Machine Learning for Healthcare Project 3 Report

Arda Arslan
aarslan@student.ethz.ch

Gökberk Özsoy
goezsoy@student.ethz.ch

Sebastian Müksch
smueksch@student.ethz.ch

1 INTRODUCTION

Brain tumor is an aggressive disease which can affect adults and children alike. Fortunately, with Magnetic Resonance Imaging (MRI), physicians have access to a non-invasive and safe imaging technique to help detect brain tumors via cross-sections of patient brains. With over a million MRI exams a year in Switzerland alone [1], automated classification techniques, such as those provided through machine learning, pose a significant area of potential support for physicians. Unfortunately, the vast majority of physicians exhibit a distrust in such approaches [4]. Improving the interpretability of machine learning approaches is one potential way to alleviate trust issues. Inspired by this insight, we examine different approaches to brain tumor detection in MRI images, both intrinsically interpretable and post-hoc explainable. We find that within a small dataset of MRI images we are able to achieve an accuracy of up to 89.3%, while still being able to interpret the factors aiding the detection process.

Section 2 briefly discusses the datasets we used. Section 3 presents our methods and Section 4 presents details of our evaluation. Finally, Section 5 reflects on our findings and Section 6 adds closing remarks and future work.

2 DATASET

We use the Kaggle Brain Tumor Detection dataset, which consists 278 MRI brain slices, 111 without and 167 with a tumor visible to even non-expert eyes [10]. We split this dataset into 80% training and 10% validation and test set each. Additionally, we work with a version of this dataset that has been preprocessed into Pyradiomics features [11]. These broadly consist of first-order statistics, texture and shape features. They allow us to explore models that cannot typically work with raw image input. Overall, 474 Pyradiomics features are extracted from the raw MRI images.

3 METHODS

3.1 Task 1: Random Forest Baseline

As an initial baseline, we implement a random forest approach using an XGBoost classifier [3]. We train it on the Pyradiomics features described in Section 2. We perform a small hyperparameter grid search, exploring parameters such as the number of trees fitted, their maximum depth and the learning rate. For this grid search we combine the training and validation split of the dataset, since the validation split is particularly small with only 28 samples. With the combined splits we perform 5-fold cross validation to find the best hyperparameter settings. After this, we combined train and validation sets, and train the model again with the best hyperparameters.

3.2 Task 2: CNN Baseline

Convolutional neural networks (CNN) are the most popular choice for image classification tasks. We use the architecture provided in [10]. We remove the final Softmax activation from the provided model class. This is because we want to use the numerically stable CrossEntropyLoss function from PyTorch which essentially combines LogSoftmax and NLLLoss functions [2].

3.3 Task 3: Additional Interpretable Models and Post-Hoc Methods

3.3.1 Interpretable Model 1: Logistic Regression with L1 Regularization. Given descriptive features provided by Pyradiomics, logistic regression is a natural choice for this task. However, the sample size is small, and feature dimensionality is high, which might lead to over-fitting. At this point, L1 regularization reduces over-fitting by forcing uninformative features to be 0. It shows this pattern because it is an approximation of L0 loss, which counts the number of non-zero parameters as penalty. At the end, the model implicitly

Rule	Support
*LH_glrlm_GrayLevelVariance > 0.2498	0.8774
*HH_glrlm_GrayLevelVariance > 0.2498	0.8544
*HL_glrlm_GrayLevelNonUniformity > 5805.3884	0.8431

Table 1: Top 3 highest support rules according to RuleFit. *=wavelet-, i.e. all features above start with "wavelet-".

selects a subset of features for prediction, and a human can understand the model's internal working by analyzing these. We perform hyperparameter search on provided validation set, whose results provided in Figure 5. Feature standardization is needed, as L1 regularization directly minimizes parameter magnitudes. Lower C means stricter regularization, hence sparser parameter vector. After hyperparameter selection, we merge train and validation set, standardize, and train the model again, ready for prediction. As discussed above, by-product parameter magnitudes will tell us which parameters are important during the training.

