Machine Learning Enabled Brain Segmentation for Small Animal Image Registration

Hendrik Klug[1] Horea-Ioan Ioanas[2] Markus Rudin[2]
[1]Department of Information Technology and Electrical Engineering, ETH
[2]Institute for Biomedical Engineering, ETH and University of Zurich

Abstract — Cross-subject and cross-study comparability of imaging data in general, and magnetic resonance imaging (MRI) data in particular, is contingent on the quality of registration to a standard reference space. Current methods rely on full image processing, with high varying intensities outside the Region Of Interest that interfere with registration. Applying the processing to a masked image improves the quality of the latter. Here, we present a deep learning enabled framework for segmentation of brain tissue in functional and structural MR images, as an additional step in the SAMRI registration workflow.

Background

In order to make meaningful comparisons across subjects inside a study, it is imperative that the images lie in a standard reference frame. Because of positioning imprecision and anatomical animal variations, this is not the case for the original MR acquired images. To solve this issue, the images need to be projected into the reference frame via registration [1, 2]. Intensities outside the brain region of a mouse MR image present high variations and bias the registration process. As a remedy, it would be useful to extract the region of interest and perform the registration on it. For this purpose, we propose a machine learning enabled brain extraction in an additional node to the workflow presented by Ioanas et al. [3]. The additional node creates a mask of the brain region with segmentation using a classifier and masks the image such that only the region of interest remains.

Convolutional Neural Networks

In recent years it has been shown that convolutional neural network give the best results for semantic image segmentation in terms of precision and flexibility [4] [5]. Training a convolutional neural network into a classifier is a supervised method, meaning that the model needs to learn its parameters based on observations of data.

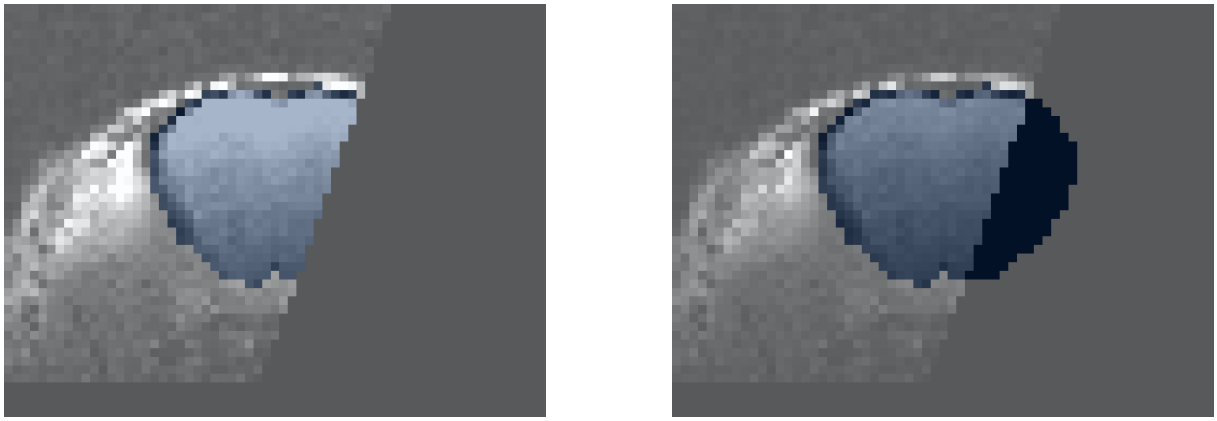
The training data set of a classifier is as important as the architecture of the model itself. To improve general-purpose application, training examples need to be drawn from a usually unknown probability distribution, that is expected to be representative of the space of occurrences. We define the space of occurrences as the space of which the data of interest is drawn from. In our case this consists of all the different mouse brain MRI data sets coming from multiple experiments, with their corresponding labels. Ideally experiment setups are uniform and the resulting data does not differ much, but small variations in the experiment setup and animal size are unavoidable. Based on an approximation of the occurrence space, the network has to build a general model that enables it to extrapolate and produce sufficiently accurate predictions in new cases. Manually creating annotations as required to train a deep-learning classifier for high-resolution data is often infeasible, since it requires manual expert segmentation of vast amounts of slices.

Here we take as data set, images that were registered through the SAMRI workflow into a reference frame defined by Toronto Hospital for Sick Children Mouse Imaging Center together with a template mask. While our purpose was to create a workflow that generates better masks than the ones used for registration, we showed that the latter could be used as training data for the deep-learning model, by applying small changes to them.

The Classifier

Model

As the architecture of the classifier, the U-Net from Ronneberger et al [5] was chosen based on its high performance in the field of biomedical image segmentation. This is a convolutional neural network that consists of a contracting path that captures context in addition to a symmetric expanding path that enables precise localisation. Localisation in this context means that a class label is assigned to each pixel. We used the U-Net implementation from zhixuhao [6], written in Keras. Keras is a high-level neural networks API, written in Python and capable of running



(a) Example of a preprocessed slice.

(b) Example of an unpreprocessed slice.

Figure 1: The preprocessing removes the mask there, where the image-pixelvalues are 0. Plots of the same image, superposed with the template mask, with and without preprocessing.

on top of TensorFlow, CNTK, or Theano. It allows for a easily readable code and makes it thus easier to reproduce.

