Title of the Paper

Michael Rebsamen, Jan Riedo, Michael Mueller

***Abstract*—Machine learning (ML), white matter (WM), grey matter (GM), cerebrospinal fluid (CSF), background (BG)**

***Index Terms*—MRI, Segmentation, Machine Learning, DF, kNN, SVM**

I. Introduction

Segmentation of brain tissues from magnetic resonance images (MRI) has many clinical applications. Clinicians gain useful information from a separation of tissue into its three main anatomical types: white matter, grey matter, and cerebrospinal fluid. Voxel-based morphometric measures have been used to investigate brain disorders like Alzheimers disease [1], Parkinson's disease [2] or temporal lobe epilepsy [3]. However, manual segmentation of MRI is a labour-intensive task requiring expert skills. Fully automatic approaches for brain tissue segmentation are therefore a topic of active research. A good algorithm classifies the tissue types with high accuracy across a variety of images from different patients. Such a classification is a typical task for machine learning. These algorithms tend to perform well given enough training data during the learning phase. The availability of ground-truth data in sufficient quantity and quality for supervised learning is a particular challenge when working with medical images due to privacy concerns and the costs for manual segmentation. Optimization of the learning phase with a limited number of training data is therefore required.

FIXME: kNN is a popular classification method for MR data and has successfully been applied in MR brain segmentation [4]–[6]

FIXME: Base paper on df [7].

II. Methods

A. Dataset

All experiments were conducted on a subset of 100 unrelated subjects from a dataset provided by the *Human Connectome Project* [8]. From each individual, a total of eight 3-tesla head MRI are available: T1 and T2-weighted image volumes not skull-stripped (but defaced for anonymization) and skull-stripped with a bias field correction, and both modalities once in native T1 space and once in MNI-atlas space [9].

Ground-truth labels are automatically generated using *FreeSurf*, assigning each voxel either to background, white matter, grey matter, or cerebrospinal fluid. The dataset was split in a training set with 70 images and a test set with 30 images.

Biomedical Engineering, University of Bern
Authors e-Mail: michael.mueller4@students.unibe.ch, jan.riedo@students.unibe.ch, michael.rebsamen@students.unibe.ch

B. Pipeline

Training and testing data are loaded sequentially, each put through the pipeline consisting of: registration, pre-processing, feature extraction and ML training/classification. For testing, two additional steps, namely post-processing and evaluation are added.

Firstly, the data is loaded and registered to an atlas with a multi-modal rigid transformation using a regular step gradient descent optimizer. FIXME: In the preprocessing module skull stripping and bias field correction are applied in order to have images of the brain only, with less influence of the MRI scanning characteristics. Furthermore, a gradient anisotropic diffusion filter and z-score normalization is applied.

Preprocessed data is then fed into the feature extraction module, where seven features are computed. The feature matrix consists of three coordinate features, a T1 and a T2 intensity feature, and a T1 and T2 gradient feature. During feature extraction, a random mask is applied in order to randomly select a fraction of the voxels available. The mask is adjustable individually for BG, WM, GM, and CF. This is where the pathways of training and testing split up: training data is lastly fed to a certain supervised machine learning algorithm for training, whereas the testing data is classified with the previously created model. The classified testing data is then forwarded to a post-processing module where a dense conditional random field [10] is applied. Finally, the classification is evaluated based on a comparison with the ground truth, where a dice coefficient is computed (see chap. II-G).

FIXME: The medical image analysis pipeline consists of seven phases. In the pre-processing phase, an intensity normalization is performed on the images to have comparable grey scale ranges among all images in the subsequent steps. Similarly, the registration phase registers the images to an atlas space to have a common coordinate system. In the feature extraction phase, seven features are calculated for each voxel: three coordinates corresponding to the position of the voxel in the atlas space, and an intensity and a gradient on both the T1 and T2 modalities. A subset of voxels is choosen randomly to be used for training. During the learning phase, the selected algorithm is trained to classify voxels based on the given features. In the segmentation phase, the trained algorithm is used to classify all voxels in previously unseen test images. The post-processing phase aims to reduce noise by eliminating small isolated regions with a dense conditional random field [10]. Finally, the evaluation phase assesses the performance of the segmentation by comparing the result to ground-truth and calculating a dice coefficient for each class.

C. Training

TODO: Describe training of machine learning algorithms
TODO: **Short** intro to used algorithms?