3.3.2 Interpretable Model 2: RuleFit. Feature importances can give insights into which features are relevant for classification. However, even more useful are rules on how to make decisions based on the values of such features, which is why we implement the RuleFit classifier [6]. It provides learnt rules such as those in Table 1. Here, support gives insight into what proportion of the dataset that the rule applies to. Intuitively, the rules in Table 1 are sensible, as large variances in gray values hint at spots on the MRI which could in turn hint at tumors.

3.3.3 Post-Hoc Method 1: Permutation Feature Importance [5]. It measures the increase in test error after permuting a feature's values, destroying the relationship between feature and label discovered during training. A feature is informative if permuting results in severe increase in test error, and uninformative if permuting results in no change. It is model-agnostic, so only its response to permuted test set is analyzed. Here, the feature importances are estimated on test set, because during training model might memorize noise, and assign unrealistic importances. We select L1 + L2 regularized logistic regression as base model to this method. The reason behind this is preventing extreme implicit feature selection by sole L1 regularization and observing behaviour for all feature set.

3.3.4 Post-Hoc Method 2: Partial Dependence Plots [9]. It reveals the relationship between label and a particular feature, and computed by estimating the marginal effect on performance of a given value for a feature. After training the model, for each unique value of the selected feature, we replace all samples with that value, leaving the other features original, and report the performance. It is a model-agnostic method, and we select L1 + L2 regularized logistic regression as base model for the same reasons explained previous section.

3.4 Bonus Tasks

3.4.1 Transfer Learning. Due to scarcity of the data points, it is hard to achieve a sufficient generalization by training a deep neural network with randomly initialized weights. Therefore, we use pre-trained ResNet18 [7], and ResNeXt50_32x4d [12] architectures. We train these architectures using different data augmentation methods.

In each experiment, we use a pre-trained model, replace the final linear layer of it with a randomly initialized linear layer which outputs two logits. We only fine-tune the last convolutional layers (layers whose parent name is "layer4" in PyTorch) and the final linear layer.

3.4.2 Data Augmentation. Another way to address the problem of data scarcity is to use data augmentation. We train our CNN pipelines with different types of image processing / augmentation methods: normalization, horizontal flipping, vertical flipping, rotation, perspective, color jittering, and cropping. Details of each augmentation is presented at Appendix A.

4 EVALUATION

For each task, we present test accuracy, and number of selected features if applicable in Table 2. As expected, both simple and advanced deep networks outperform both XgBoost and Logistic Regression. ResNeXt50 with augmentation reaches the maximum test performance of 1.0 accuracy, with the support of extensive augmentations. For logistic regression, regularization coefficient plays key role affecting the model's performance.

We experiment intensively on transfer learning and data augmentation to improve test accuracy. Validation results for these experiments can be found in Table 3 which is in Appendix.

In Figure 1 and Figure 7, we present absolute values of parameter magnitudes as a result of training L1 regularized Logistic Regression with C= 1.0 and C= 0.1, respectively. Here, higher magnitude means higher feature importance. In Figure 6 and Figure 2, we present permutation importances for L1 + L2 regularized Logistic Regression for training and test sets, respectively. Finally, in Figure 3, we present partial dependence plots of random subset of features. The y-axis is the probability of a sample having tumor.