The implementation of the U-Net from zhixuhao has two drop-out layers in addition to the original one. A drop-out layer randomly sets a fraction of input units from the layer to 0 at each update during training time. The set fraction rate is 0.5. It is known that dropout helps prevent overfitting and greatly improves the performance of deep learning models [7].

Three losses were tested for the training of the model, namely the Dice-loss, the binary-cross-entropy loss and the sum of both.

The Dice-loss is computed from the Dice score. It calculates the similarity of two binary samples X and Y with

$$D_{coef} = \frac{2|X \cap Y|}{|X| + |Y|} \tag{1}$$

It is a quantity ranging from 0 to 1 that is to be maximised. The loss is then calculated with $1 - D_{coef}$. Because the Dice loss is not differentiable, small changes need to be made. In our case, the two samples to be compared are normalised, grey valued images and are thus not binary but have values between 0 and 1. Additionally, instead of using the logical operation *and*, element wise products are used to approximate the non-differentiable intersection operation. To avoid a division by zero, +1 is added on the numerator and denominator.

Because many more pixels in the masks are 0 than 1, there is a class imbalance problem. It is a problem, because in this case a false positive gives a much higher loss than a false negative. For example, predicting only black would give an acceptable loss, while predicting only white pixels would not. Using the Dice coefficient as a loss function for training should make it invariant to this class imbalance problem as stated by Fausto Milletari et al. in [8].

The binary cross-entropy loss, also called Log loss, is defined by:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1-y_i) \cdot log(1-p(y_i)) \tag{2}$$

For pixel values of 0 and 1, it adds $log(p(y))$ for each white pixel (y=1) and $log(1 - p(y))$ for every black pixel (y=0) to the loss.

We quickly realised that the Dice-loss gives the best results for our problem and therefore used it to train the model.)TODO:compare results)

Data Set

The data set consists of 3D MR images taken from an aggregation of three studies; irsabi , opfvta [9], drlfom [10] and other unpublished data, acquired with similar parameters.

The images are transformed into a standard space with one defined mask via SAMRI [11] and are thus defined in the same affine space. SAMRI is a data analysis package of the ETH/UZH Institute for Biomedical Engineering. It is equipped with an optimized registration workflow and standard geometric space for small animal brain imaging [3].

Because of variance in mouse brain anatomy and in the experiment setup, some of the transformed data do not overlap perfectly with the reference template. To filter these images out, most of the incongruent slices were removed manually from the data set.

For the registration of the images, a padding was needed to make the originally not affine space affine. Due to this, the 3D volumes present many zero-valued slices, some of them overlapping with the mask.

Since it is not wanted for the model to predict a mask on black slices, the mask is set to zero where the image is as well. Because

some pixels representing the brain tissue are zero-valued, holes result from this operation. To patch these, the function $binary\_fill\_holes$ from scipy.ndimage.morphology [12] is used. An example of this can be seen in fig. 1.

In the coronal view, each slice of the transformed data is originally of shape (63, 48), matching the reference space resolution of 200 µm. It is then reshaped into (64, 64) by adding a zero padding to the border.

Finally, the images are normalised by first clipping them from the minimum to the 99th percentile of the data in order to remove outliers and then divided by the maximum.

The data set is separated into Training, Validation and Test sets such that 90% of the total data are used for training and 10% for testing. The Validation set is used for the optimisation of hyper parameters while the Test set is used as a measure of extrapolation capability.

Data Augmentation

Because of diverse settings in the experiment setup, including animal manipulations causing artifacts, MR image quality can differ substantially between labs and even individual study populations. To account for these variations, we apply an extensive set of transformations to our data. This includes rotations of up to 90°, a width and height shift range of 30 pixels, a shear range of 5 pixels, zoom range of 0.2 and horizontal as well as vertical flips.

This not only increases the data set size but also makes it more representative of the general data distribution of Mice brain MR images and results in a model with a better generalisation capability.

Many more sophisticated methods have been tested, but it has been shown that one of the more successful data augmentation strategies is the simple transformations mentioned above [13].

Training

The model was trained slice wise, with the coronal view and 600 as the maximum number of epochs. The coronal view was chosen over the axial one, because the shapes of the masks are much simpler in the coronal view and thus easier to learn for the network. Additionally the coronal view has the advantage of higher resolution as the MR images were recorded coronally.

Because of the extensive transformations on the images (see section 2.3), the augmented data presents much variation. To alleviate the learning process of the model, we have first trained it on the data without augmentation and then on the augmented data.

To improve the learning process of the network, two callbacks from Keras were used [14]. "ReduceLROnPlateau" reduces the learning rate when the validation loss has stopped improving and "EarlyStopping" stops the training when the validation loss has stopped improving for a number of epochs. The latter reduces computation time and prevents overfitting.