D. Support Vector Machine (SVM)

Classification using Support Vector Machines (SVM) tries to find a hyperplane separating the data in a high-dimensional feature space. Given the feature vector x_i and the binary label y_i, the SVM solves the following optimization problem during training:

$$\begin{aligned} \min_{w,b,\xi} \ & \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i \\ \text{s.t. } & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \ i = 1, \ldots, m \\ & \xi_i \geq 0, \ i = 1, \ldots, m \end{aligned} \tag{1}$$

where $w \in \mathbb{R}^n$ is the normal vector and $b \in \mathbb{R}$ the offset of the separating hyperplane and $\phi(x_i)$ maps x_i into a higher-dimensional space.

The SVM implementation is based on libSVM [11]. Multi-class classification is solved with a *one-against-one* approach. To output probabilities, the predictions are calibrated using *Platt* scaling in which multiclass problems require an additional cross-validation which is an expensive operation for large datasets.

Given the relative low number of available features, we have chosen a radial basis function (RBF) kernel. A regularization term C and a model capacity γ needs to be chosen. These hyperparameters were determined with an exhaustive search and cross-validated on a subset of the training data, yielding $C = 500$ and $\gamma = 0.00005$.

E. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is widely used in machine learning for solving optimization problems iteratively. SGD has proven to be efficient for large-scale linear predictions [12].

In the current context, SGD learns a linear scoring function $f(x) = w^T x + b$ with model parameters w and b by minimizing the training error

$$\arg\min_{w,b} \frac{1}{m} \sum_{i=1}^{m} L(y_i, f(x_i)) + \alpha R(w) \tag{2}$$

where L is a loss function that measures miss-classifications, R is a regularization term penalising model complexity, and α is a non-negative hyperparameter.

In each iteration, a sample is randomly chosen from the training set and the model parameters are updated according

$$w \leftarrow w - \eta \left(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w} \right) \tag{3}$$

where η is the learning rate controlling the step size.

We use a smoothed hinge loss (*modified_huber*) for the loss function L, a l_2 penalty ($||w||_2$) for the regularization term R, and a gradually decaying learning rate. This makes SGD similar to a linear SVM. Again, the hyperparameters $\eta = 0.5$ and $\alpha = 0.01$ were determined with an exhaustive search and cross-validated on a subset of the training data.

F. k-Nearest Neighbors (kNN)

The k-Nearest Neighbors algorithm does not construct a general model, however stores instances of the training data. A query point is assigned to a certain class based on the votes, which come from the nearest neighbors of the point. A weight function assigns a weight which is inverse proportional to its distance to the point. The characteristics of kNN lead to a relatively low training time, and a rather high computation time for new points, depending on the amount of neighbors defined to vote. Despite the simple implementation of this algorithm is has been successful in various tasks such as biometric classification [13], and gained importance as recommendation algorithm [14]. A hyperparameter search was conducted for the k-value. The higher k, the better the overall dice with the drawback of a higher computation time. A value of $k = 20$ was found to be fast with only little trade-off in computation time.

G. Performance Evaluation

The Dice coefficient is a commonly used metric to compare the spatial overlap, ranging from 0 (no overlap) to 1 (perfect overlap). To evaluate the accuracy of the segmentation, a Dice coefficient is calculated between the prediction (E) and ground-truth (G) for each of the three labels.