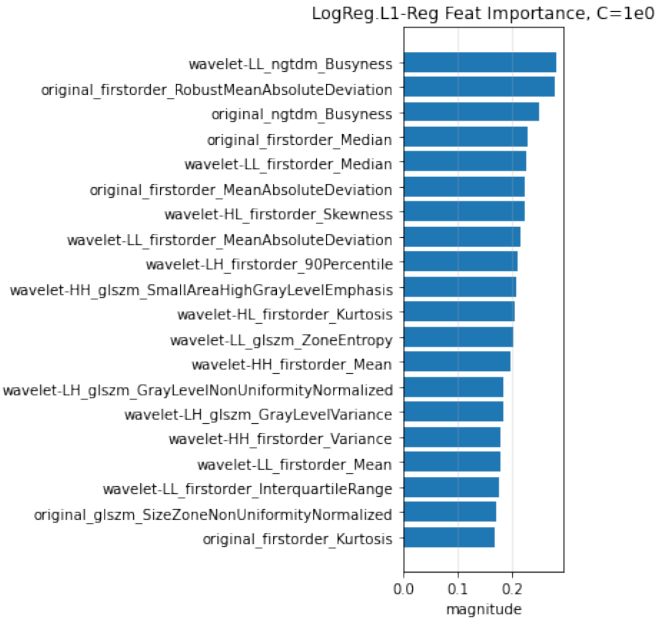

Figure 1: Logistic Regression Implicit Feature Selection with C = 1.0.

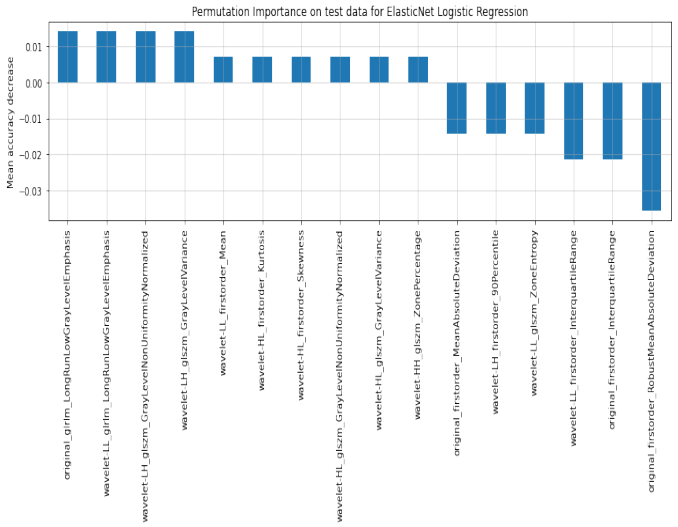

Figure 2: Permutation Feature Importances for ElasticNet Logistic Regression on Test Set.

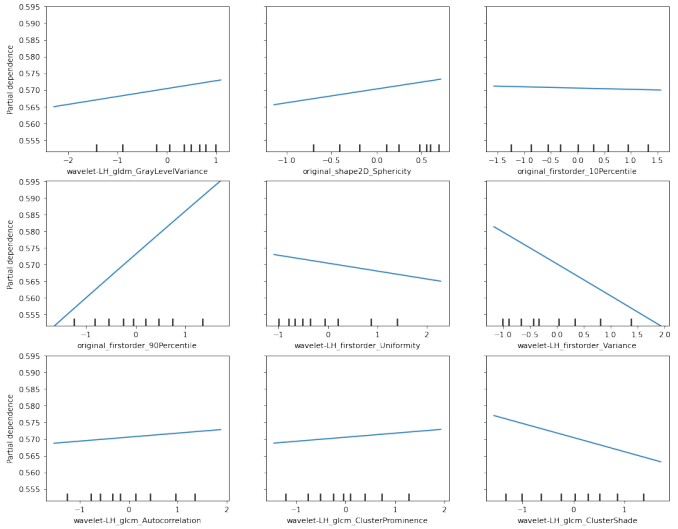

Figure 3: Partial Dependence Plots for selected subset of features.

5 DISCUSSION

For task 3, we have observed that high regularization (C=0.1) results in performance drop, but little regularization (C=10.0) gets affected by curse of dimensionality. In Figure 7, we see that high regularization pushes model to utilize only a few features for prediction, forcing all features to have 0 values and if not very small ones. On the contrary, as in Figure 1, correct regularization creates balanced feature importances, and hence higher performance and more trust on implicit feature selection.

