Predicting Survival Time for Patients Diagnosed with Gliomas Using Compressed Representation of Brain MRI Scans

Anmol Sharma
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
anmol_sharma@sfu.ca

Bdour Abed
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
bdour_alzeer@sfu.ca

Baraa Orabi
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
borabi@sfu.ca

Ben Cardoen
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
bcardoen@sfu.ca

Rimika Chaudhury
Department of Computing Science
Simon Fraser University
Burnaby, BC V5A 1S6
rimika_chaudhury@sfu.ca

Abstract

In this paper we investigate the problem of predicting survival time (in days) for patients diagnosed with high grade gliomas (gliobastoma multiforme) using their brain MRI studies. To approach this problem, we first reduce the input features into a compressed representation, learned using an unsupervised 3D convolutional neural network based autoencoder which is trained to predict its own input. Once the most informative features are learned, they are further reduced in dimensionality using singular value decomposition. We then try a number of regression based machine learning methods on this reduced data. We observe the best mean squared error (MSE) of 125048 using K-nearest neighbours. Our observed results in terms of MSE are close to the state of the art methods as of writing this paper. However given the regressors' correlation results and domain knowledge, we conclude that using brain MRI data alone in this framework is insufficient to produce predictions with high correlation.

1 Introduction

Brain tumors have high mortality rates making them one of the deadliest cancers[1, 2]. With respect to their origin, brain tumors can be either primary (originating from the brain) or metastatic (originating from other sites). Gliomas (or gliobastoma multiforme) constitute 70% of malignant primary brain tumors in adults [2], and are usually classified as High Grade Gliomas (HGG) or Low Grade Gliomas (LGG). HGG encompasses grade III and IV of the WHO categorization [3], and exhibits a rapid proliferating behaviour, with a patient survival time of about one year [2].

Magnetic Resonance Imaging (MRI), Positron Emission Tomography (PET) and Computed Tomography (CT) are some of the standard radio imaging techniques used for diagnosing abnormalities in the brain. Due to its improved soft tissue contrast, as compared to plain radiography or CT [4], MRI has been extensively employed for diagnosing brain and nervous system abnormalities over the last few decades [4]. MR images are usually procured in multiple sequences or modalities, depending upon the different excitation and repetition times used during the scan. This enables MRI to capture distinct structures of interest, by producing noticeably different tissue contrasts [2]. These include sequences T_1-weighted, T_2-weighted, post-contrast T_1-weighted (T_{1Gd}), and T_2-weighed with fluid-attenuated inversion recovery (T_{2Flair}). The rationale behind using these four sequences lies in the fact that different tumor regions may be visible in different sequences, allowing for a more accurate marking of tumor region.

Recently, the problem of predicting patient survival time (in days) was introduced in the BRATS 2017 competition [5], as part of the MICCAI 2017 conference. The problem is significant since an accurate prediction can help in effective treatment, where a low survival period can trigger a more immediate response in terms of patient care and treatment.

Methods of machine learning (ML) have been applied to medical image analysis for decades, most prominently in developing computer aided detection/diagnosis systems for various imaging modalities. ML in the form of Deep Learning (DL) have also been adopted in the medical imaging domain, with studies targeting modalities such as mammography [6], CT [7] and MRI [8], for various different tasks such as image classification, and localization and segmentation of abnormalities. In this work, we apply a combination of DL and ML methods to predict overall survival time (in days) of patients using the patients' brain MRI scans.

2 Dataset

The dataset used in this project was part of the BRATS 2017 challenge [5]. The dataset contains multi-institutional, clinically-acquired, pre-operative multi-sequence MRI scans of glioblastoma multiforme (GBM/HGG) and lower grade glioma (LGG), with pathologically confirmed diagnosis and available overall survival time for most patients. The number of institutions contributing the data was 19. The scans were acquired in four sequences, which include T_1-weighted, T_2-weighted, post-contrast T1-weighted (T_{1Gd}), and T_2-weighed with fluid-attenuated inversion recovery (T_{2Flair}). The dataset contains 210 patients, out of whom 163 patients had available overall survival data in the form of a supplementary CSV file. The MRI scans in the dataset are pre-processed, i.e. co-registered to the same anatomical template, interpolated to the same resolution (1 mm^3), and skull-stripped. The spatial dimensions of a patient datapoint is 240x240x155x4 (height, width, number of slices, and number of sequences). To keep the computation burden and memory usage low, we resize each patient volume from 240x240x155 to 120x120x155 using cubic interpolation.

