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 as an additional step in the SAMRI registration workflow, a deep learning enabled framework for segmentation of brain tissue in functional and structural MR images.

Background

To make meaningful comparisons across the subjects of 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 original MR acquired images. To solve this issue, the images need to be projected into the reference frame via registration [1, 2]. As reported by Ioanas et al. [3], the general approach for mouse-brain image registration is to use high-level functions designed and optimized for human brain images. This requires the mouse-data to be adapted to the processing function instead of vice-versa. To provide contrast, they compare two workflows, a Legacy workflow that adapts the data to the processing functions and a Generic workflow, which is optimized to the data. While the Legacy workflow expands voxel size and deletes orientation information of the affine matrix to fit human brain data, the Generic workflow uses functions provided by the ANTs package [4], with spatial parameters adapted to the mouse brain. A quality control shows that the Generic workflow improves volume conservation, smoothness conservation and provides a reduction in variance. While the performance increase is considerable, registration quality can be improved further by computing the transformation solely on the brain volume to reduce disturbances induced by intensity variations outside the brain region.

The Improved Workflow

Intensities outside the brain region of a mouse MR image present high variations and bias the registration process. This can lead to stretching or skewing of the brain during the registration process to fit unwanted intensities inside the template brain region. As a remedy, we extract the brain volume from the images and compute the registration transformation on an image specific region of interest. For this purpose, we propose a machine learning enabled brain extraction in an additional node to the Generic workflow presented by Ioanas et al. in [3]. It creates a mask of the brain region using a classifier, which is then used to extract the region of interest. Two classifiers were trained, one for T2 contrast images and one for BOLD [5] and CBV [6] contrasts. The brain extraction nodes of the workflow return both the masked input and the binary mask. The latter is used to concentrate computing effort on a region of interest in preprocessing functions while the extracted brain volume is used to compute the registration transformation.

The Classifier

Classifiying each pixel with "brain" and "not brain" for the brain region extraction is done via a deep learning method. 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 [7, 8]. Training a neural network into a classifier is a supervised method. This means that the model needs to learn its parameters based on observations of data.

The training data set of a neural network 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, which 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 space defined by the Toronto Hospital for Sick Children Mouse Imaging Center [9]. A mask, defined in the same reference space is used as ground truth.

The registration process is good but not perfect, which is why the mask does not always align perfectly with the brain region of every slice. While our purpose was to create a workflow that generates better masks than the one from the template space, we show that the latter can be used as training data for the deep-learning model, by applying small changes to it.

Evaluation

For the quality control of the workflow, we first evaluate the segmentation process, followed by a benchmark between the Generic and the improved workflow. 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 [3] a major challenge of registration quality control (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 use these metrics to benchmark our workflow against theirs. It is worth noting that the quality of the Generic workflow is already remarkable. Even slight improvements should be considered an amelioration.

Segmentation

Quality control of our classifier is difficult in the sense that it should predict a better mask than the template. Nevertheless, it is useful 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 average Dice score taken on every slice of the test data set is $D_{coef} = 0.952 \sim 1$. The prediction thus have only minor changes in comparison with the template.

As an evaluation of the registration, we make use of the quality control from [3]. We denote the original workflow as Generic and our modified version as Generic*.

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.

A Wilcoxon signed-rank test on the absolute distance to 1 of both VCF distributions shows that the VCF is sensitive to the workflow (rank = 844.0, p = 0.044).

Moreover, the root mean squared error ratio favours the Generic* workflow ($\mathrm{RMSE_{G*}}/\mathrm{RMSE_G} \simeq 0.61$).

Further, we note that there is a significant variance decrease in all conditions for the Generic* workflow (0.25-fold).

With respect to the data break-up by contrast (CBV versus BOLD, fig. 2a), we see no notable main effect for the contrast variable (VCF of −0.03, 95%CI: −0.06 to −0.01), nor do we report a notable effect for the contrast-workflow interaction (VCF of 0.01, 95%CI: −0.02 to 0.04).

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 [10]. We employ a Smoothness Conservation Factor (SCF), expressing the ratio between the smoothness of the preprocessed images and the smoothness of the original images.

While the performance of the Generic SAMRI workflow is only slightly different from that of the Generic* workflow, the root mean squared error ratio favours the Generic* workflow ($\mathrm{RMSE_{G*}}/\mathrm{RMSE_G} \simeq 0.85$).

Descriptively, we observe that neither the Generic nor the Generic* workflow introduce a strong smoothing (SCF of 0.00, 95%CI: −0.02 to 0.01).

Further, we note that there is a notable variance decrease for the Generic* workflow (0.73 -fold).

Given the break-up by contrast shown in fig. 2b, we see no effect for the contrast variable (SCF of 0.06, 95%CI: 0.03 to 0.08) and the contrast-workflow interaction (SCF of 0.00, 95%CI: −0.02 to 0.02).

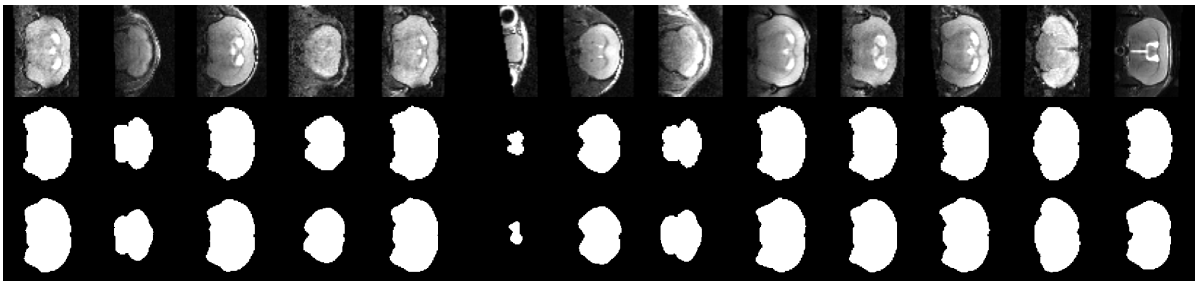
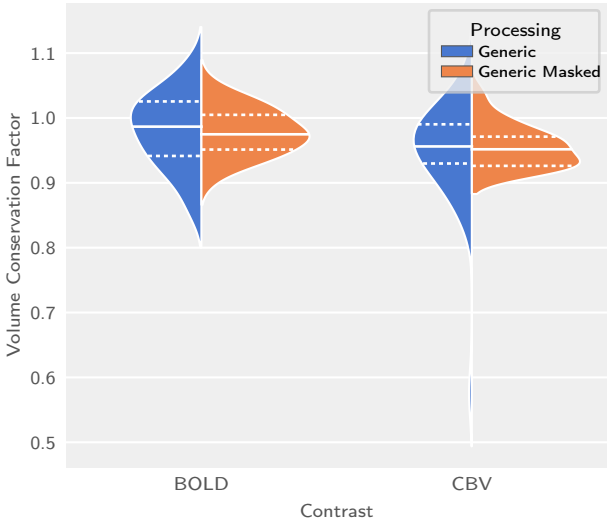
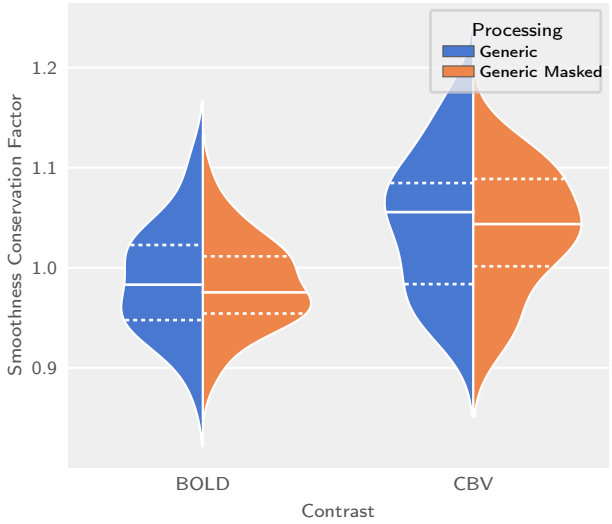


