Brain Tumor MRI Classification

Hastuti Hera Hardiyanti,[1] Jeremy Citrawira,[2]
Veronica Angelin Setiyo,[3] Victoria Magdalena[4]

National University of Singapore[1,2,3,4]

e0878482@u.nus.edu,[1] e0758878@u.nus.edu,[2]
e0775085@u.nus.edu,[3] e0866233@u.nus.edu[4]

Introduction

Each year over 500 adults and 30 children undergo treatments in hospitals for brain tumors. The 5-year relative survival rate of a malignant brain tumor is almost 36%. This poses the importance of early detection of malignant tumors.

Magnetic Resonance Imaging (MRI) is currently the most popular method of detecting brain tumors. However, this manual examination is still prone to error due to the sheer complexities of tumors.

Applications of Machine-Learning (ML) and Artificial Intelligence (AI) are therefore essential to improve efficiency, improve accuracy, and minimise errors (especially Type-II errors). Combining this problem statement with our experience of learning several AI/ML algorithms in the module, our group aimed to simulate applications of Convolutional Neural Network (CNN), Support Vector Machine (SVM) and Decision Tree (DT) to help in image classification of brain tumors.

Literature Review

We learnt that there are numerous models that are capable of tackling the problem, mainly consisting of CNN, SVM and DT algorithms.

CNN is a Deep Learning algorithm, which assigns importance of local features in a picture, using trainable weights and biases. It involves a combination of convolutional layers that allows for automatic feature extraction at minimal cost. As such, CNN models work by automatically learning the important local features in image classification. Current literature. Current literature shows that CNN model is capable of achieving up to 96% accuracy in classifying brain tumours (Saeedi et al., 2023).

SVM builds a hyperplane in multi- dimensional space to differentiate categorical variables. In training, it will continue to improve the parameters of the hyperplane based on the loss functions, aiming to achieve minimum error. SVM models are more effective in high-dimensional data, especially when the number of dimensions is larger than the number of observations, due to the regularisation methods involved to prevent overfitting. In fact, a hybrid SVM-CNN deep learning model for tumor detection of brain images is reported to produce accuracy as high as 98.5% (Khairandish et al., 2022).

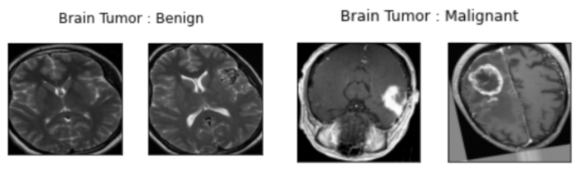
DT is a model built upon a set of simple rules. It is a nonparametric model because they do not require any assumptions about the distribution of the variables in each class. Just like its name, DT is structured with a tree representation. The decision-making process involves selecting between multiple choices or actions based on certain criteria or conditions.

There have been successful image classifications that produced accuracy up to 96% (Naik & Patel, 2014). These DT models that have produced high accuracies have one thing in common: they have used some methods to extract useful features in the images. One of the most commonly used methods to extract image features is Grey Level Co-occurrence Matrix (GCLM), which can extract features such as texture and energy (Naik & Patel, 2014).

In this project, we want to explore how DT would perform without the feature extraction.

Dataset

For this project, we are using the brain tumor dataset. It consists of 231 MRI images of two types of brain tumors: 154 malignant and 77 benign. In each folder, the images are saved as JPG files. Each image is in grayscale and has a size of 244x244.

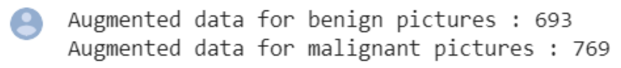


Data Pre-Processing

Data Augmentation

A brief look of the dataset shows us two problems that need to be tackled. There is a skewed class proportion in the dataset (2:1 ratio), and there is few labelled data (~200 images). To create more data based on the original data, we augmented our data to add more samples and balance the numbers of data in both classes.