3 Approach

Brain MRI studies are enormously high dimensional 3D volumes. In order to effectively apply machine learning methods for our task of predicting survival days for the patients, the data must be compressed into a lower dimensional representation, ideally without losing too much information. One way to do this traditionally is to use hand crafted features like 3D LBP [9] or 3D Voxel HOG [10]. However hand crafted features may not always be the most ideal compact representation of the input data. Hence in this work we use an autoencoder based on 3D convolutional neural networks, trained in an unsupervised fashion to automatically learn the most informative features from the input data. A detailed discussion of this pipeline is followed in the Section 3.1. Once the network is trained, we split the network and extract the encoder part. The encoder part is then used as an extracted feature set for our patient data.

After autoencoder feature extraction, we use singular value decomposition (SVD) to reduce that dimensionality of the extracted features. This step is needed to enable us to feasibly explore a wide range of ML methods on the dataset. With this dimensionality-reduced feature set, we explore different regression models. And then finally see if any of these models' predictions correlate with the actual data using correlation values.

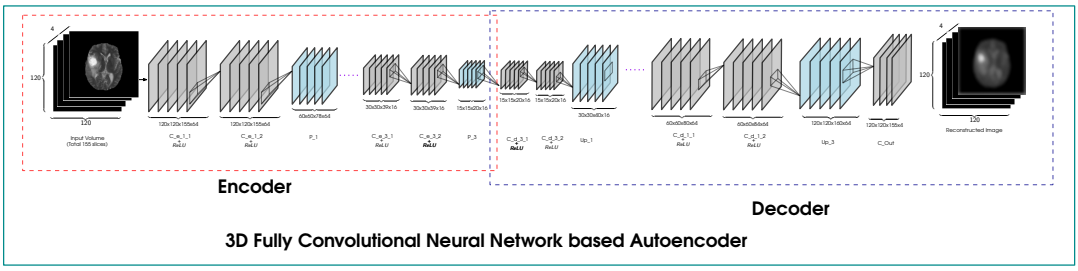


Figure 1: Proposed 3D Convolutional Neural Network based architecture for unsupervised feature learning from brain MRI data.

3.1 Feature extraction

3.1.1 Convolutional neural networks

Convolutional neural networks (ConvNets or CNNs) [11] are neural networks that are suitable for processing input of a grid-like topology, e.g, time series data and images. Compared to a traditional artificial neural network, a ConvNet uses convolution operation as compared to matrix multiplication in some or all of its layers. In this work we use a 3D ConvNet in the form of an autoencoder architecture for unsupervised feature extraction from Brain MRI studies. The architecture is illustrated in Figure 1.

3.1.2 CNN architecture

For the task of learning the most informative features from the input brain MRI data in an unsupervised fashion, we construct a 19-layer autoencoder based on 3D convolution and pooling layers. The network is fully convolutional, in the sense that it does not have any fully connected layers at the end, and hence is compatible with any input shape. The network takes in a full 4D volume of a patient's brain MRI study (width, height, number of slices, channels) as input, and tries to recreate or reconstruct its that input by learning kernels in the convolution layers. A usual autoencoder architecture has the first half of the network compressing the input to a smaller dimension, and the second part of the network using that compressed representation to reconstruct the original input. The network consists of six blocks of two convolution layers, where three blocks are in the encoder section and three in the decoder section. Each block is followed by a pooling layer in the encoder section or an upsample layer in the decoder section. The network relies entirely on the patient MRI study, and does not need any supervision in the form of labels (tumor location or segmentation), and hence is a fully unsupervised network. The hyperparameters used in the network are presented in the Table 1. A convolution type "same" implies that required zero padding is performed on the input to the layer to account for output downsampling, while a "valid" convolution implies no zero padding, leading to downsized output. The number of trainable parameters in the network is 1,472,964.