Figure 1: The Classifier predicts a similar mask to the ground truth. Random plots from the Test set illustrate the predictions of the classifier. The first row presents the input image, the second the ground truth and the third row shows the predictions of the classifier.



(a) Comparison across workflows and functional contrasts.



(b) Comparison across workflows and functional contrasts.

Figure 2: Both the SAMRI Generic and the Generic* workflow optimally and reliably conserve volume and smoothness, the latter showing values that are closely distributed to 1. 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.

Functional Analysis

Functional analysis is a frequently used avenue for preprocessing QC. Its viability derives from the fact that the metric being maximized in the registration process is not the same output metric as that used for QC. The method is however primarily suited to examine workflow effects in light of higher-level applications, and less suited for wide-spread QC (as it is computationally intensive and only applicable to stimulus-evoked functional data). For this analysis we compute the Mean Significance (MS), expressing the significance detected across all voxels of a scan.

We observe that the Generic* level of the workflow variable does not introduce a notable significance loss (MS of −0.03, 95%CI: −0.05 to 0.00). Furthermore, we note a slight variance decrease in all conditions for the Generic* workflow (0.94-fold).

With respect to the data break-up by contrast (fig. S1), we see no notable main effect for the contrast variable (MS of −0.07, 95%CI: −0.85 to 0.71).

Variance Analysis

An additional way to assess preprocessing quality focuses on the robustness to variability across repeated measurements, and whether this is attained without overfitting (i.e. compromising physiologically meaningful variability). The core assumption of this analysis of variance is that adult mouse brains in the absence of intervention retain size, shape, and implant position throughout the 8 week study period. Consequently, when examining similarity scores of preprocessed scans with respect to the target template, more variation should be found across levels of the subject variable rather than session variable. This comparison can be performed using a type 3 ANOVA, modelling both the subject and the session variables. For

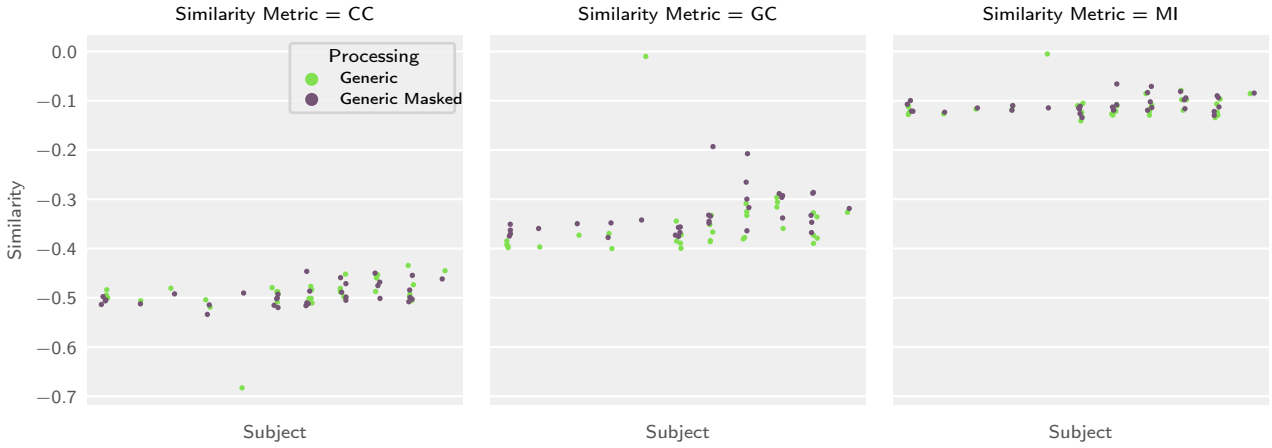


Figure 3: The SAMRI Generic* workflow conserves subject-wise variability and minimizes trial-to-trial variability. Swarmplots illustrate similarity metric scores of preprocessed images with respect to the corresponding workflow template, plotted across subjects (separated into x-axis bins) and sessions (individual points in each x-axis bin), for the CBV contrast.

this assessment we select three metrics with maximal sensitivity to different features: Neighborhood Cross Correlation (CC, sensitive to localized correlation), Global Correlation (GC, sensitive to whole-image correlation), and Mutual Information (MI, sensitive to whole-image information similarity).

Figure 3 renders the similarity metric scores for both the SAMRI Generic and Generic* workflows. Both, the Generic and the Generic* workflow produce results which show a higher F-statistic for the subject than for the session variable. For the Generic* workflow, F-statistics show: CC (subject: $F_{10,19} = 7.498$, $p = 9.63 \times 10^{-5}$, session: $F_{4,19} = 3.808$, $p = 0.019$), GC (subject: $F_{10,19} = 4.994$, $p = 0.0013$, session: $F_{4,19} = 1.586$, $p = 0.22$), and MI (subject: $F_{10,19} = 1.266$, $p = 0.31$, session: $F_{4,19} = 3.14$, $p = 0.039$).

For the Generic SAMRI workflow, resulting data F-statistics follow the same trend: CC (subject: $F_{10,19} = 6.702$, $p = 0.00021$, session: $F_{4,19} = 2.753$, $p = 0.058$), GC (subject: $F_{10,19} = 2.316$, $p = 0.055$, session: $F_{4,19} = 1.889$, $p = 0.15$), and MI (subject: $F_{10,19} = 2.529$, $p = 0.039$, session: $F_{4,19} = 1.67$, $p = 0.2$).

Discussion

The classifier improves the volume conservation, smoothness conservation, and session-to-session consistency of the SAMRI Generic workflow in terms of accuracy and precision.

Region assignment validity is also revealed in a qualitative examination of higher-level functional maps (fig. S2), where both the Generic and the Generic* workflow provide accurate coverage of the sampled volume for both BOLD and CBV fMRI data.