Masking

In order to improve the SAMRI registration workflow, an additional node is implemented where the images are masked, such that only the brain region remains. For this process, the image needs to be preprocessed using the operations described in section 2.2. Since the classifier was trained to predict on images of shape (64, 64), the input needs to be reshaped. If one dimension is bigger than 64, it reshaped into 64 using the function $cv2.resize$ from the opencv python package. TODO:cite If one dimension is smaller, a zero padding is added to the border. For each slice in the image, the classifier then predicts a segmentation of the brain, which is used to create a 3D mask. The latter is then reshaped into the original shape inverting the preprocessing step, either with the opencv resize method or by cropping. The workflow then continues with only the Region Of Interest inside the image.

Evaluation

For the quality control of the workflow, we first evaluate the segmentation process and then the registration. The segmentations of the classifier should be as precise as possible for the masking process, which in return is then used to improve the registration.

As stated by Ioanas et al. in TODO:cite a major challenge of registration QC is that a perfect mapping from the measured image to the template is undefined. To address this challenge, they developed four alternative evaluation metrics: volume conservation, smoothness conservation, functional analysis, and variance analysis. We will use these metrics to benchmark our workflow against theirs.

Segmentation

Quality control of our classifier is difficult in the sense that it should predict a better mask than the template. Nevertheless, it is use-full to verify if the output is similar, as it should be. As a similarity metric we have used the Dice score (see eq. (1)). The median Dice score taken on every slice of the test data set is $D_{coef} = 0.973 \sim 1$. The prediction thus have only minor changes in comparison with the template, which should represent small improvements.

It is hard to measure how much better the predicted masks from the classifier are, compared with the template, because both are very accurate. They both correlate similarily with the images for example, as can be seen in **??**: 0.782 against 0.782.

Where the classifier clearly has the advantage is on the blacklisted slices. Due to the data agumentation on the training set, the classifier is trained to predict well on incongruent data and thus has a better correlation with the blacklisted slices than the template (0.737 against 0.685). Here the argument could be

	x_test	y_test	y_pred
x_test	1.0	0.7823169405702793	0.7816228746651158
y_test	0.7823169405702793	1.0	0.9873468560807918
y_pred	0.7816228746651158	0.9873468560807918	1.0

(a) Cross Correlations between the Test Set and the predictions.

	x_bl	y_bl	y_pred
x_bl	1.0	0.6854015792918978	0.7370047119017832
y_bl	0.6854015792918978	1.0	0.8685613856702904
y_pred	0.7370047119017832	0.8685613856702904	1.0

(b) Cross Correlations between the blacklisted slices and the predictions.

Figure 2: Overall the predictions of the classifier correlate better with the images, when taking the blacklisted slices into account. Two tables comparing the correlation of the classifiers prediction and the template mask with the images.

brought that one could do the inverse: taking all the bad predictions of the classifier and comparing them to the template. However, in total, the classifier has less predicitons with a lower correlation than: compared to the template: TODO

As an evaluation of the registration, we make use of the quality control from TODO:cite

Volume Conservation

Volume conservation is based on the assumption that the total volume of the scanned segment of the brain should remain roughly constant after preprocessing. Beyond just size differences between the acquired data and the target template, a volume increase may indicate that the brain was stretched to fill in template brain space not covered by the scan, while a volume decrease might indicate that non-brain voxels were introduced into the template brain space. For this analysis we compute a Volume Conservation Factor (VCF), whereby volume conservation is highest for a VCF equal to 1.

we note that in the described dataset VCF is sensitive to the workflow ($F_{1,134} = 9.805$, $p = 0.0021$),

The performance of the Generic SAMRI workflow in conjunction with the Generic template is significantly different from that of the Generic SAMRI workflow with the additional masking node, yielding a two-tailed p-value of 0.0018. Moreover, the root mean squared error ratio strongly favours the Generic workflow

($\mathrm{RMSE_{GM}}/\mathrm{RMSE_{G}} \simeq 2.3$).

Descriptively, we observe that the Generic level of the template variable introduces a notable volume loss (VCF of ?? while the Legacy level of the workflow variable introduces a volume gain (VCF of 0.06, 95%CI: 0.03 to 0.10).

With respect to the data break-up by contrast (CBV versus BOLD, fig. 3a), we see no notable main effect for the contrast variable (VCF of -0.02, 95%CI: -0.08 to 0.03).

Smoothness Conservation

A further aspect of preprocessing quality is the resulting image smoothness. Although controlled smoothing is a valuable preprocessing tool used to increase the signal-to-noise ratio (SNR), uncontrolled smoothness limits operator discretion in the trade-off between SNR and feature granularity. Uncontrolled smoothness can thus lead to undocumented and implicit loss of spatial resolution and is therefore associated with inferior anatomical alignment [**?**]. We employ a Smoothness Conservation Factor (SCF), expressing the ratio between the smoothness of the preprocessed images and the smoothness of the original images.

With respect to the data shown in **??**, we note that SCF is sensitive to the workflow ($F_{1,134} = 3.144$, $p = 0.078$),

The performance of the Generic SAMRI workflow in conjunction with the Generic template is significantly different from that of the Legacy workflow in conjunction with the Legacy template, yielding a two-tailed p-value of 0.036. In this comparison, the root mean squared error ratio favours the Generic workflow

($\mathrm{RMSE_{GM}}/\mathrm{RMSE_{G}} \simeq 1.4$).