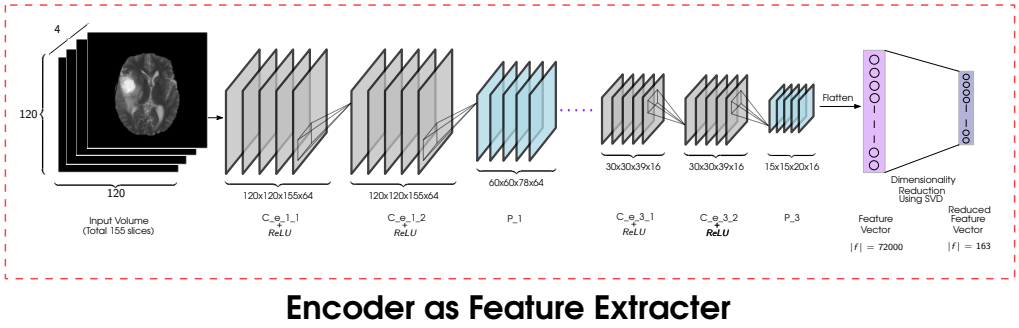


Figure 2: Illustration of the feature extraction pipeline. The trained encoder part is used as feature extractor, which is followed by dimensionality reduction using SVD.

Table 1: 3D fully convolutional neural network based autoencoder

Layer	Filter Size	Stride	Conv Type	Input	Output
C_e_1_1	64x5x5x5	1x1x1	same	120x120x155x4	120x120x155x64
C_e_1_2	64x5x5x5	1x1x1	same	120x120x155x64	120x120x155x64
P_1	2x2x2	2x2x2	–	120x120x155x64	60x60x78x64
C_e_2_1	32x3x3x3	1x1x1	same	60x60x78x64	60x60x78x32
C_e_2_2	32x3x3x3	1x1x1	same	60x60x78x32	60x60x78x32
P_2	2x2x2	2x2x2	–	60x60x78x32	30x30x39x32
C_e_3_1	16x3x3x3	1x1x1	same	30x30x39x32	30x30x39x16
C_e_3_2	16x3x3x3	1x1x1	same	30x30x39x16	30x30x39x16
P_3 (features)	2x2x2	2x2x2	–	30x30x39x16	15x15x20x16
C_d_3_1	16x3x3x3	1x1x1	same	15x15x20x16	15x15x20x16
C_d_3_2	16x3x3x3	1x1x1	same	15x15x20x16	15x15x20x16
Up_1	2x2x2	2x2x2	–	15x15x20x16	30x30x40x16
C_d_2_1	32x3x3x3	1x1x1	same	30x30x40x16	30x30x40x32
C_d_2_2	32x3x3x3	1x1x1	same	30x30x40x32	30x30x40x32
Up_2	2x2x2	2x2x2	–	30x30x40x32	60x60x80x32
C_d_1_1	64x5x5x5	1x1x1	same	60x60x80x32	60x60x80x64
C_d_1_2	64x5x5x5	1x1x1	same	60x60x80x64	60x60x80x64
Up_3	2x2x2	2x2x2	–	60x60x80x64	120x120x160x64
C_Out	4x1x1x6	1x1x1	valid	120x120x160x64	120x120x155x4

3.1.3 Feature extraction using trained encoder

Once the ConvNet is trained, we split the network into its encoder and decoder components. The encoder part is then used for generating compact representations of the patient studies. The final encoder layer outputs a tensor of size 15x15x20x16, which is then flattened into a vector f with $|f| = 72000$. This represents a compression rate of $124\times$. The feature extraction pipeline is illustrated in Figure 2.

3.2 Dimensionality reduction

Since the amount of training data is much smaller than the size of the ConvNet extracted feature (163 vs 72000), we need to further reduce the dimensionality of the features to allow of effective ML methods application. For this end, we use SVD.