These benefits of the classifier are robust to the functional contrast (figs. 2a and 2b), with the Generic* workflow being less or equally susceptible to the contrast variable, when compared to the Generic workflow.

The classifier improves the workflow, while maintaining transparency and parameterization as well as the veracity of resulting data headers. The complete workflow of this report is fully reproducible and thus easily falsifiable. We make public the functions used for the masking in the workflow as well as those used to train the classifier, through the *mlebe* python package.

Our workflow has the advantage that the performance of a Neural Network can increase when trained further with new data. A user could easily recreate the steps described herein. Registering new data with the Generic workflow can increase the size of the training data set of the classifier. After removing bad registrations, the latter can be trained again, which will improve its generalisation capability. It is worth noting that classifiers can be shared between users to maximize efficiency. Another advantage of the trainability of the classifier and the openly published code is that this workflow can be adapted to a wast variety of data types.

Quality Control

We recommend reusing the presented data for workflow benchmarking, as they include (a) multiple sources of variation (contrast, session, subjects), (b) functional activity with broad coverage but spatially distinct features, and (c) significant distortions due to implant properties — which are appropriate for testing workflow robustness. In addition to the workflow code [11, 12], we openly release the re-executable source code [13] for all statistics and figures in this document. It is thus not just the novel method, but also the benchmarking process which is fully transparent and reusable for further data.

Conclusion

We present a remodeled version of the SAMRI Genetic registration workflow, which offers several advantages. In depth multivariate comparison with the original revealed superior performance of the SAMRI Generic* workflow in terms of volume and smoothness conservation, as well as variance structure across subjects and sessions. The easily accessible, optimized registration parameters of the SAMRI Generic Workflow as well as the open source code to the classifier training functions make the pipeline transferable to any other imaging applications. The open source software choices in both the workflow and this article's source code empower users to better verify, understand, remix, and reuse our work.

Methods

For the benchmarking of the two workflows, the same methods that are described in the original paper have been applied in this work. A more detailed description can be found there.

Model

As the architecture of the classifier, the U-Net from Ronneberger et al [8] 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 [14], written in Keras. Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It allows for a easily readable code and thus makes the workflow easier to reproduce.

The implementation of the U-Net from zhixuhao has, in addition to the original architecture, two drop-out layers. 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 [15].

The model was trained using the Dice loss, which 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. This 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 [16].

Data Set

The data set consists of 3D MR images taken from an aggregation of three studies: irsabi [17], opfvta [18], drlfom [19] and other unpublished data, acquired with similar parameters.

The measured animals were fitted with an optic fiber implant ($l = 3.2\,mm$ $d = 400\,\mu m$) targeting the Dorsal Raphe (DR) nucleus in the brain stem. Using this dataset shows that the workflow is robust to these types of experiment setups.

Images from the irsabi study are only used for quality control of the registration and are thus unknown to the classifier. It is the the same dataset that was used to benchmark the Generic workflow in the original paper and thus allows for a better estimation of the general performance of our improved pipeline.

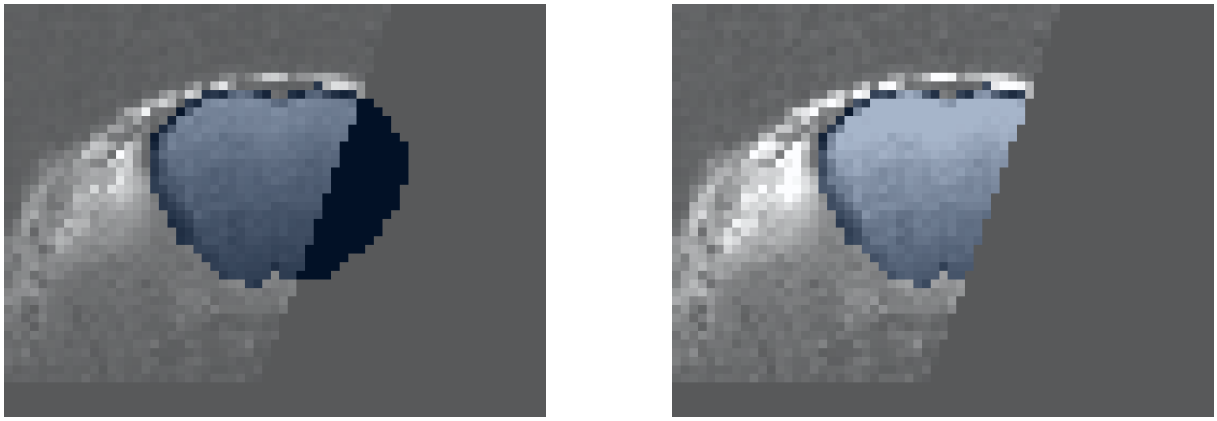
The images are transformed into a standard space using a template mask via SAMRI [20] and are thus defined in the same affine space. SAMRI is a data analysis package of the ETH/UZH Institute for Biomedical Engineering. It is equipped with an optimized registration workflow and standard geometric space 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. As a result, 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 zero-valued. 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 [21] is used. An example of the preprocessing can be seen in fig. 4.

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 (128, 128) by first zero-padding the smaller di-



(a) Example of an unpreprocessed slice.

(b) Example of a preprocessed slice.

Figure 4: 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.

mension to the same size as the bigger one and then reshaping the image into 128 using the function *cv2.resize* from the opencv python package [22].

Finally, the images are normalised by first clipping them from the minimum to the 99th percentile of the data 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 validation while 10% are used for testing. This is done with the help of the function *train_test_split* from the package sklearn.model_selection [23]. 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 0.5 pixels, zoom range of 0.3, brightness range of (0.7, 1.3) and horizontal as well as vertical flips. Additionally a gaussian noise with a variance range of (0, 0.001) is added to the image.

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 [24].

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.

To improve the learning process of the network, two callbacks from Keras were used [25]. "*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

To improve the SAMRI registration workflow, an additional node is implemented where the images are masked, such that only the brain region remains. To alleviate the task of the classifier, the image is first bias-corrected using the "*N4BiasFieldCorrection*" function of the ANTs package, with spatial parameters used in the samri functions. The image is then resampled into the resolution of the template space, which has a voxel size of $0.2 \times 0.2 \times 0.2$. This is done with the *Resample* command from the FSL library which is an analysis tool for FMRI, MRI and DTI brain imaging data [26]. Then, the image is preprocessed using the operations described in section 6.2. Since the classifier was trained to predict on images of shape (128, 128), the input needs to be reshaped. The