$$D = \frac{2|E \bigcap G|}{|E| + |G|} = \frac{2TP}{2TP + FP + FN} \tag{4}$$

H. Infrastructure

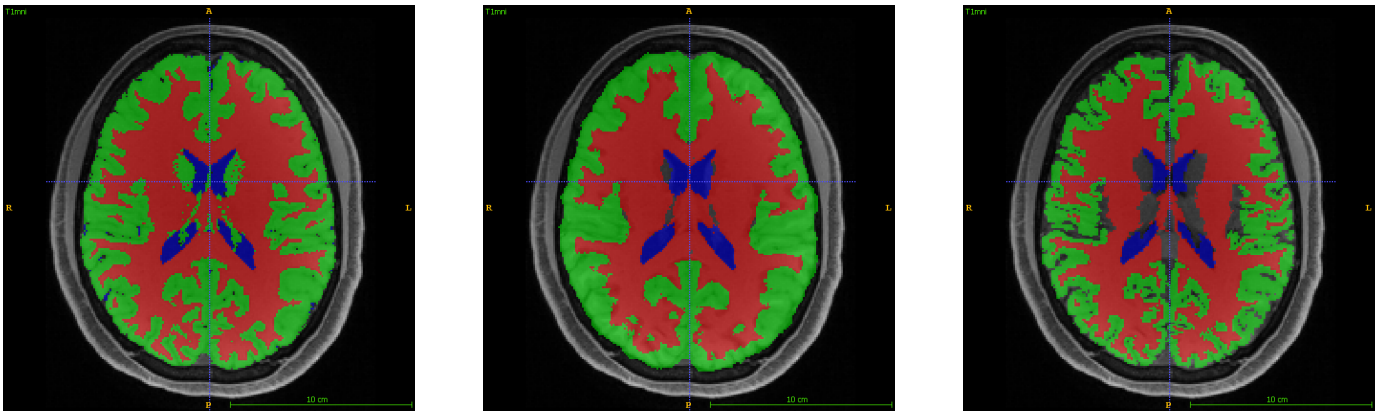
TODO: Describe UBELIX, libraries

III. Results

All algorithms tested and optimized were able to yield a good result for brain segmentation. The performance measured with the dice coefficient can be found in Tab. (I). Comparisons of computation time can be seen in Tab. (II).

A. Ground Truth Validity

The importance of looking at the images and not only at the numbers shall be presented based on one example MRI image, segmented with kNN. In Fig. (1) we see one slice of a brain, segmented in three different ways. On the left, we see a kNN segmentation based on non-coordinate features. The one in the middle is segmented with kNN based on all features. On the right we see the ground truth. Although the middle image shows a substantially better result in CSF dice than the left (0.68 vs. 0.76) we can barely see an improved CSF segmentation with bare eyes. What we see however, is a big difference on how detailed the white and grey matter are segmented, thus leading to a better overall segmentation. Another fact which shall be presented here is that the ground truth image is not a real ground truth. It is an image segmented by another algorithm. In Fig (1a) a better segmentation of the center part (white matter) is achieved compared to the ground truth image in Fig (1c) (background).



(a) (b) (c)

Fig. 1. (a) kNN segmented image based on non-coordinate features with dice values 0.86/0.82/0.68. (b) kNN segmented image based on all features with dice values 0.80/0.79/0.76. (c) Ground truth.

B. Feature Inspection

The following Fig (2) shows the scatter matrix of all the features. The whole dataset of 100 images was used for this evaluation. On the diagonal are the histograms to visualize the data distribution of each feature. The first three histograms are of the coordinate features and are normally distributed. The fifth and sixth histograms belong to the intensity features and follow a non normal distribution. The last two histograms are of the gradient features and are half normal distributed. The right upper part of the diagonal visualizes the linear correlation between each of the feature with the associated correlation coefficient. The left bottom part of the diagonal is redundant to the upper part.

The coordinate features with an correlation coefficient between -0.11 and 0.05 are independent and do not correlate with any other feature at all. The first intensity feature has a moderate linear relationship with the second intensity feature and a weak with the first gradient feature. The second intensity feature has also a moderate correlation with both of gradient features. The gradient features correlate with an correlation coefficient of 0.82 very strong among themselves.

TODO: Change label in ScatterMatrixPlot, Mike

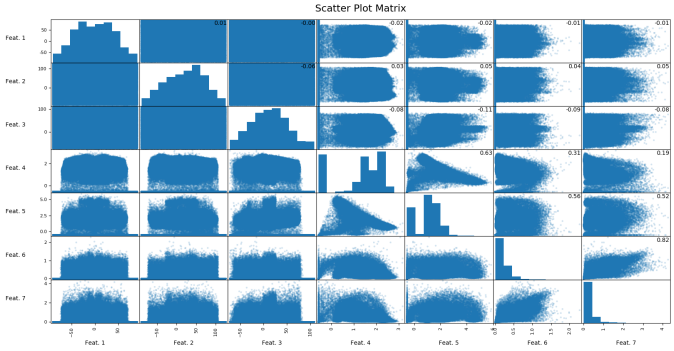


Fig. 2. Scatter plot of the features with correlation coefficient

The following Fig (3) visualizes the influence of each feature type and the used algorithms. The feature types are divided in three coordinate, two intensity and two gradient features. The first column in this figure is calculated with all the features combined and is considered as a reference. Each column belongs to a single feature type in which the dice of the four used algorithms is visualized. Vertically in line is each dice for the gray matter, white matter and the cerebrospinal fluid disposed.