We choose SVD for its numerical stability [12], and even though its computationally expensive, we only need to perform SVD once for downstream ML methods. The procedure starts by normalizing the input matrix, X to zero mean and unit variance, given by:

$$X \leftarrow \frac{X - \mu}{\sigma} \tag{1}$$

The correlation matrix, C, is then constructed by

$$C = \frac{X^T X}{n - 1} \tag{2}$$

where n is the number of patients. Then, we decompose using:

$$USV^T = X \tag{3}$$

where X is diagonalized and S is the singular values matrix. Using 3 in 2, we get:

$$C = \frac{VSU^T USV^T}{n - 1} = V\frac{S^2}{n - 1}V^T \tag{4}$$

The principal components are then given by

$$XV = USV^TV = US \tag{5}$$

From those principal components, the top n components are selected as the new reduced feature set. For this task, scikit-learn's SVD implementation is used.

3.3 Regression models pipeline

Having the final reduced feature set for each patient, we apply a number of different ML regression models. Our work pipeline uses scikit-learn [13] and AutoML [14] Python libraries. The pipeline we implemented is modular and allows for easy addition of ML regressors by software class inheritance.

For each ML regressor, the pipeline holds out one-fifth of the dataset (33 datapoints) for testing, and gives the remaining (130 datapoints) for the regressor to train on. The regressor then trains using 5-fold cross validation on the training dataset given to it. The loss function of the pipeline is mean squared error. It is given as:

$$L = \frac{1}{N}\sum_{i=1}^{n}(\hat{y}_i - t_i)^2 \tag{6}$$

where N is the number of patients, $\hat{y}_i$ is the actual patient survival time and t_i is the predicted patient survival time.

To measure the quality of the results we use the Pearson correlation coefficient.

$$R = \frac{\sum \hat{y}t}{\sqrt{\sum \hat{y}^2 \sum t^2}} \tag{7}$$

The range of R is [-1,1], where -1 indicate a negative linear correlation, 0 no correlation, and 1 positive linear correlation. In this work an R value close to 1 is ideal, whereas a value of 0 indicates that we cannot extract information from the regression, regardless of the MSE scores. The pipeline includes the following regression models: linear regression, KNN regression, logistic regression, and random forest regression. On top of that, it includes two ensemble based models: gradient boosting and AutoML.

The library packages that we use in the pipeline come with different parameters. For each model we find the the best performing parameters by grid searching. For KNN regression we try k values in increment of 5, and we try to use uniform weighing of neighbors or distance based weighing. For logistic regression we try an inverse regularization penalty in the range of $\{0.0001, 0.001, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2\}$, alongside L_2 norm. For random forest regression we try 1 to 100 trees in increments of 10. For gradient booster, an ensemble of 100 trees of maximum depth 25 with least squares loss function is created. We tried Huber loss function (which combines L1 and L2 for robustness) but it did not yield performance improvements. The number of estimators and learning rate was chosen based on [15] where Ridgeway et al argue that the learning rate should be kept small as gradient booster can overfit contrary to common knowledge. In AutoML we let the automatic ML library try all of its internal ML methods. AutoML uses Bayesian optimization and Sequential Model-based Algorithm Configuration (SMAC) on each of its models then builds an ensemble from those models.

4 Experiments and results

4.1 Training autoencoder

Network architectures and hyperparameters of *Same* layers are given in Tables 1 and 2. The increasing number of kernels in subsequent layers in the network was inspired by the success of VGG Net (aka Oxford Net) [16]. The final architecture was chosen using a heuristic process, in which a deep network was first developed that could overfit, which we then regularized using weight decay with ℓ_2 norm.

Table 2: Hyperparameters for the proposed method chosen using a validation set.

Name	Hyperparameter	Value
Initialization	weights	Xavier [17]
	bias	Xavier [17]
Regularization l_2 norm	λ	0.2
Training	iterations	50
	optimizer	ADADELTA
	batch_size	2
	learning rate l_r	1.0
	ρ	0.95
	ϵ	$1e^{-08}$

Table 3: Results using different regression methods.