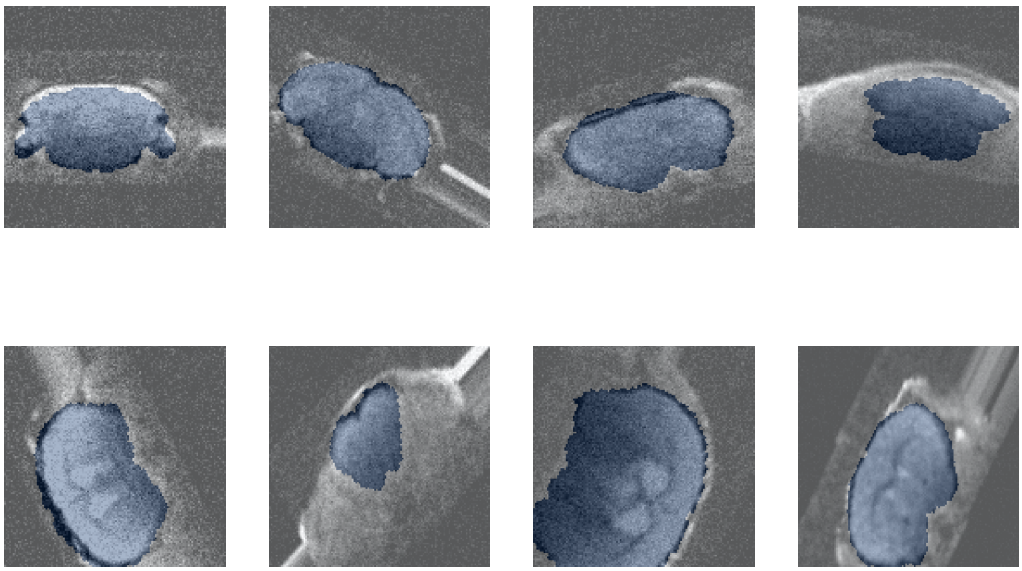


Figure 5: Augmented samples from the Training set.

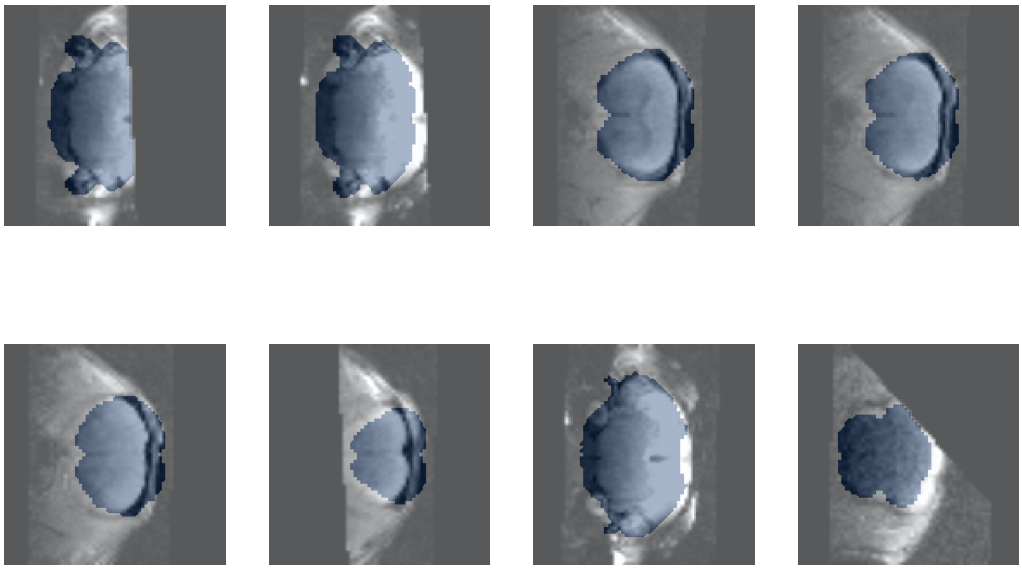


Figure 6: Slices where the mask includes too much outer-brain intensities are excluded from the data set. Examples from the slices that were excluded from the data set. The mask is shown in blue, on top of the brain image.

slice-wise predictions of the model are reconstructed to a 3D mask via the command *Nifit1Image* from the neuroimaging python package nibabel [27]. This is done using the same affine space as the input image. The latter is then reshaped into the original shape inverting the preprocessing step, either with the opencv resize method or by cropping. Additionally, the binary mask is resampled into its original affine space, before being multiplied with the brain image to extract the ROI. The workflow then continues with only the Region Of Interest as the image.

Metrics

For the current VCF implementation brain volume is defined as estimated by the 66^{th} voxel intensity percentile of the raw scan before any processing. The arbitrary unit equivalent of this percentile threshold is recorded for each scan and applied to all preprocessing workflow results for that particular scan, to obtain VCF estimates — eq. (2), where v is the voxel volume in the original space, v' the voxel volume in the transformed space, n the number of voxels in the original space, m the number of voxels in the transformed space, s a voxel value sampled from the vector S containing all values in the original data, and s' a voxel value sampled from the transformed data.

$$VCF = \frac{v' \sum_{i=1}^{m} [s'_i \geq P_{66}(S)]}{v \sum_{i=1}^{n} [s_i \geq P_{66}(S)]} = \frac{v' \sum_{i=1}^{m} [s'_i \geq P_{66}(S)]}{v \lceil 0.66n \rceil} \tag{2}$$

The SCF metric is based on the ratio of smoothness before and after processing. The smoothness measure is the full-width at half-maximum (FWHM) of the signal amplitude spatial autocorrelation function (ACF [28]). Since fMRI data usually do not have a Gaussian-shaped spatial ACF, we use AFNI [29] to fit the following function in order to compute the FWHM — eq. (3), where r is the distance of two amplitude distribution samples, a is the relative weight of the Gaussian term in the model, b is the width of the Gaussian and c the decay of the mono-exponential term [30].

$$ACF(r) = a * e^{-r^2/(2*b^2)} + (1-a) + e^{-r/c} \tag{3}$$

We examine statistical power as relevant for the MS metric via the negative logarithm of first-level p-value maps. This produces voxelwise statistical estimates for the probability that a time course could — by chance alone — be at least as well correlated with the stimulation regressor as the voxel time course measured. We compute the per-scan average of these values as seen in eq. (4), where n represents the number of statistical estimates in the scan, and p is a p-value.

$$MS = \frac{\sum_{i=1}^{n} -log(p_i)}{n} \tag{4}$$

Software

The workflow functions make use of the Nipype [31] package, which provides high-level workflow management and execution features. Via this package, functions provided by any other package can be encapsulated in a node (complete with error reporting and isolated re-execution support) and integrated into a directed workflow graph. Parallelization can also be managed via a number of execution plugins, allowing scalability. Most importantly, Nipype can generate graph descriptor language (DOT) summaries, as well as visual workflow representations suitable for operator inspection, graph theoretical analysis, and programmatic comparison between workflow variants.

Via Nipype, we utilize basic MRI preprocessing functions from the FSL package [26] and registration functions from the ANTs package [4]. The choice of the ANTs package (in addition to FSL, which also provides registration functions) owes to the package's functions being more highly parameterized. This feature allows us to avoid maladaptive optimization choices, and instead fine-tune the registration to the overarching characteristics of the brain type at hand. Additionally, we have implemented a number of functions in our workflow directly, e.g. to read BIDS [32] inputs, and perform dummy scans management.