Descriptively, we observe that the Legacy level of the template variable introduces a smoothness reduction while the Legacy level of the workflow variable introduces a smoothness gain (SCF of 0.02, 95%CI: 0.00 to 0.04). Further, we note that there is a strong variance increase for the Legacy workflow ratio = $generic_masked/generic$ = (1.85

Given the break-up by contrast shown in fig. 3b, we see only very weak effect sizes for the contrast variable (SCF of 0.06, 95%CI: 0.02 to 0.09)

Methods

The slice-wise predictions of the model are reconstructed to a 3D mask via the command *Nifit1Image* from the neuro-imaging python package nibabel [15]. This is done using the same affine space as the input image.

For the training of the classifier, the data are separated into a Training, Validation and Test set with the help of the function *train_test_split* from the package sklearn.model_selection [16].

References

[1] J B Antoine Maintz and Max A Viergever. An Overview of Medical Image Registration Methods. page 22.

[2] Aristeidis Sotiras, Christos Davatzikos, and Nikos Paragios. Deformable Medical Image Registration: A Survey. *IEEE Transactions on Medical Imaging*, 32(7):1153–1190, July 2013. ISSN 1558-254X. doi: 10.1109/TMI.2013.2265603.

[3] Horea-Ioan Ioanas, Markus Marks, Mehmet Fatih Yanik, and Markus Rudin. An Optimized Registration Workflow and Standard Geometric Space for Small Animal Brain Imaging. preprint, Neuroscience, April 2019. URL `http://biorxiv.org/lookup/doi/10.1101/619650`.

[4] Qichuan Geng, Zhong Zhou, and Xiaochun Cao. Survey of recent progress in semantic image segmentation with CNNs. *Science China Information Sciences*, 61(5):051101, May 2018. ISSN 1674-733X, 1869-1919. doi: 10.1007/s11432-017-9189-6. URL `http://link.springer.com/10.1007/s11432-017-9189-6`.

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv:1505.04597 [cs]*, May 2015. URL `http://arxiv.org/abs/1505.04597`. arXiv: 1505.04597 version: 1.

[6] zhixuhao. zhixuhao/unet, January 2020. URL `https://github.com/zhixuhao/unet`. original-date: 2017-04-06T01:58:15Z.

[7] Nitish Srivastava, Georey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overtting. page 30.

[8] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *arXiv:1606.04797 [cs]*, June 2016. URL `http://arxiv.org/abs/1606.04797`. arXiv: 1606.04797.

[9] Horea-Ioan Ioanas, Bechara John Saab, and Markus Rudin. A Whole-Brain Map and Assay Parameter Analysis of Mouse VTA Dopaminergic Activation. page 19, .

[10] Horea-Ioan Ioanas, Bechara John Saab, and Markus Rudin. Effects of Acute and Chronic Reuptake Inhibition on Optogenetically Induced Serotonergic Activity. page 20, .

[11] IBT-FMI/SAMRI, December 2019. URL `https://github.com/IBT-FMI/SAMRI`. original-date: 2015-04-27T00:26:08Z.

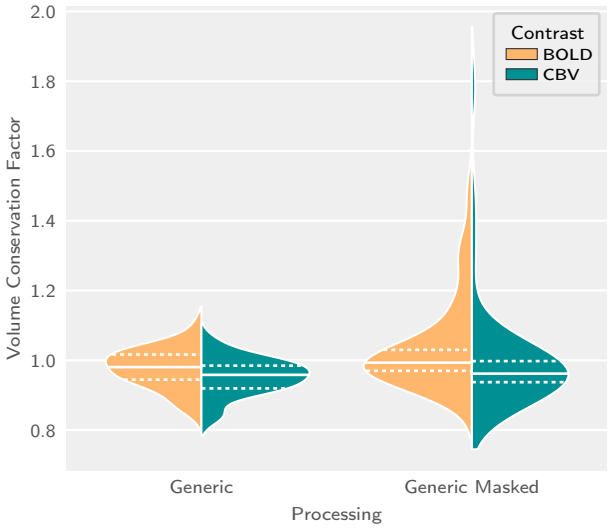
[12] Multi-dimensional image processing (scipy.ndimage) — SciPy v1.4.1 Reference Guide, . URL `https://docs.scipy.org/doc/scipy/reference/ndimage.html#morphology`.

[13] Luis Perez and Jason Wang. The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *arXiv:1712.04621 [cs]*, December 2017. URL `http://arxiv.org/abs/1712.04621`. arXiv: 1712.04621.

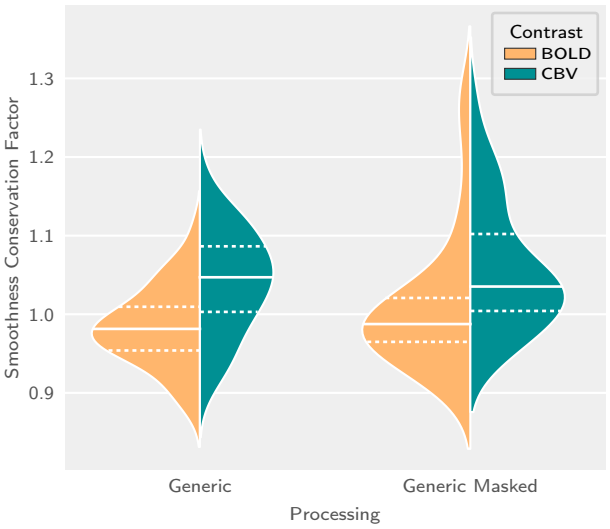
[14] Callbacks - Keras Documentation, . URL `https://keras.io/callbacks/`.