For permutation post-hoc method, we observe that train and test feature importances are different, proving again the need of using this method with test data. Some advantages of this method are that it provides concise and global view of feature importances and does not require re-training. However, if features are highly correlated (as in our task) permuting an important feature

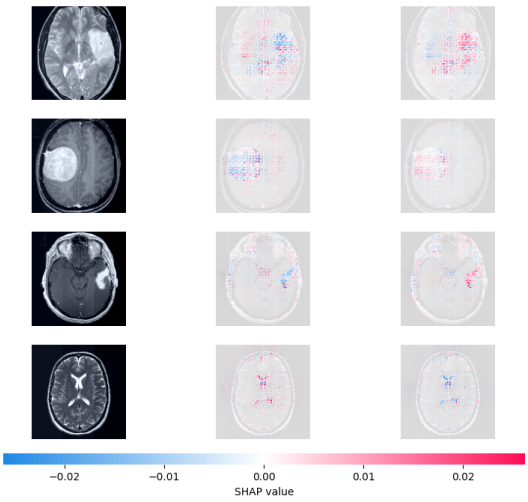

Figure 4: Shap values for the most-interpretable model (Baseline CNN, all augmentations, test set).

Model Name	# Selected Feat.	Test Acc.
XGBoost RF	240/474 ≈ 50.63%	0.75
CNN Baseline	–	0.857
Logistic Reg. + L1 reg.	263/474 ≈ 55.48%	0.79
RuleFit + RF	–	0.75
Bonus		
CNN Baseline + Aug.	–	0.893
ResNet18	–	0.929
ResNeXt50 + Aug.	–	**1.0**

Table 2: Performance comparison on test split. RF = Random Forest, Aug. = Augmentation.

might not affect the result as expected. In addition, we need labeled test set for measuring importances.

For partial dependence plots post-hoc method, each feature needs to be analyzed independently as in Figure 3. Features having less relationship with corresponding label have flat curves, while the important features will change model's outcome. One drawback of this method is it only works for uncorrelated features which is not the case in our task. This can be observed from the fact that none of the features can change the class label solely as apparent from the figure.

Since we have limited amount of data, we used data augmentation to increase number of datapoints. By this way, we achieved a better performance in both validation and test splits for Baseline CNN. For ResNet18 using data augmentation worsened the results. And for ResNeXt50, data augmentation whether had no effect or worsened the results. The last two cases can be due to the pre-trained neural architectures being already successful for solving our task, however we are creating a difference between train and evaluation splits.

To deal with data scarcity, we also used transfer learning. Fine-tuning a ResNet18 instead of using Baseline CNN improved the results on both validation and test splits. ResNeXt50 yielded a better result in validation split in average, and also a better result in test split.

For our neural network experiments, we also plot Shap [8] values produced by a tool named DeepExplainer. In Shap value images, the red pixels correspond to the regions where the neural network gives the most importance to, in order to predict a certain class. And the blue pixels correspond to the regions where the neural network looks at, in order to not predict a certain class. Due to an operation used in ResNeXt, we are not able to generate Shap values for this architecture. And we observed that our second best-performing model (ResNet18, no augmentation) does not produce interpretable Shap values (See Figure 8). Therefore, the most interpretable Shap values belong to the experiment where we used Baseline CNN with all augmentations (See Figure 4). One can see that the images at the right have red pixels on tumor areas, and images at the center have blue pixels on these areas.

Overall, we select Baseline CNN with data augmentation and Shap post-hoc method as the most useful one in real-world. A physician naturally tend to look for MRI images, instead of Pyradiomics features for diagnosing, thus this model would gain trust with its easy-to-understand and intuitive visualization, and high test accuracy at the same time.

6 CONCLUSION

Automated tumor classification helps physicians for fast diagnosing, yet it needs to be explainable to gain trust. In this work, we examined both interpretable models and post-hoc methods and managed to create one highly performant and visually intuitive model using data augmentations.