For Quality Control we distribute as part of this publication additional workflows using the NumPy [33], SciPy [34], pandas [35], and matplotlib packages [36], as well as Seaborn [37] for plotting, and Statsmodels [38] for top-level statistics, using the HC3 heteroscedasticity consistent covariance matrix [39]. Specifically, distribution densities for plots are drawn using the Scott bandwidth density estimator [40].

Data and Code Availability

The data archive relevant for this article is freely available [**?**], and automatically accessible via the Gentoo Linux package manager. The code relevant for reproducing this document is also freely available [13], as are its dependencies, and most prominently, MLEBE [12] and SAMRI [12], the packages via which the herein described workflows are distributed.

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 Op-

timized 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] Brian B Avants, Nick Tustison, and Gang Song. Advanced normalization tools (ants). *Insight j*, 2 (365):1–35, 2009.

[5] Seiji Ogawa, Tso-Ming Lee, Asha S. Nayak, and Paul Glynn. Oxygenation-sensitive contrast in magnetic resonance image of rodent brain at high magnetic fields. *Magnetic Resonance in Medicine*, 14(1):68–78, April 1990. doi: 10.1002/mrm.1910140108. URL https://doi.org/10.1002/mrm.1910140108.

[6] John J.A. Marota, C. Ayata, Michael A. Moskowitz, Robert M. Weisskoff, Bruce R. Rosen, and Joseph B. Mandeville. Investigation of the early response to rat forepaw stimulation. *Magnetic Resonance in Medicine*, 41(2):247–252, February 1999. doi: 10.1002/(sici)1522-2594(199902)41:2<247::aid-mrm6>3.0.co;2-u. URL https://doi.org/10.1002/(sici)1522-2594(199902)41:2<247::aid-mrm6>3.0.co;2-u.

[7] 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.

[8] 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.

[9] A.E. Dorr, J.P. Lerch, S. Spring, N. Kabani, and R.M. Henkelman. High resolution three-dimensional brain atlas using an average magnetic resonance image of 40 adult c57bl/6j mice. *NeuroImage*, 42(1):60–69, August 2008. doi: 10.1016/j.neuroimage.2008.03.037. URL https://doi.org/10.1016/j.neuroimage.2008.03.037.

[10] Oscar Esteban, Christopher Markiewicz, Ross W Blair, Craig Moodie, Ayse Ilkay Isik, Asier Erramuzpe Aliaga, James Kent, Mathias Goncalves, Elizabeth DuPre, Madeleine Snyder, et al. FMRIPrep: a robust preprocessing pipeline for functional MRI. *Nature Methods*, page 111–116, December 2019. doi: 10.1038/s41592-018-0235-4. URL https://doi.org/10.1038/s41592-018-0235-4.

[11] Hendrik_Klug. Jimmy2027/MLEBE, . URL https://github.com/Jimmy2027/MLEBE. original-date: 2019-10-13T09:57:20Z.

[12] Horea-Ioan Ioanas, Markus Marks, Dominik Schmidt, Florian Aymanns, and Markus Rudin. SAMRI — Small Animal Magnetic Resonance Imaging, November 2017. URL https://doi.org/10.5281/zenodo.1044033.

[13] Hendrik_Klug. Jimmy2027/mlebe_repsep, . URL https://github.com/Jimmy2027/mlebe_RepSep. original-date: 2020-01-20T16:51:45Z.

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

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.

[16] 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.

[17] Horea-Ioan Ioanas and Markus Rudin. BIDS Data for "An Optimized Registration Workflow and Standard Geometric Space for Small Animal Brain Imaging", April 2019. URL https://doi.org/10.5281/zenodo.2651640.

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

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

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

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

[22] opencv-python: Wrapper package for OpenCV python bindings., . URL https://github.com/skvark/opencv-python.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] 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.

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

[26] Mark Jenkinson, Christian F Beckmann, Timothy EJ Behrens, Mark W Woolrich, and Stephen M Smith. Fsl. *Neuroimage*, 62(2):782–790, 2012.

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

[28] Anders Eklund, Thomas E Nichols, and Hans Knutsson. Cluster failure: why fMRI inferences for spatial extent have inflated false-positive rates. *Proceedings of the National Academy of Sciences*, page 201602413, June 2016. doi: 10.1073/pnas.1602413113. URL https://doi.org/10.1073/pnas.1602413113.

[29] Robert W Cox. AFNI: software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomedical research*, 29(3):162–173, June 1996. doi: 10.1006/cbmr.1996.0014. URL https://www.sciencedirect.com/science/article/pii/S0010480996900142.

[30] Robert W Cox, Gang Chen, Daniel R Glen, Richard C Reynolds, and Paul A Taylor. FMRI clustering in AFNI: false-positive rates redux. *Brain connectivity*, 7(3):152–171, April 2017. doi: 10.1089/brain.2016.0475. URL https://doi.org/10.1089/brain.2016.0475.

[31] Krzysztof Gorgolewski, Christopher D. Burns, Dav Madison, Cindeeand Clark, Yaroslav O. Halchenko, Michael L. Waskom, and Satrajit S. Ghosh. Nipype: A flexible, lightweight and extensible neuroimaging data processing framework in Python. *Front. Neuroinform.*, 5, 2011. ISSN 1662-5196. doi: 10.3389/fninf.2011.00013. URL http://dx.doi.org/10.3389/fninf.2011.00013.

[32] Krzysztof J Gorgolewski, Tibor Auer, Vince D Calhoun, R Cameron Craddock, Samir Das, Eugene P Duff, Guillaume Flandin, Satrajit S Ghosh, Tristan Glatard, Yaroslav O Halchenko, et al. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. *Scientific Data*, 3:160044, June 2016. doi: 10.1038/sdata.2016.44. URL https://doi.org/10.1038/sdata.2016.44.

[33] Travis E. Oliphant. *Guide to NumPy*. Provo, UT, March 2006. URL https://www.numpy.org/devdocs/contents.html.

[34] K. Jarrod Millman and Michael Aivazis. Python for scientists and engineers. *Computing in Science & Engineering*, 13(2):9–12, March 2011. doi: 10.1109/mcse.2011.36. URL https://doi.org/10.1109/mcse.2011.36.

[35] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, June 2010.

[36] John D. Hunter. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, 9(3):90–95, June 2007. doi: 10.1109/mcse.2007.55. URL https://doi.org/10.1109/mcse.2007.55.

[37] Michael Waskom, Olga Botvinnik, Drew O'Kane, Paul Hobson, Saulius Lukauskas, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Alistair Miles, Yoav Ram, Tal Yarkoni, Mike Lee Williams, Constantine Evans, Clark Fitzgerald, Brian, Chris Fonnesbeck, Antony Lee, and Adel Qalieh. Seaborn: v0.8.1, September 2017. URL https://doi.org/10.5281/zenodo.883859.