[15] Neuroimaging in Python — NiBabel 2.5.0 documentation, . URL `https://nipy.org/nibabel/`.

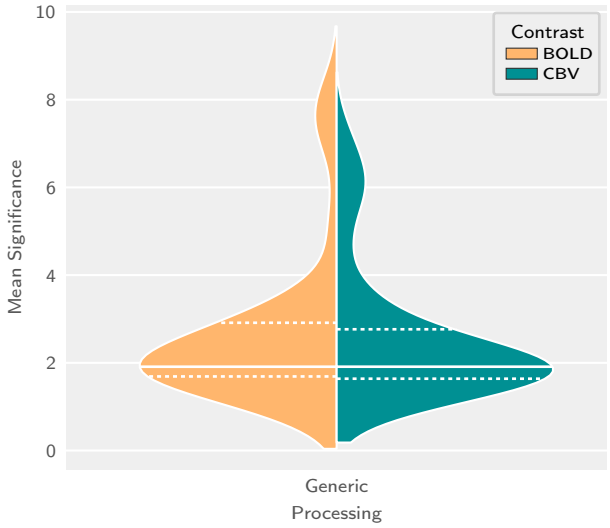
[16] scikit-learn/scikit-learn, January 2020. URL `https://github.com/scikit-learn/scikit-learn`. original-date: 2010-08-17T09:43:38Z.



(a) Comparison across workflows and functional contrasts, considering only matching template-workflow combinations.



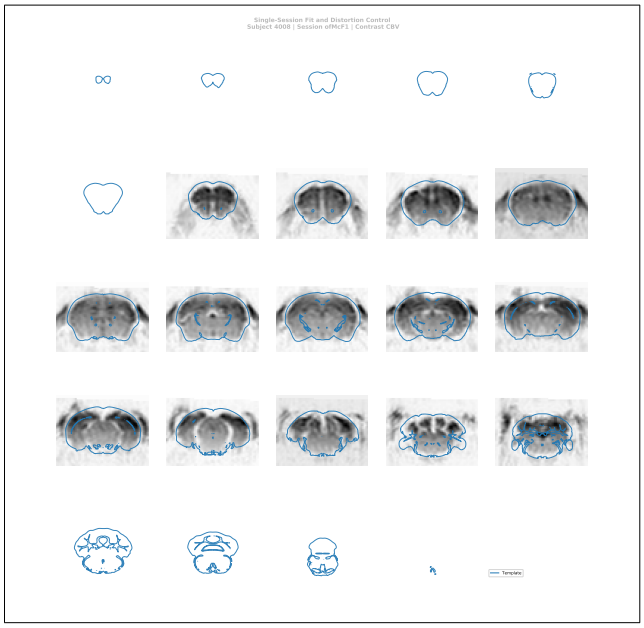
(b) Comparison across workflows and functional contrasts, considering only matching template-workflow combinations.



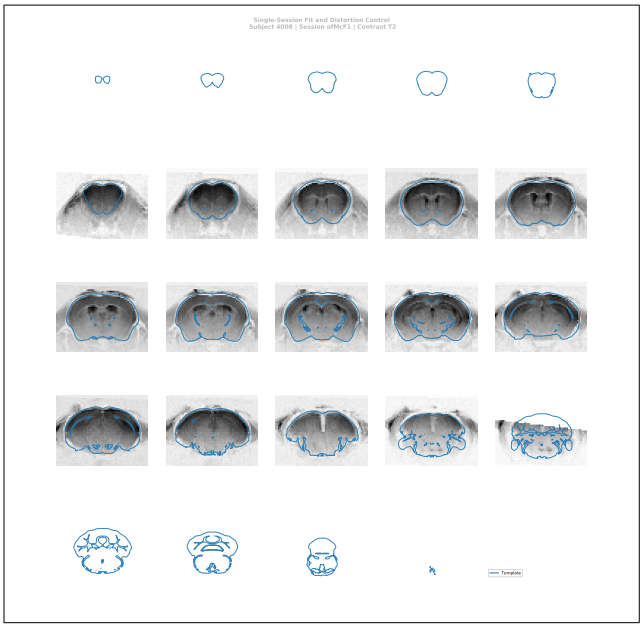
(c) Comparison across workflows and functional contrasts, considering only matching template-workflow combinations.

Figure 3: The SAMRI Generic workflow and template optimally and reliably conserve volume and smoothness — unlike the Legacy workflow and template. Plots of three target metrics, with coloured patch widths estimating distribution density, solid lines indicating the sample mean, and dashed lines indicate the inner quartiles.