Regressor	Average MSE of 5-fold testing	Average R value of 5-fold testing
Linear Regression	313047.8720	0.0758216
KNN	125048.0946	-0.1133987
Random Forest	142876.4466	-0.1243413
Logistic Regression	198153.7039	0.1151202
Gradient Booster	256059.8549	-0.0892946
	MSE of one time training	R value of one time training
AutoML	153341.6372	0.0914679

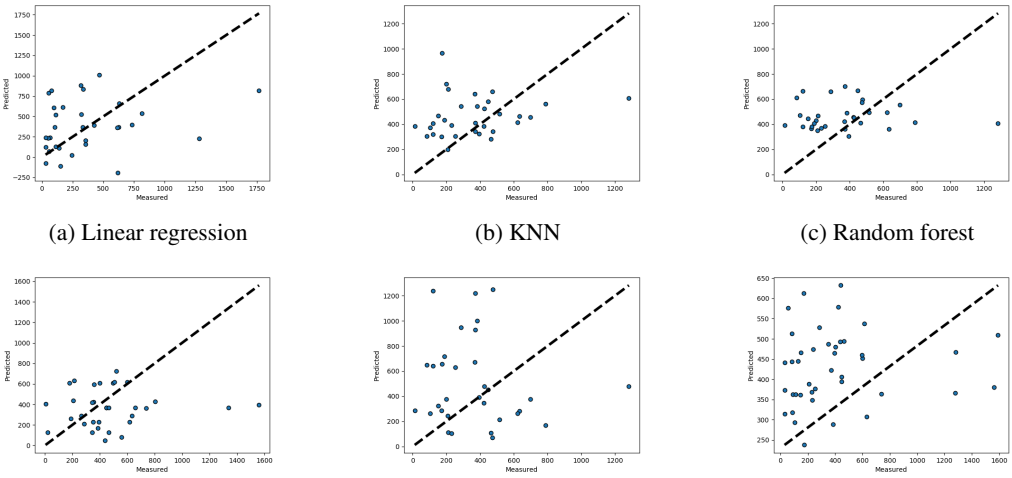
The hyperparameters used in this study that were required for the training process are given in the Table 2. These were chosen using a validation set consisting of one patient from the database. To train the network, a total of 210 patient studies were used, including those patients without reported survival time. The loss function to minimize was chosen to be MSE between the input MRI study and the reconstructed study. The weights were updated according to (ADADELTA) [18] method based on stochastic gradient descent (SGD) which adapts a learning rate using first order information. The main advantage of the method lies in the fact that it rules out the need for manual tuning of learning rate and is shown to be robust against noisy gradient values, different model architectures, various data modalities, and selection of hyperparameters. The proposed network was trained until the validation error stops improving or the training diverges. It was developed using Tensorflow [19] with a wrapper library Keras [20] in Python.

4.2 Regression models

Table 3 shows the MSE scores and R values for the regression models in our pipeline. We plot the Measured vs. Predicted patient survival times for the models as shown in Figure 3. We also investigate the effect of using features directly computed by the CNN, without further reduction using SVD, and found that it performs similar compared to using reduced feature set. All methods run until convergence, with the exception of AutoML, where it was allowed to run for 24 hours, although we observed that after an hour of searching through hyperparameter space it converges to its best performing model.

5 Discussion and conclusion

From the results in Table 3 we see that although we obtain MSE values similar to those in the state of the art [21] the R values are more informative. Our results show that none of the regressors outputs predictions linear in correlation with the ground truth. To ensure that this result, initially generated by KNN and linear regression, is not an artifact of a single regression technique, we resorted to ensemble methods to validate our observations. However, both gradient boosting regression and random forests gave similar results. Given that any regression technique is highly dependent on op-



(a) Linear regression (b) KNN (c) Random forest

(d) Logistic regression (e) Gradient booster (f) AutoML

Figure 3: x-axis of each sub-figure indicates the actual survival time of the patient, while the y-axis indicates the predicted survival time of the patient by the best performing parameter set of each model.

timal hyperparameter configuration, we used a state of the art optimization package to give further confidence in the results we obtained using grid search. This approach offers no improvement in MSE or R values. To exclude the potential effect of dimensionality reduction using SVD we tested our best regressor on the 72000 dimensional data and obtained similar results. The compressed representation produced by the encoder contains enough information for approximate reconstruction of the input. However the features do not necessarily translate to correlated prediction of survival time. Gliobastoma multiforme is a complex disease for which brain MRI captures only one manifestation, at a single point in time. The response of the patient to treatment is contingent on multiple factors such as genetic traits, treatment quality, age, sex, and medical history. From our empirical results and the stated domain knowledge we can conclude that in this framework predicting survival using brain MRI data alone is insufficient to produce predictions with high correlation.