[38] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with Python. In *9th Python in Science Conference*, June 2010. URL http://conference.scipy.org/proceedings/scipy2010/pdfs/seabold.pdf.

[39] J Scott Long and Laurie H Ervin. Using heteroscedasticity consistent standard errors in the linear regression model. *The American Statistician*, 54(3):217–224, August 2000. doi: 10.2307/2685594. URL https://www.jstor.org/stable/2685594.

[40] David W. Scott. On optimal and data-based histograms. *Biometrika*, 66(3):605–610, December 1979. doi: 10.1093/biomet/66.3.605. URL https://doi.org/10.1093/biomet/66.3.605.

Supplementary Materials

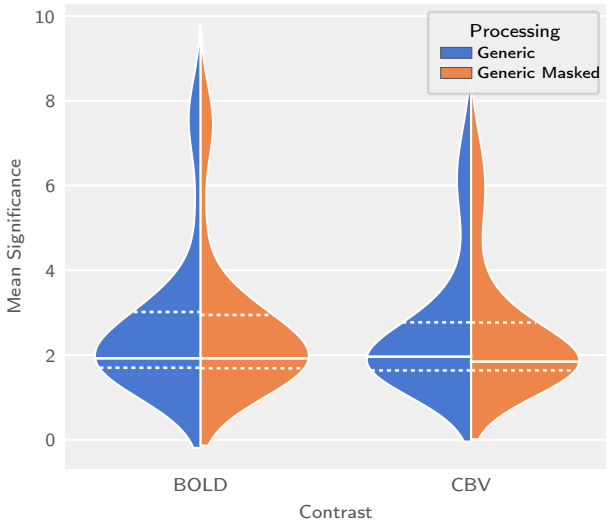
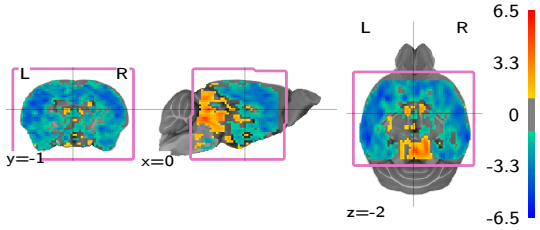
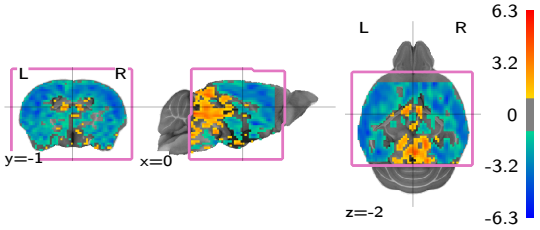


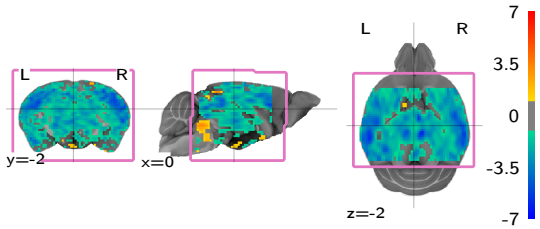
Figure S1: The Generic* workflow does not introduce a significance loss. Comparison across workflows and functional contrasts.



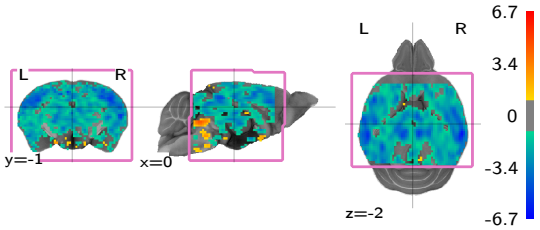
(a) Generic workflow with CBV map, showing correct slice orientation and coverage correctly bounded to the acquisition area.



(b) Generic* workflow with CBV map, showing correct slice orientation and coverage correctly bounded to the acquisition area.

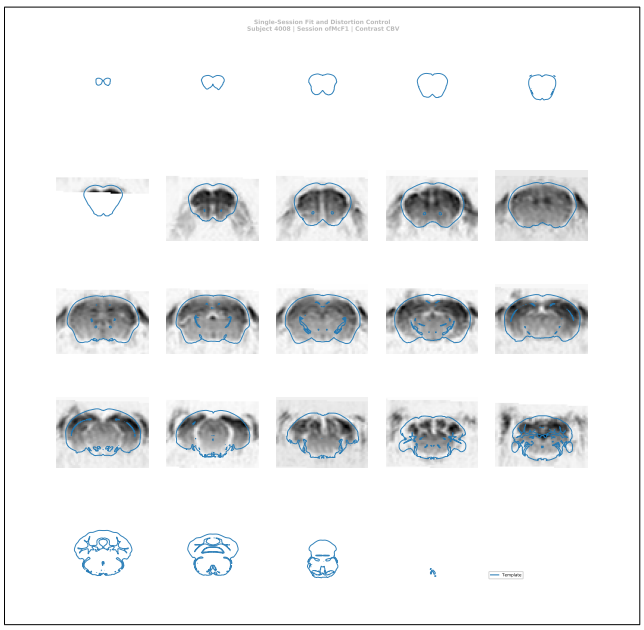


(c) Generic workflow with BOLD map, showing correct slice orientation and coverage correctly bounded to the acquisition area.

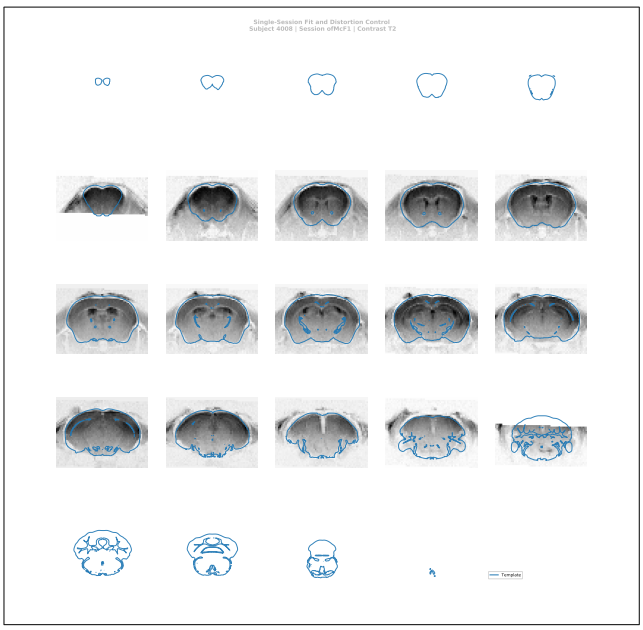


(d) Generic* workflow with BOLD map, showing correct slice orientation and coverage correctly bounded to the acquisition area.