REFERENCES

[1] [n. d.]. Medical and technical equipment in hospitals and medical practices in 2019. https://www.bfs.admin.ch/bfs/

en/home/statistics/catalogues-databases/press-releases.assetdetail.16584132.html. Accessed: 2022-05-17.

[2] 2022. CROSSENTROPYLOSS. https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

[3] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[4] Keith J Dreyer. [n. d.]. Ensuring Trustworthiness of Diagnostic Imaging AI.

[5] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. 2018. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. (2018). https://doi.org/10.48550/ARXIV.1801.01489

[6] Jerome H Friedman and Bogdan E Popescu. 2008. Predictive learning via rule ensembles. *The annals of applied statistics* 2, 3 (2008), 916–954.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. https://doi.org/10.48550/ARXIV.1512.03385

[8] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4765–4774. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[9] Christoph Molnar. 2022. *Interpretable Machine Learning* (2 ed.). https://christophm.github.io/interpretable-ml-book

[10] Alain Ryser. 2022. interpretability-project. https://github.com/alain-ryser/interpretability-project

[11] Joost JM Van Griethuysen, Andriy Fedorov, Chintan Parmar, Ahmed Hosny, Nicole Aucoin, Vivek Narayan, Regina GH Beets-Tan, Jean-Christophe Fillion-Robin, Steve Pieper, and Hugo JWL Aerts. 2017. Computational radiomics system to decode the radiographic phenotype. *Cancer research* 77, 21 (2017), e104–e107.

[12] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2016. Aggregated Residual Transformations for Deep Neural Networks. https://doi.org/10.48550/ARXIV.1611.05431

A APPENDIX

A.1 Augmentation Details

Normalization: We normalize each image using predefined mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225) values. For example, to normalize red channel of an image, we subtract 0.485 from each pixel's red value (which is between 0 and 1) and divide it by 0.229.

Horizontal Flip, Vertical Flip and Perspective: Each of these operations are applied with a probability of 0.5. Random Perspective's distortion scale is 0.25.

Color Jittering: This operation randomly adjusts brightness, contrast, saturation and hue of an image. We used the value 0.4 for each of them.

Random Cropping: We first resize images to 1.10 of its original width and height, then we randomly select a region whose size is compatible with the neural architecture.

A.2 Additional Plots

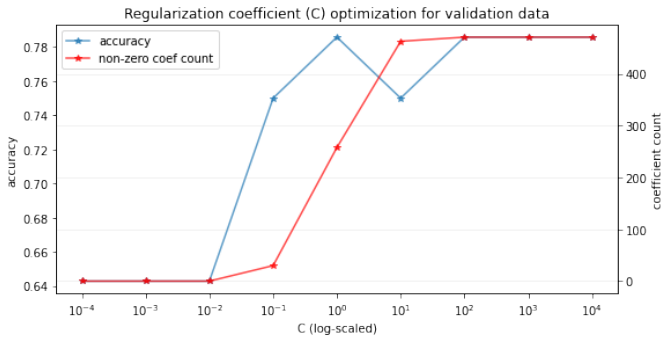

Figure 5: Hyperparameter optimization for Logistic Regression with L1 regularization

Processing and Augmentations	Val Acc.
Baseline CNN	
-	0.750
Normalization	0.750
Normalization + Horizontal Flip	0.750
Normalization + Vertical Flip	0.857
Normalization + Rotation	0.929
Normalization + Perspective	**0.964**
Normalization + Color Jittering	0.750
Normalization + Random Cropping	0.857
All	0.893
ResNet18 CNN	
-	**1.0**
Normalization	0.929
Normalization + Horizontal Flip	0.893
Normalization + Vertical Flip	0.964
Normalization + Rotation	0.929
Normalization + Perspective	0.964
Normalization + Color Jittering	0.857
Normalization + Random Cropping	0.929
All	0.929
ResNeXt50 CNN	
-	0.964
Normalization	0.964
Normalization + Horizontal Flip	0.964
Normalization + Vertical Flip	0.964
Normalization + Rotation	0.929
Normalization + Perspective	0.964
Normalization + Color Jittering	0.964
Normalization + Random Cropping	0.964
All	**0.964**

Table 3: Performance comparison on validation split for CNN models and different data augmentation methods.

