0.1 but the images that are output from each of the models is not too different from each other.

We were also able to visually compare how well the deformed images compared to the fixed images for each of the other models and it seems as though all of the models were able to produce clear results with no extra noise or artifacts. We noticed that no matter which model was run, the level of clarity from the images is similar to each other with no difference in amount of artifact or noise. This shows that all four of the models are reconstructing the images at the same level despite the lower level of parameters. We were also able to compare the moved, original and output for each of the models. There is no significant difference to note between the 4 models.

The last factor to review is the time it took to run each of the models. Comparing all four, the fastest two models were the two that used the max-pooling reduction technique. Both of those models ran 10 epochs in 1 hour and 11 minutes while both of the other models ran 10 epochs in around 1 hour and 40 minutes. This is due to the batch size that can be run based on the size of the architecture. Without the maxpool increased and present outside of the UNET, the model is too computationally intensive for any batch size greater than 2, resulting in a greater time to complete 10 epochs and converge [5]. For the two faster models, the highest possible batch size was 8, resulting in a much faster time to complete 10 epochs and converge.

CONCLUSION

In conclusion, our study primarily concentrated on a comprehensive comparison of model architectures for image reconstruction, specifically evaluating the impact of changes such as increased max pooling and reduced UNET size. While assessing two distinct loss functions, Mean Squared Error (MSE) and Normalized Cross-Correlation (NCC), integrated with a smooth loss framework, the emphasis shifted toward understanding the interplay between architecture adjustments and reconstruction outcomes. The findings demonstrated that MSE-based reconstructions exhibited suboptimal performance, highlighting the limitations of this loss function in capturing image intricacies. In contrast, NCC, when coupled with a smooth loss framework, showcased promising outcomes, with improved alignment and visually pleasing results. The study revealed that careful consideration of model parameters significantly influenced convergence rates, with reductions in UNET size enhancing NCC performance. Despite variations in convergence across model permutations, the output images consistently displayed similar levels of clarity and quality, emphasizing the robustness of the proposed architectural modifications. Notably, the two models with increased max pooling outside UNET converged faster, contributing to a reduction in training time without compromising image quality. This study provides valuable insights into optimizing model architecture for image reconstruction tasks, surpassing the focus on loss functions alone.

REFERENCES

[1] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, "An unsupervised learning model for deformable medical image registration," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9252–9260.

[2] A. Klein, J. Andersson, B. A. Ardekani, J. Ashburner, B. Avants, M.- C. Chiang, G. E. Christensen, D. L. Collins, J. Gee, P. Hellier *et al.*, "Evaluation of 14 nonlinear deformation algorithms applied to human brain mri registration," *Neuroimage*, vol. 46(3), pp. 786–802, 2009.

[3] B. B. Avants, N. J. Tustison, G. Song, P. A. Cook, A. Klein, and J. C. Gee, "A reproducible evaluation of ants similarity metric performance in brain image registration," *Neuroimage*, vol. 54, no. 3, pp. 2033–2044, 2011.

[4] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 2, 2017, p. 6.

[5] M. Modat, G. R. Ridgway, Z. A. Taylor, M. Lehmann, J. Barnes, D. J. Hawkes, N. C. Fox, and S. Ourselin, "Fast free-form deformation using graphics processing units," *Computer methods and programs in biomedicine*, vol. 98, no. 3, pp. 278–284, 2010.

Contributions:

Alan - Alan created the loss function, VxmDense, and the SpatialTransformer. Alan his corresponding sections to his code in the methods section

Vincent - Vincent created the original UNET, the parameter change for our code, and the train function as well as the plot function. Vincent wrote the UNET part of the paper as well as readme.

Sai - For the code, Sai contributed by creating the train loader and bringing in the OASIS dataset. Sai wrote the abstract, introduction, trainloader and plotting in methods, results, discussion, and conclusion.

Everyone - We all reviewed each section of the paper and we reviewed each section of the code. We all helped to comment out the code as well as clean it up.