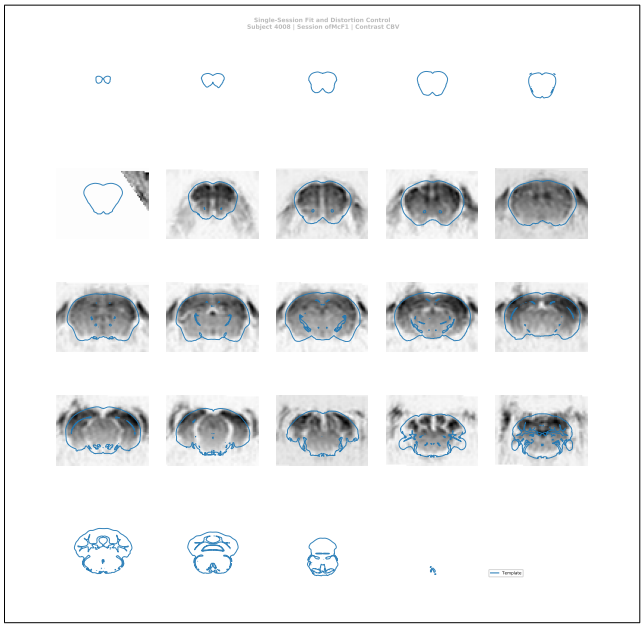
Figure S2: The Generic* workflow does not induce statistic coverage misalignment nor does it induce overflow of the statistic maps into adjacent anatomical regions. Illustrated are multiplanar depictions of second-level omnibus statistic maps separately evaluating CBV and BOLD scans, and thresholded at $|t| \geq 2$. The acquisition area is bracketed in pink, and in comparing it to statistic coverage it is important to note that the latter is always underestimated, as the omnibus statistic contrast is only defined for voxels captured in *every* evaluated scan.



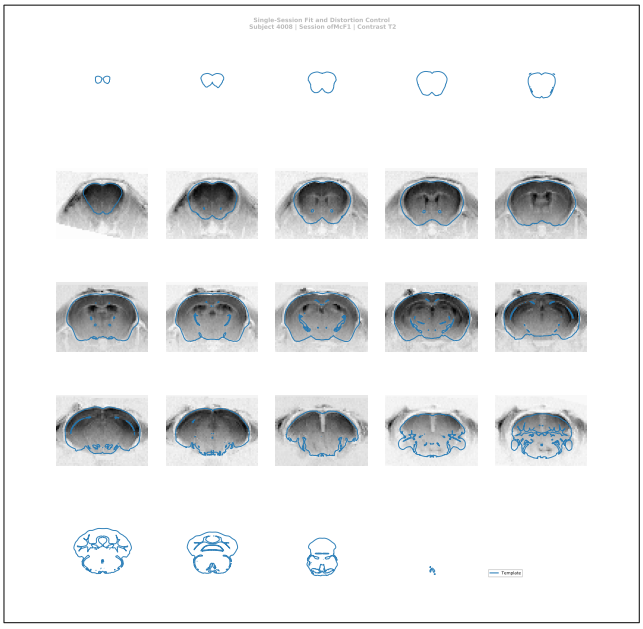
(a) SAMRI Generic workflow, note the undistorted mapping and conservative smoothing.



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



(c) SAMRI Generic Masked workflow, note the undistorted mapping and conservative smoothing.



(d) SAMRI Generic Masked workflow, inspecting the structural scan intermediary; note the undistorted mapping and conservative smoothing.

Figure S3: The SAMRI Generic* provides a more accurate coverage of the template space. 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.

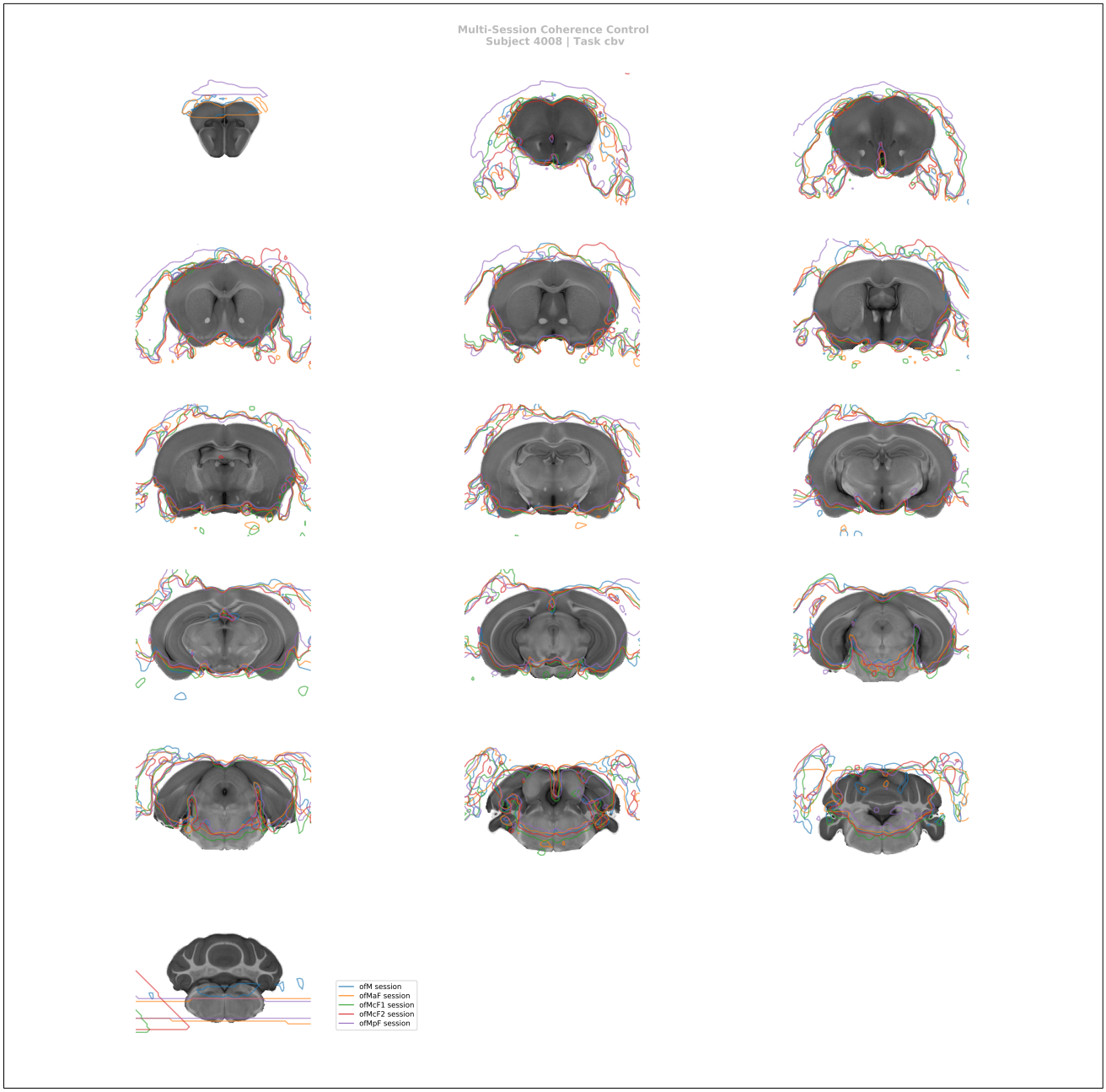


Figure S4: The SAMRI Generic workflow consistently maps high-salience features such as the implant site across sessions. Automatically created operator overview graphic, allowing a slice-by-slice (spacing analogous to acquisition) inspection of registration coherence. This representation permits a coarse 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).