6 Contributions

Our initial project looked at combining gene expression data with brain MRI to classify patients, however, significant technical challenges in the preprocessing stage of the MRI data made this project unfeasible. We then focussed on this project where our previous work could be reused. The whole team worked together on this project. We had almost weekly meetings for the two months preceding the submission of this report. We were all involved in all the steps of the project, while at the same time each of us was mainly focused on specific tasks that they were responsible for. Anmol headed the effort on data retrieval, preprocessing, CNN design, and graphics. Baraa undertook design and implementation of a modular and scalable pipeline, as well as planning and organizing the report. Bdour lead the effort on applying linear regression, and plotting regression results for all classifiers. Ben while contributing to the pipeline design with Baraa, developed the dimensionality reduction (SVD) part of the pipeline, and investigated ensemble learners. Rimika worked on KNN regression and contributed the grid search skeleton code for all methods. We enjoyed working together as a team, and believe that the final state of the project would not have been possible if it were not for every one of us. The source code and data is available at repository[1] for reproducibility.

[1] https://bitbucket.org/bcardoen/unlearning

References

[1] L. M. DeAngelis, "Brain tumors," *New England Journal of Medicine*, vol. 344, no. 2, pp. 114–123, 2001.

[2] S. Bauer, R. Wiest, L.-P. Nolte, and M. Reyes, "A survey of MRI-based medical image analysis for brain tumor studies," *Physics in Medicine and Biology*, vol. 58, no. 13, pp. R97—-R129, 2013.

[3] D. N. Louis, H. Ohgaki, O. D. Wiestler, W. K. Cavenee, P. C. Burger, A. Jouvet, B. W. Scheithauer, and P. Kleihues, "The 2007 WHO classification of tumours of the central nervous system," *Acta Neuropathologica*, vol. 114, no. 2, pp. 97–109, 2007.

[4] R. R. Edelman and S. Warach, "Magnetic Resonance Imaging," *New England Journal of Medicine*, vol. 328, no. 10, pp. 708–716, 1993.

[5] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, and Others, "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE transactions on medical imaging*, vol. 34, no. 10, pp. 1993–2024, 2015.

[6] D. C. Cire\csan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *International Conference on Medical Image Computing and Computer-assisted Intervention*, pp. 411–418, Springer, 2013.

[7] J.-Z. Cheng, D. Ni, Y.-H. Chou, J. Qin, C.-M. Tiu, Y.-C. Chang, C.-S. Huang, D. Shen, and C.-M. Chen, "Computer-aided diagnosis with deep learning architecture: applications to breast lesions in US images and pulmonary nodules in CT scans," *Scientific reports*, vol. 6, p. 24454, 2016.

[8] S. M. Plis, D. R. Hjelm, R. Salakhutdinov, E. A. Allen, H. J. Bockholt, J. D. Long, H. J. Johnson, J. S. Paulsen, J. A. Turner, and V. D. Calhoun, "Deep learning for neuroimaging: a validation study," *Frontiers in neuroscience*, vol. 8, 2014.

[9] J. Fehr and H. Burkhardt, "3d rotation invariant local binary patterns," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pp. 1–4, IEEE, 2008.

[10] R. Dupre and V. Argyriou, "3d voxel hog and risk estimation," in *Digital Signal Processing (DSP), 2015 IEEE International Conference on*, pp. 482–486, IEEE, 2015.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] G. Golub and W. Kahan, "Calculating the Singular Values and Pseudo-Inverse of a Matrix," *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, vol. 2, pp. 205–224, jan 1965.

[13] 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*, vol. 12, pp. 2825–2830, 2011.

[14] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2962–2970, Curran Associates, Inc., 2015.

[15] G. Ridgeway, "Generalized Boosted Models: A guide to the gbm package," 2007.

[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *CoRR*, vol. abs/1409.1, 2014.

[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[18] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[19] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[20] F. Chollet, "Keras," in *Github Repository*, https://github.com/fchollet/keras, 2017.

[21] "International MICCAI BraTS Challenge Pre-Conference Proceedings," 2017.