TODO: Describe the plot, insert Legend of W, V or C? in figure 3, Mike

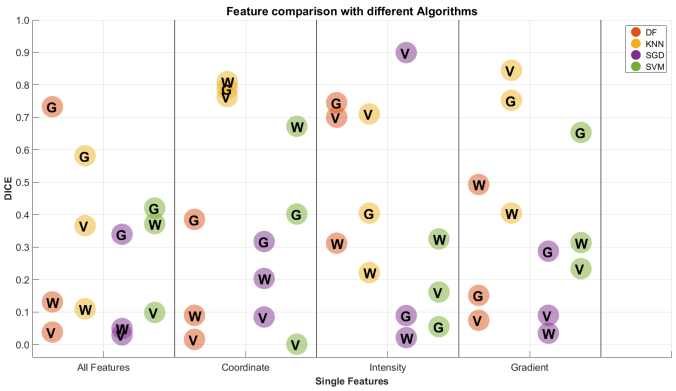
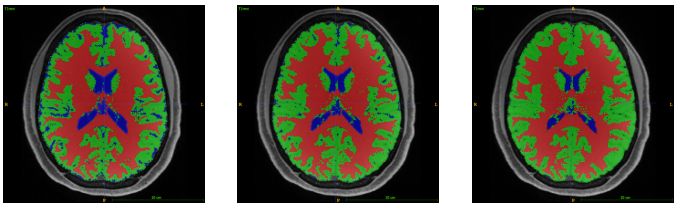


Fig. 3. Feature evaluation of single features with different algorithms, preprocessed.

C. Random Mask Optimization

One major task to handle was the low value for the CSF. Dice values above 0.5 were hard to achieve. One way to improve the dice for CSF was to optimize the random mask with respect to the fraction of CSF voxels taken into account. The effects of the random mask on the CSF dice can be seen in Fig. (4). Best results were achieved with a fraction of 0.004 CSF, approximately the same fraction as for white matter and grey matter. All following results are based on this optimized mask.



(a) (b) (c)

Fig. 4. Optimization of the random mask parameter for CSF. Fraction of CSF voxels taken into account f_v and dice value for this certain parameter d_v are: (f_v / d_v) (a) 0.4 / 0.22, (b) 0.04 / 0.44, and (c) 0.004 / 0.62.

D. Algorithm Performance

The decision forest algorithm was enhanced with normalized features, a higher number of CSF voxels in the training set and the optimization of the hyperparameters (see Fig. III-D). With this settings, the max dice coefficient was lifted from 0.703 to 0.754. This result was achieved with 80 trees and 3000 max nodes.

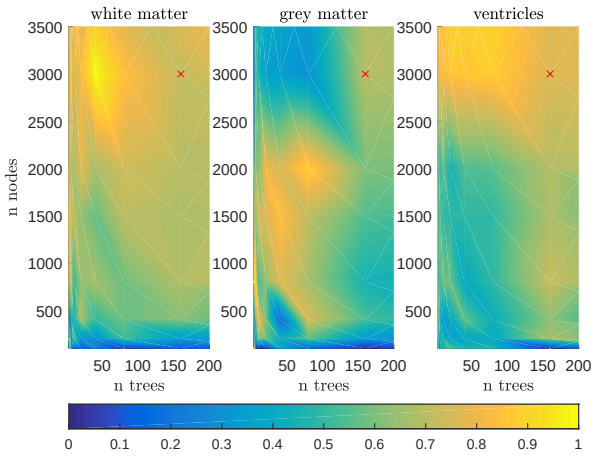


Fig. 5. DF plot of grid search for white matter, grey matter and cerebrospinal fluid. The red cross marks the chosen hyperparameters number of trees = 160 and maximum nodes per tree = 3000. Color does not represent dice, the data is stretched individually for all three plots.

Statistical distribution of the dice coefficients can be seen in Fig. III-D. DF and SVM achieve a similar mean dice score but SVM has a lower variance for CSF.

Comparison of computation time for training and testing is shown in Fig. III-D.

IV. Discussion

TODO: feature importance? which algorithm to choose for which use-case?

We have observed a rather small influence of the size of the training set, DF, SGD, and SVM reaching a similar dice coefficient with either 3 or 70 training samples.