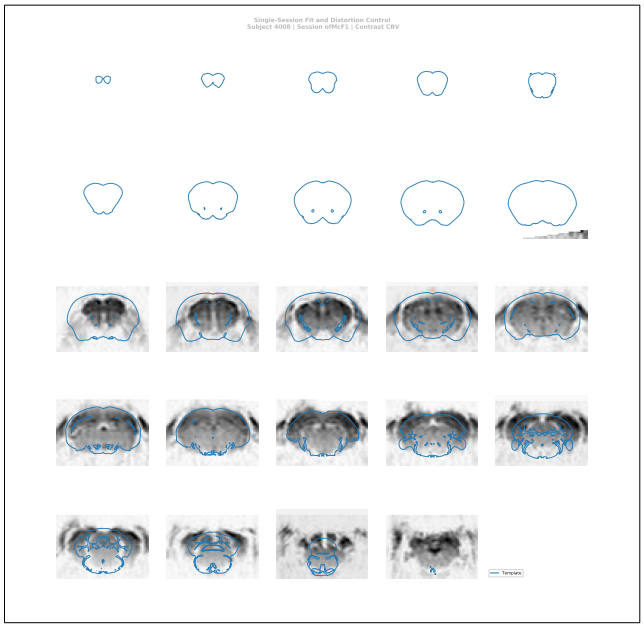
Supplementary Materials



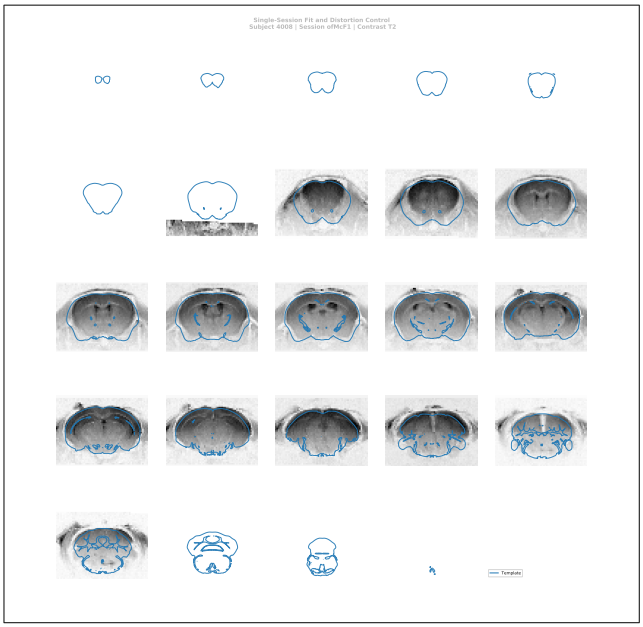
(a) SAMRI Generic workflow with Generic template, note the undistorted mapping and conservative smoothing.



(b) SAMRI Generic workflow with Generic template, inspecting the structural scan intermediary; note the undistorted mapping and conservative smoothing.



(c) SAMRI Generic Masked workflow, note the strong smoothing and the mapping distortion in the rostral and caudal areas.



(d) SAMRI Generic Masked workflow, note the strong smoothing and the mapping distortion in the rostral and caudal areas.

Figure S1: The SAMRI Generic workflow induces less smoothness, and provides more accurate coverage. Depicted are automatically created operator overview graphics, which allow a slice-by-slice (spacing analogous to acquisition) inspection of the registration fit. This representation affords a particularly detailed view of the preprocessed MRI data, and highly accurate template contours.