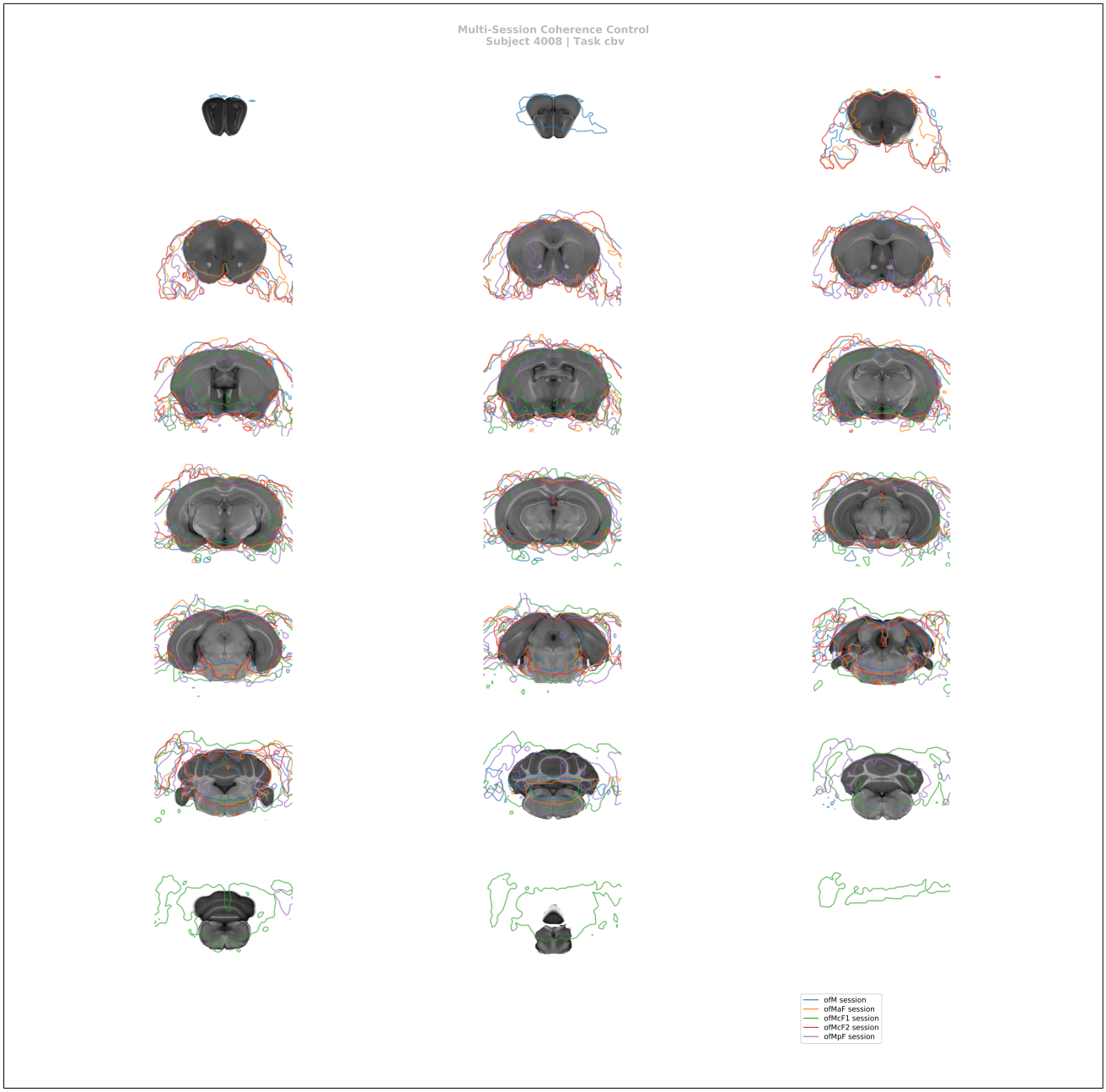
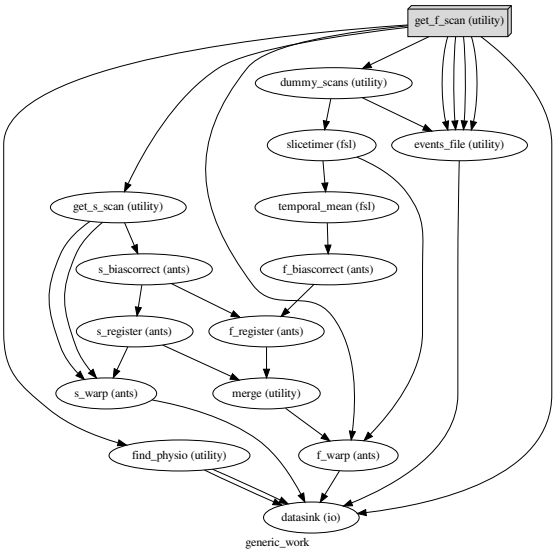
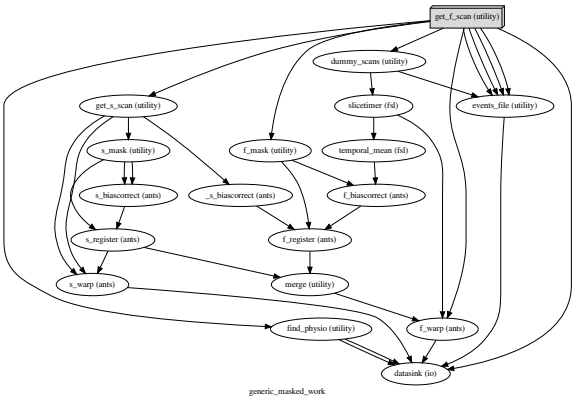


Figure S5: The SAMRI Generic Masked workflow consistently maps high-salience features such as the implant site across sessions. Automatically created operator overview graphic, allowing a slice-by-slice (spacing analogous to acquisition) inspection of registration coherence. This representation permits 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" workflow, based on the `antsRegistration` function.



(b) "SAMRI Generic Masked" workflow, which is based on the `antsRegistration` function. Two additional nodes provide the workflow with both the masked image and the binary mask.

Figure S6: 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.