SGD, DF, and kNN allow for an incremental training where input data can be processed sequentially in batches of arbitrary sizes. The SVM on the other hand requires all training data to be hold in memory, which inherently limits its

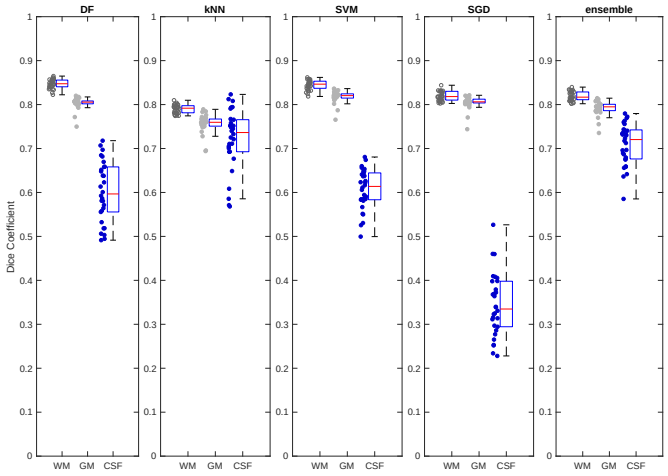


Fig. 6. Distribution of dice coefficients with optimal hyper-parameters for each algorithm on the full training set of 70 images.

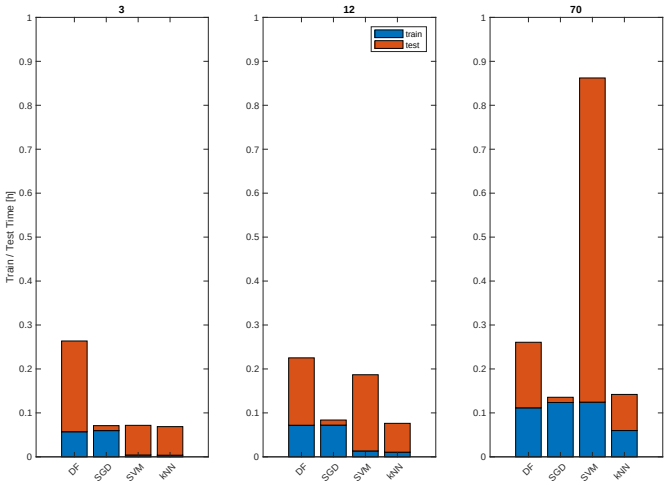


Fig. 7. Time for training and testing of the algorithms with training set sizes of 3, 12 and 70 samples. Test time is for one sample.

application. Large amounts of data are trained most efficiently with SGD, few data with SVM. This observation is consistent with the mathematical foundation of these two algorithms. The linear scaling behaviour of stochastic gradient descent is based on the principle of approximating a gradient by randomly (stochastically) choosing samples out of a large dataset and not necessary having to consult all data points in each iteration. A support vector machine might find a solution with a limited number of samples by mapping low dimensional features to a higher dimensional feature space where the data is more likely to be separable. Although this comes with high computational costs for fitting such a complex model to a larger amount of data.

V. Conclusion

The major challenge remains the quality of ground-truth data. As long as the test set is the output from an other (imperfect) algorithm, any approach is just an approximation of the other mechanism.

TABLE I
PERFORMANCE COMPARISON OF ML ALGORITHMS

Features	Size Dataset	DF	GMM	kNN	SGD	SVM	ensemble
All (f1-f7)	3	0.85/0.81/0.62	-	0.70/0.57/0.50	0.82/0.80/0.35	0.83/0.80/0.61	-
	12	0.85/0.81/0.59	-	0.75/0.66/0.67	0.82/0.80/0.34	0.84/0.81/0.61	-
	70	0.85/0.80/0.60	-	0.80/0.76/0.72	0.82/0.80/0.34	0.84/0.82/0.61	0.82/0.79/0.71
Coordinates only (f1-f3)	3	0.67/0.63/0.22	-	0.70/0.55/0.41	0.17/0.23/0.00	0.59/0.52/0.0	-
	12	0.67/0.64/0.11	-	0.74/0.63/0.56	0.19/0.22/0.00	0.59/0.57/0.0	-
	70	0.67/0.64/0.16	-	0.77/0.71/0.62	0.17/0.21/0.00	0.60/0.58/0.31	-
All non-coordinates (f4-f7)	3	0.84/0.80/0.50	-	0.85/0.80/0.45	0.82/0.80/0.34	0.84/0.79/0.0	-
	12	0.85/0.80/0.49	-	0.85/0.81/0.45	0.82/0.80/0.33	0.85/0.80/0.45	-
	70	0.85/0.80/0.48	-	0.85/0.81/0.54	0.82/0.80/0.34	0.85/0.80/0.44	-

Overview of achieved accuracy for the different algorithms. Mean dice scores for white matter/grey matter/cerebrospinal fluid. f1-f3: Coordinate features, f4: T1 intensity, f5: T1 gradient, f6: T2 intensity, f7: T2 gradient.