We created a function `data_augment()` and addressed the unbalanced class problem successfully, resulting in ~1.1:1 ratio.



```
Augmented data for benign pictures : 693
Augmented data for malignant pictures : 769
```

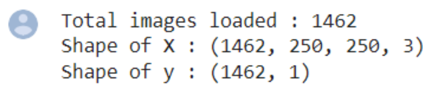
Cropping Image Data

We further preprocessed the augmented data visually by cropping the image to focus on the brain. We defined another function `crop_image()` to crop the images.

Loading Data into X,y

With the `crop_image()` function defined, we were able to crop and resize the images, which we then loaded into Numpy Arrays.

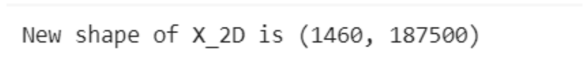
X stores the image, while y stores the label.



```
Total images loaded : 1462
Shape of X : (1462, 250, 250, 3)
Shape of y : (1462, 1)
```

Converting 4D data into 2D data

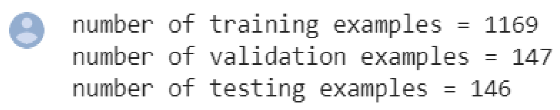
Some of the models that we used were not able to handle 4 dimensional data, which is the current representation of our data. As such, we created a function `convert_2D()` function that returns the 2D version of the data which is converted from the 4D version.



```
New shape of X_2D is (1460, 187500)
```

Splitting Data into Training, Validation and Test Data

Lastly, using `train_test_split()` (from sklearn. models_selection) , we split data into training data, validation data and test data, using 8:1:1 ratio of splitting.



```
number of training examples = 1169
number of validation examples = 147
number of testing examples = 146
```

Various Models for Image Classification

Predefining Important Evaluation Function(s)

Firstly, in the context of the problem statement, we realised that Type II errors are relatively more deadly and expensive, which leads us to define `weighted_accuracy()`, which takes into account the relative importance of False Negative to False Positive.

Secondly, we defined `kfold()` that implements the K-Fold Cross Validation Technique in assessing our models' performance to observe the average performance of our models in different sets of training.

Last but not least, we created `load_weights()`, which is an alternative way of running the project, by loading pre-trained weights from local directories (specifically for CNN and SVM).

Convolutional Neural Network (CNN)

The `cnn()` model is implemented based on CNN. It is defined in these few sequences:

1. Initialise the Input Layer

2. Initialise the Convolutional Layer
   It uses 32 filters (kernel), which is a 7x7 matrix, that will slide (convolve) over each 7x7 set of pixels from the input. We wish to use Rectified Linear Unit (ReLU) as the activation layer.

3. Initialise the MaxPooling Layer
   Reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer. In our case, we downsampled our inputs using an input window of size 4x4.

4. Initialise Batch Normalisation Layer
   This is done to avoid instability in our neural networks that may cause imbalance gradients which lead to vanishing/exploding gradient problems by normalising our data on the same scale.

5. Repeat Step 2-Step 4, with 64 filters and 3x3 kernel size for the Convolutional Layer.

6. Initialise Flatten & Dense Layer
   After obtaining the pooled feature map from the previous steps, we flattened our pooled map into a column vector. This vector was fed into a dense layer, with an activation layer using sigmoid function, to return the output.

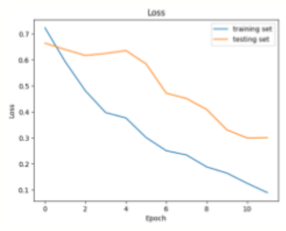


Figure 1.1 CNN Loss History

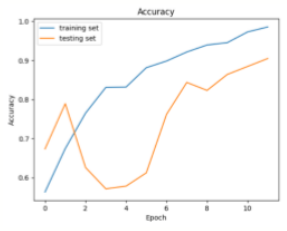


Figure 1.2 CNN Accuracy History

Figure 1.1 and 1.2 shows the training history of binary cross entropy losses and accuracy of the model in each epoch using adam optimizer.