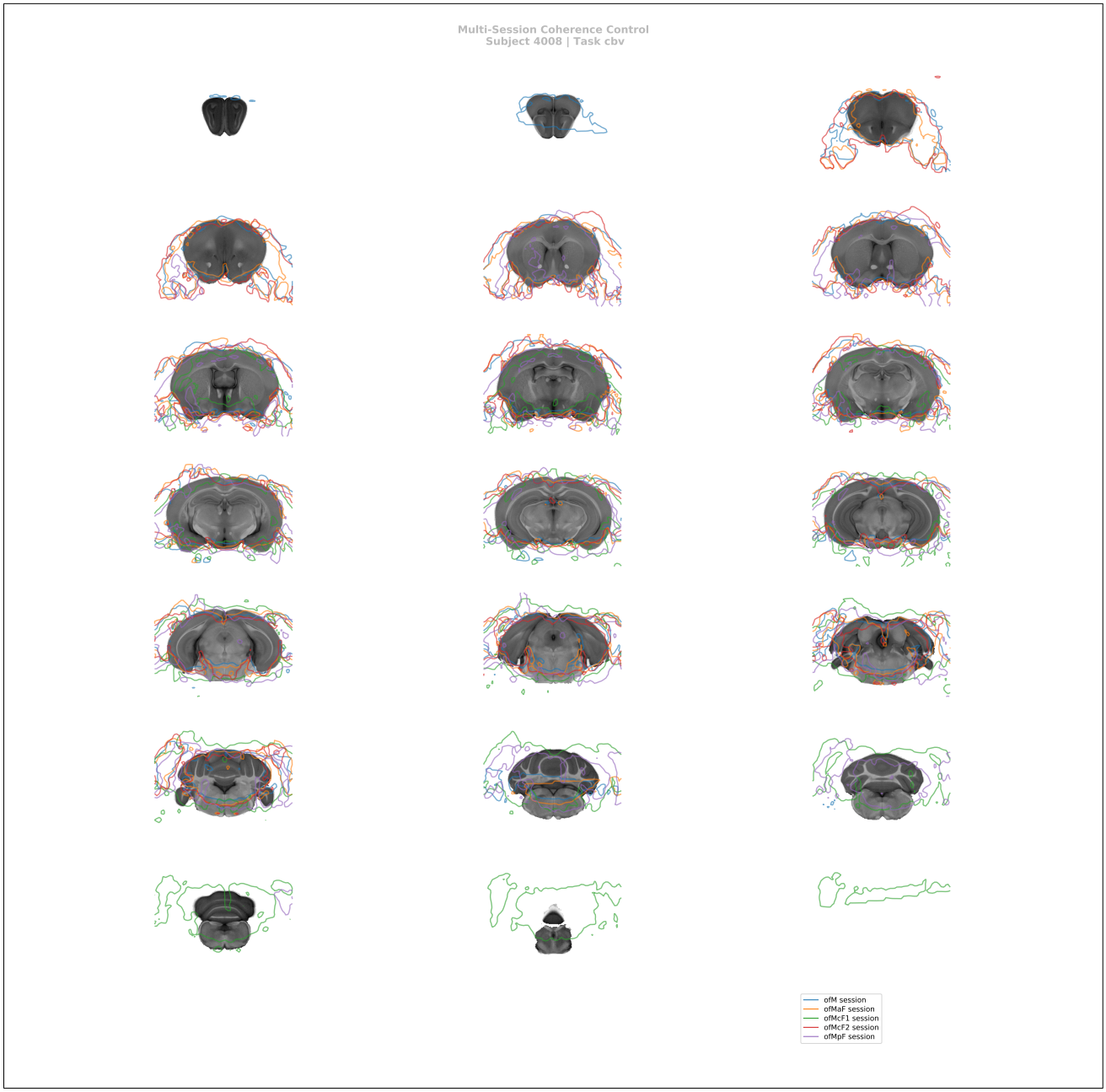


Figure S2: The SAMRI Generic workflow consistently maps high-salience features such as the implant site across sessions. Automatically created operator overview graphic, allowing a slice-by-slice (spacing analogous to acquisition) inspection of registration coherence. This representation permits a coarse assessment of registration consistency for multiple sessions — though at the cost of some clarity. Particularly, this visualization, allows an operator to track the position of high-amplitude fixed features across scans in order to ascertain coherence (similarly to what is automatically assessed by the Variance analysis of the session factor).