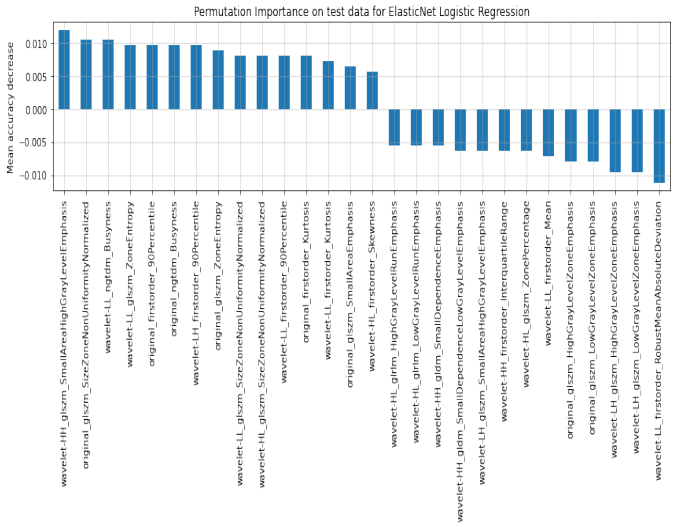

Figure 6: Permutation Feature Importances for ElasticNet Logistic Regression on Training Set

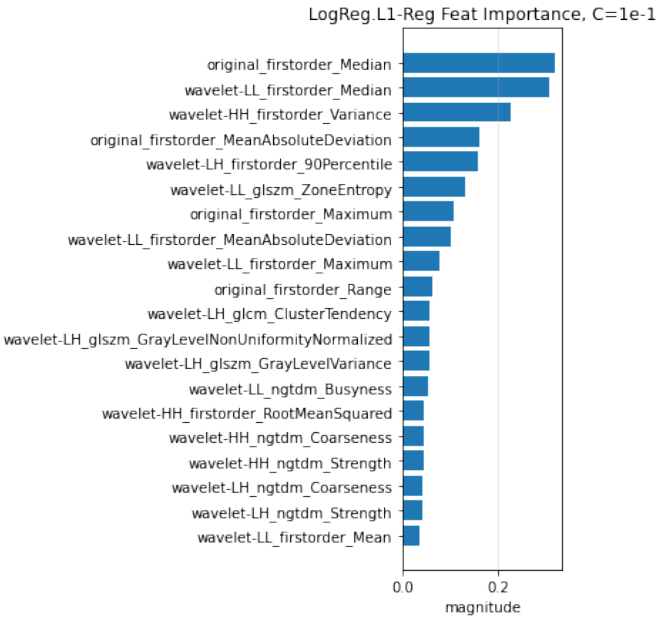

Figure 7: Logistic Regression Implicit Feature Selection with C = 0.1

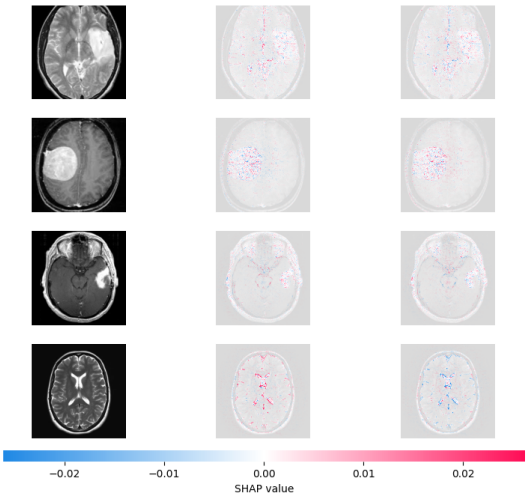

Figure 8: Shap values for the best-performing model (ResNet18, no augmentation, test set).