Support Vector Machines (SVM)

In this model, we explored the implementations of Hybrid SVM-CNN Model.

The `svm_cnn()` model is defined in these few sequences:

1. Follow CNN Procedure Step 1-5
   This is to utilise CNN layer's strengths in extracting the important local features.

2. Initialise Flatten & Dense Layer
   Unlike the use of sigmoid functions as the activation function in the Dense Layer, we will be using the l2 regularisation kernel (a standard form of SVM) with a learning rate of 0.01, combined with linear activation function, which minimises its loss function depending on the sum of the squared of the weights.

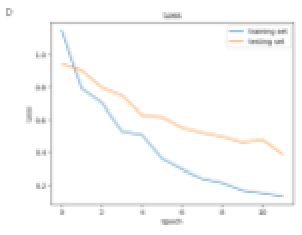


Figure 2.1 SVM Loss History

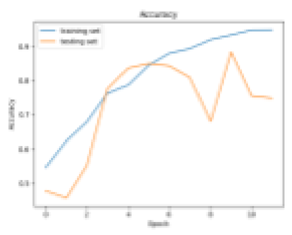


Figure 2.2 SVM Accuracy History

Figure 2.1 and 2.2 shows the training history of hinge losses and accuracy of the model in each epoch using adam optimizer.

Decision Tree (DT)

The `tree_cv()` model is implemented based on Decision Tree. As decision tree is a non-parametric model, there are no parameters to be trained. However, decision trees still have hyperparameters that can be tuned to improve their performance.

We chosen the following hyperparameters to be tuned:

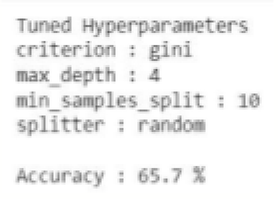
1. criterion
   This determines how to split the data at each node. We chose "gini" and "entropy" as they are the most commonly used criteria in decision trees. This is because they are effective, computationally efficient, and can be used in a wide range of scenarios.

2. splitter
   This determines how to divide the data at each node. "best" chooses the best split while "random" chooses the best random split.

3. max_depth
   This is the maximum depth that we allow the tree to grow to. Deeper trees make more complex models, and generally will decrease the training error. However, if the tree grows too deep, it can cause overfitting & increase in testing error. Conversely, if the tree is too shallow, it might be too inflexible to capture the patterns and interaction in the training data.

4. min_samples_split
   This is the minimum number of samples required to be in a node to split the node. Setting it too low can result in overfitting while setting it too high can result in underfitting.

`GridSearchCV()`, a cross-validation technique, is used to find the optimal hyperparameters from the set of hyperparameters specified. cv = 5 was chosen to be consistent with the k-fold cross validation of the other models. After running fit with all sets of the hyperparameters, the following best hyperparameters were obtained with a mean cross-validated score of the best estimator of 65.7% (Figure 3).



```
Tuned Hyperparameters
criterion : gini
max_depth : 4
min_samples_split : 10
splitter : random

Accuracy : 65.7 %
```

Figure 3 Tuned Hyperparameter and Accuracy of Decision Tree

Results & Discussions

We ran the models more than 5 times. Due to the randomised train-test-validation splitting each time, we got different performance results each time.

In order to observe the average performance, we utilised 5-fold cross-validation techniques for all our models. The metrics that we are looking at are the standard metrics (Accuracy, Precision, Recall, F1) and weighted accuracy.

For CNN and SVM models, in each fold, we generated predictions to our X_test images of size 146, and compiled the statistics into a PrettyTable format (Appendix A & B) for our analysis.

Convolutional Neural Network (CNN)

From the table (see Appendix A), we can observe that CNN performs generally and steadily well across the 5 folds, achieving 81% accuracy. Although, this is sub-par as compared to the current 96.0% accuracy achieved (Saeedi et al., 2023), our model only utilised 2 convolutional layers, as compared to the 8 layers used in current research.