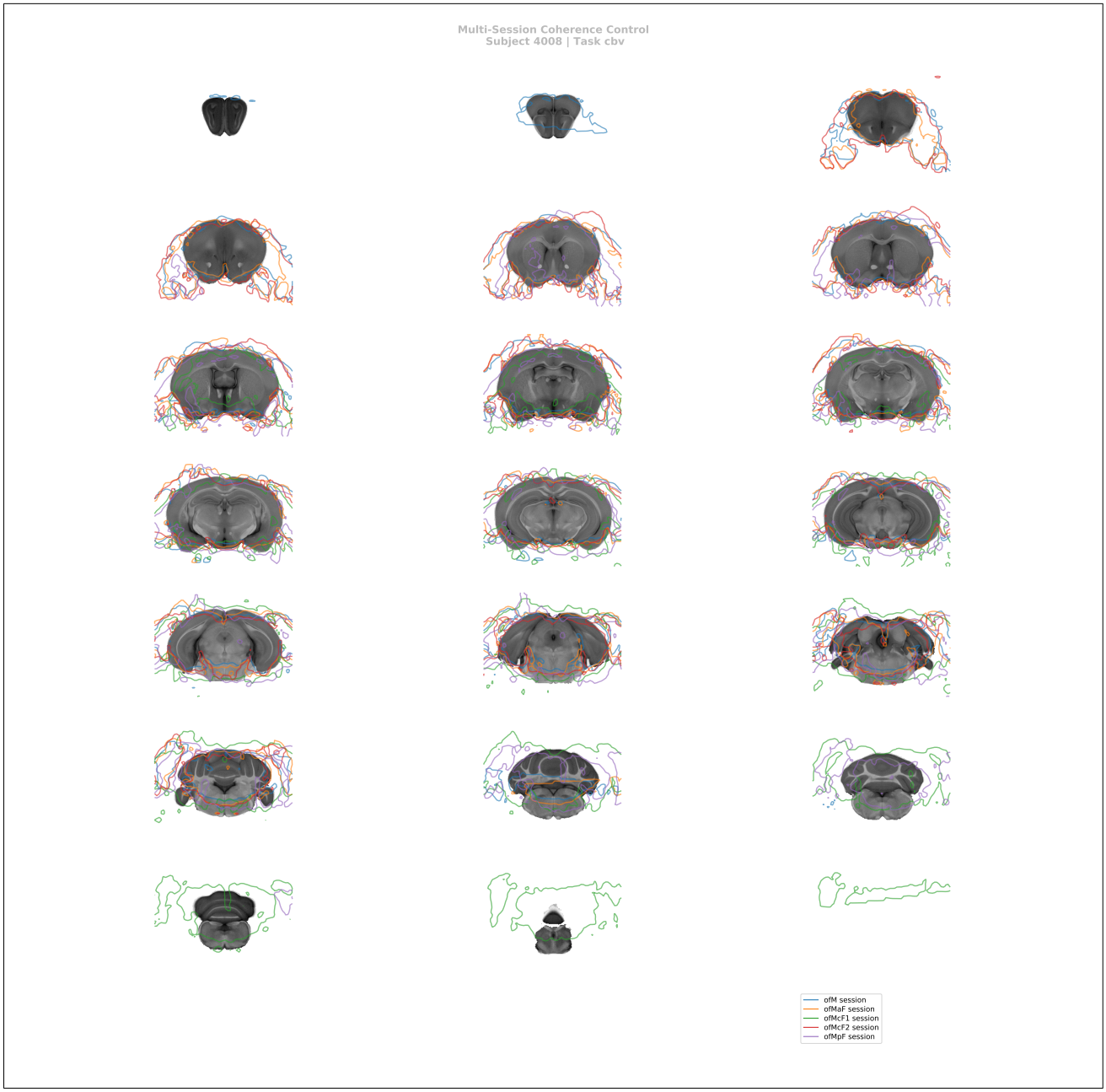
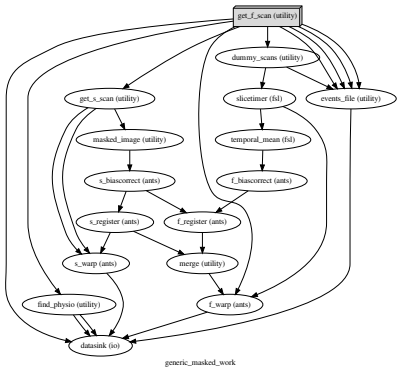
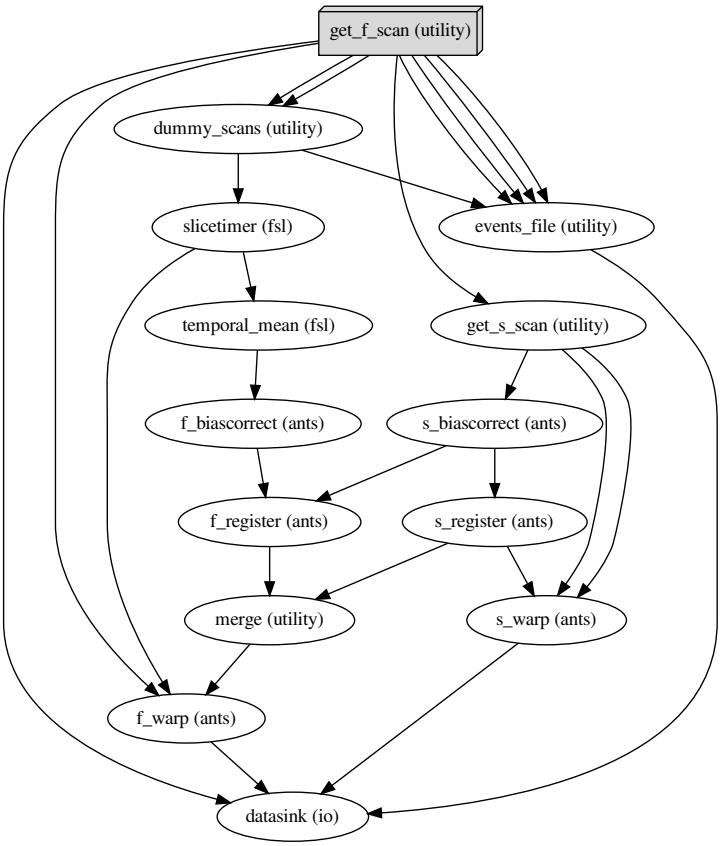


Figure S3: The SAMRI Generic Masked workflow consistently maps high-salience features such as the implant site across sessions. Automatically created operator overview graphic, allowing a slice-by-slice (spacing analogous to acquisition) inspection of registration coherence. This representation permits a coarse assessment of registration consistency for multiple sessions — though at the cost of some clarity. Particularly, this visualization, allows an operator to track the position of high-amplitude fixed features across scans in order to ascertain coherence (similarly to what is automatically assessed by the Variance analysis of the session factor).



(a) "SAMRI Generic Masked" workflow, which is based on the `antsIntroduction.sh` function (and other functions with hard-coded parameters optimized for human brain registration), and also performs destructive affine manipulations.



(b) "SAMRI Generic" workflow, based on the `antsRegistration` function. The pipeline uses a higher-resolution structural scan intermediary for registration (note the two processing streams), which facilitates differential handling of anatomical variation and susceptibility artefacts. The function used is highly parameterized, and both of its instances — "s_register" and "f_register" — are supplied in the workflow with defaults optimized for mouse brain registration.

Figure S4: Directed acyclic graphs detailing the precise node names (as seen in the SAMRI source code) for the two alternate MRI registration workflows. The package correspondence of each processing node is appended in parentheses to the node name. The "utility" indication corresponds to nodes based on Python functions specific to the workflow, distributed alongside it, and dynamically wrapped via Nipype. The "extra_interfaces" indication corresponds to nodes using explicitly defined Nipype-style interfaces, which are specific to the workflow and distributed alongside it.