TABLE II
RUNTIME

Features	Size Dataset	DF	GMM	kNN	SGD	SVM
All (f1-f7)	3	205.4/22310.2	-	13.4/7023.7	214.9/1219.6	15.1/7289.7
	12	258.7/16563.6	-	38.1/7090.0	259.9/1250.0	48.2/18730.5
	70	401.4/16116.2	-	215.5/8873.5	445.3/1279.6	448.1/79668.4
Coordinates only (f1-f3)	3	-	-	10.4/4391.5	-	15.4/12178.7
	12	-	-	34.7/5449.3	-	62.2/43404.3
	70	-	-	196.4/6112.8	-	957.9/221440.5
All non-coordinates (f4-f7)	3	-	-	10.1/10084.7	-	12.4/6647.9
	12	-	-	34.6/18768.6	-	39.8/18691.1
	70	-	-	194.2/16555.7	-	323.2/80532.7

FIXME: Overview of the computation time in seconds for all algorithms (training time/testing time). Computation time includes pre- and post-processing.

In the current setup, the number of available features was limited to seven which is known to be on the lower bound for the examined algorithms. A deep learning approach that is directly processing the raw input data and implicitly learning how to extract features might be a better choice in this case. Whether such a neural network outperforms the classical machine learning algorithms remains to be investigated.

ACKNOWLEDGEMENT

Calculations were performed on UBELIX (http://www.id.unibe.ch/hpc), the HPC cluster at the University of Bern.

REFERENCES

[1] G. F. Busatto, G. E. Garrido, O. P. Almeida, C. C. Castro, C. H. Camargo, C. G. Cid, C. A. Buchpiguel, S. Furuie, and C. M. Bottino, "A voxel-based morphometry study of temporal lobe gray matter reductions in alzheimers disease," *Neurobiology of aging*, vol. 24, no. 2, pp. 221–231, 2003.

[2] S. Price, D. Paviour, R. Scahill, J. Stevens, M. Rossor, A. Lees, and N. Fox, "Voxel-based morphometry detects patterns of atrophy that help differentiate progressive supranuclear palsy and parkinson's disease," *Neuroimage*, vol. 23, no. 2, pp. 663–669, 2004.

[3] C. Rummel, N. Slavova, A. Seiler, E. Abela, M. Hauf, Y. Burren, C. Weisstanner, S. Vulliemoz, M. Seeck, K. Schindler *et al.*, "Personalized structural image analysis in patients with temporal lobe epilepsy," *Scientific reports*, vol. 7, no. 1, p. 10883, 2017.

[4] P. Anbeek, K. L. Vincken, M. J. van Osch, R. H. Bisschops, and J. van der Grond, "Probabilistic segmentation of white matter lesions in MR imaging," *NeuroImage*, vol. 21, no. 3, pp. 1037–1044, mar 2004.

[5] C. A. Cocosco, A. P. Zijdenbos, and A. C. Evans, "A fully automatic and robust brain MRI tissue classification method," *Medical Image Analysis*, vol. 7, no. 4, pp. 513–527, dec 2003.

[6] S. Warfield, M. Kaus, F. A. Jolesz, and R. Kikinis, "Adaptive, template moderated, spatially varying statistical classification," *Medical Image Analysis*, vol. 4, no. 1, pp. 43–55, mar 2000.

[7] L. Breiman, *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] D. C. Van Essen, S. M. Smith, D. M. Barch, T. E. Behrens, E. Yacoub, K. Ugurbil, W.-M. H. Consortium *et al.*, "The wu-minn human connectome project: an overview," *Neuroimage*, vol. 80, pp. 62–79, 2013.

[9] J. Mazziotta, A. Toga, A. Evans, P. Fox, J. Lancaster, K. Zilles, R. Woods, T. Paus, G. Simpson, B. Pike *et al.*, "A probabilistic atlas and reference system for the human brain: International consortium for brain mapping (icbm)," *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, vol. 356, no. 1412, pp. 1293–1322, 2001.

[10] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in neural information processing systems*, 2011, pp. 109–117.

[11] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[12] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 116.

[13] D. Dhriti and M. Kaur, "K-nearest neighbor classification approach for face and fingerprint at feature level fusion," *International Journal of Computer Applications*, vol. 60, no. 14, pp. 13–17, dec 2012.

[14] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the tenth*

international conference on World Wide Web. ACM Press, 2001, pp. 285–295.