In particular, CNN is very competent in weighted accuracy, consistently outperforming relative to other metrics. This may be indicative of the fact that our prediction errors consist of a lesser proportion of False Negatives. This is good news to us, as we require our models to commit minimal Type II error.

Support Vector Machines (SVM)

Since our SVM model incorporates the CNN layering for feature extraction, we expect the similar performance in the metrics assessed (hence the similar performance peaks in weighted accuracy), achieving 79% accuracy (see Appendix B). This is inferior as compared to performance in the literature of 98.5%, achieved in using hybrid SVM-CNN model (Khairandish et al., 2022).

Albeit not extreme, we can observe lesser consistency and performance as compared to our CNN model. This is possibly caused by the fact that our brain tumour images naturally have a lot of noises, causing the distinction of target classes ("malignant" and "benign") to be slightly overlapping. As such, that might explain the worse accuracy achieved when using SVM classifiers to detect the classes.

Decision Tree (DT)

DT performed much more poorly as compared to the literature (Naik & Patel, 2014), with weighted accuracy of 57.742% (see Appendix C). This tallies with our initial expectation that DT would perform poorly as DT is not known to be very good with image processing. For decision trees to perform well, they require features to be meaningful (Calomme, 2019). However, images have some of the least meaningful features as each attribute is just the brightness of a pixel. We can see this better from analysing the decision tree.

As we can see splitting in the decision tree (see Appendix C), the decisions are decided based on the brightness of particular pixels. For 2 of the same image (hence the same class) but different orientation, the classification can be starkly different merely because of the location of the pixels. With this in mind, it makes sense that using DT to classify unprocessed images would produce poorly performing models.

Visualizing Performance across Models

The graphical representation (in Appendix E) indicates that CNN reigns superior in all metrics, achieving the highest accuracy, precision, recall, F1-score, and weighted accuracy. SVM model performs nearly as well as CNN, where Decision Tree performs worse than the other two models.

Future Improvements

For CNN, we could further explore the gain in accuracy (and other metrics) over computational cost when we tune its hyperparameter (e.g. adjusting the number of Convolutional Layers, incorporating Dropout layer to further prevent overfitting).

For SVM, exploration of different standalone models (such as Support Vector Classifier (SVC), and LinearSVC) could be much more indicative of SVM's capability of classifying images, where currently with the hybrid SVM-CNN model, it is slightly ambiguous whether SVM classifier has bigger impacts on the current performance.

For Decision Tree, feature extraction using GLCM could be explored to obtain more meaningful features. With some image feature extraction to the image before fitting it to the decision tree, we expect the performance to be better than without any image feature extraction. Moreover, with higher computing power, more hyperparameter options could be added to explore which combinations would make the model perform better.

Interestingly, loading pre-trained weights of SVM and CNN for evaluation generates their respective evaluation metrics with higher variance than expected (when we compare the metrics in several runs). There is room for further discussion as to the cause behind this phenomenon, such as a possible cause of overfitting in our models.

References

Basic facts about brain tumours. Brain Tumour Society (Singapore) Limited. (n.d.). Retrieved March 29, 2023, from https://www.braintumoursociety.org.sg/basic-facts-about-brain-tumours/#:~:text=Each%20year%20over%20500%20adults,different%20types%20of%20brain%20tumours.

Brain Tumor - statistics. Cancer.Net. (2023, March 9). Retrieved March 29, 2023, from https://www.cancer.net/cancer-types/brain-tumor/statistics#:~:text=The%205%2Dyear%20relative%20survival%20rate%20for%20a%20cancerous%20brain,vary%20based%20on%20several%20factors

Calomme, V. (2019). Use of decision trees for classifying images. Stack Exchange. Retrieved April 1, 2023, from https://datascience.stackexchange.com/questions/64289/use-of-decision-trees-for-classifying-images

Convolutional Neural Network. Engati. (n.d.). Retrieved March 29, 2023, from https://www.engati.com/glossary/convolutional-neural-network

How to interpret classification report of scikit-learn? (n.d.). Data Science Stack Exchange. https://datascience.stackexchange.com/questions/64441/how-to-interpret-classification-report-of-scikit-learn

Joseph, V. R. (2022). Optimal ratio for data splitting. Statistical Analysis and Data Mining, 15(4), 531–538. Retrieved March 29, 2023, from https://doi.org/10.1002/sam.11583

K, D. (2023, February 7). *Top 4 advantages and disadvantages of support vector machine or SVM*. Medium. Retrieved March 29, 2023, from https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107

Khairandish, M. O., Sharma, M., Jain, V., Chatterjee, J. M., & Jhanjhi, N. Z. (2022). A hybrid CNN-SVM threshold segmentation approach for tumor detection and classification of MRI Brain Images. *IRBM*, *43*(4), 290–299. Retrieved March 29, 2023, from https://doi.org/10.1016/j.irbm.2021.06.003

Khan, M. S. I., Rahman, A., Debnath, T., Karim, M. R., Nasir, M. K., Shamshirband, S., Mosavi, A., & Dehzangi, I. (2022). Accurate brain tumor detection using deep convolutional neural network. Computational and Structural Biotechnology Journal, 20, 4733–4745. Retrieved March 29, 2023, from https://doi.org/10.1016/j.csbj.2022.08.039

Liu, Y., Tsalik, E. L., Jiang, Y., Ko, E. M., Woods, C. W., Henao, R., & Evans, S. R. (2020). Average Weighted Accuracy: Pragmatic Analysis for a Rapid Diagnostics in Categorizing Acute Lung Infections (RADICAL) Study. Clinical Infectious Diseases, 70(12), 2736–2742. https://doi.org/10.1093/cid/ciz437

Naik, J., & Patel, S. (2014). Tumor Detection and Classification using Decision Tree in Brain MRI. International Journal of Computer Science and Network Security, 14(6), 40-44. Retrieved March 29, 2023, from http://paper.ijcsns.org/07_book/201406/20140615.pdf

OpenCV: Contours Hierarchy. (n.d.). https://docs.opencv.org/4.x/d9/d8b/tutorial_py_contours_hierarchy.html

Rock, V. (2022, August 11). Brain Tumor FAQs - Learn More or Donate Today! | ABTA. American Brain Tumor Association. https://www.abta.org/about-brain-tumors/brain-tumor-education/#:~:text=More%20than%2084%2C000%20people%20were,(CNS)%20tumors%20are%20malignant

S. (2020, December 11). Brain-Tumor-Detection/CNN on Harvard Medical Dataset.ipynb at main · saikat15010/Brain-Tumor-Detection. GitHub. Retrieved March 29, 2023, from https://github.com/saikat15010/Brain-Tumor-Detection/blob/main/CNN%20on%20Harvard%20Medical%20Dataset.ipynb

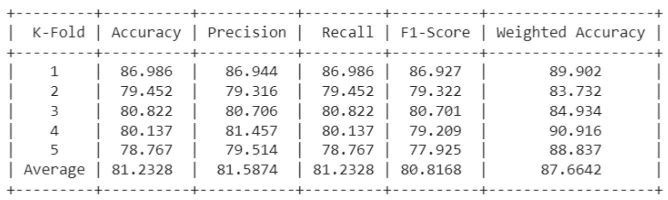
S. (2021, November 10). Image classification using SVM. Kaggle. https://www.kaggle.com/code/saumyamohandas/image-classification-using-svm

Saeedi, S., Rezayi, S., Keshavarz, H., & R. Niakan Kalhori, S. (2023, January 23). *MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques - BMC Medical Informatics and decision making*. BioMed Central. Retrieved April 2, 2023, from https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-023-02114-6#:~:text=The%20training%20accuracy%20of%20the%20proposed%202D%20CNN%20was%20found,developed%20to%20classify%20brain%20tumors.

Taha, B. (2021, August 27). Build an Image Classifier With SVM! Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/build-an-image-classifier-with-svm/

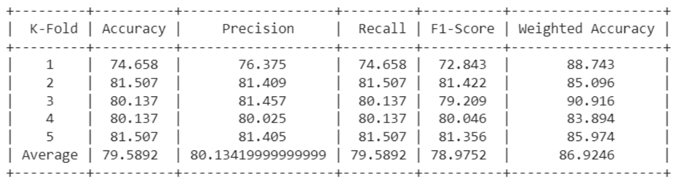
Types of Morphological Operations - MATLAB & Simulink.(n.d.). https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html

Appendix



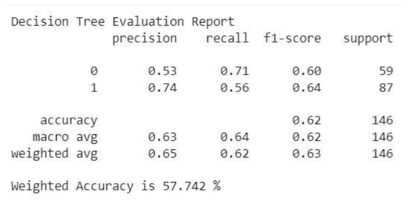
```
+---------+----------+-----------+---------+----------+-------------------+
|  K-Fold | Accuracy | Precision |  Recall | F1-Score | Weighted Accuracy |
+---------+----------+-----------+---------+----------+-------------------+
|    1    |  86.986  |   86.944  |  86.986 |  86.927  |       89.902      |
|    2    |  79.452  |   79.316  |  79.452 |  79.322  |       83.732      |
|    3    |  80.822  |   80.706  |  80.822 |  80.701  |       84.934      |
|    4    |  80.137  |   81.457  |  80.137 |  79.209  |       90.916      |
|    5    |  78.767  |   79.514  |  78.767 |  77.925  |       88.837      |
| Average  | 81.2328  |  81.5874  | 81.2328 | 80.8168  |      87.6642      |
+---------+----------+-----------+---------+----------+-------------------+
```

Appendix A: Performance of K-Fold CNN



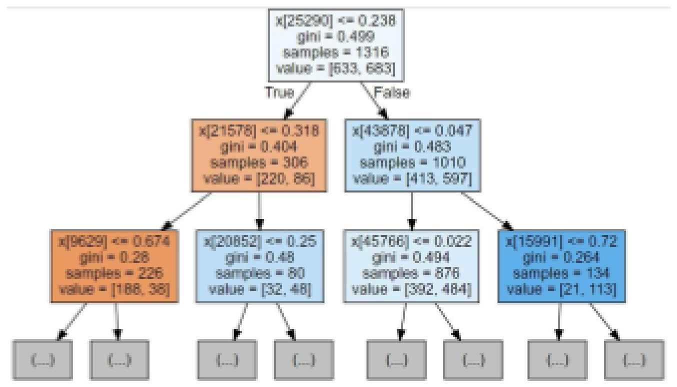
```
+---------+----------+--------------------+---------+----------+-------------------+
| K-Fold  | Accuracy |     Precision      |  Recall | F1-Score | Weighted Accuracy |
+---------+----------+--------------------+---------+----------+-------------------+
|    1    |  74.658  |       76.375       |  74.658 |  72.843  |       88.743      |
|    2    |  81.507  |       81.409       |  81.507 |  81.422  |       85.096      |
|    3    |  80.137  |       81.457       |  80.137 |  79.209  |       90.916      |
|    4    |  80.137  |       80.025       |  80.137 |  80.046  |       83.894      |
|    5    |  81.507  |       81.405       |  81.507 |  81.356  |       85.974      |
| Average | 79.5892  | 80.13419999999999  | 79.5892 | 78.9752  |      86.9246      |
+---------+----------+--------------------+---------+----------+-------------------+
```

Appendix B: Performance of K-Fold SVM-CNN

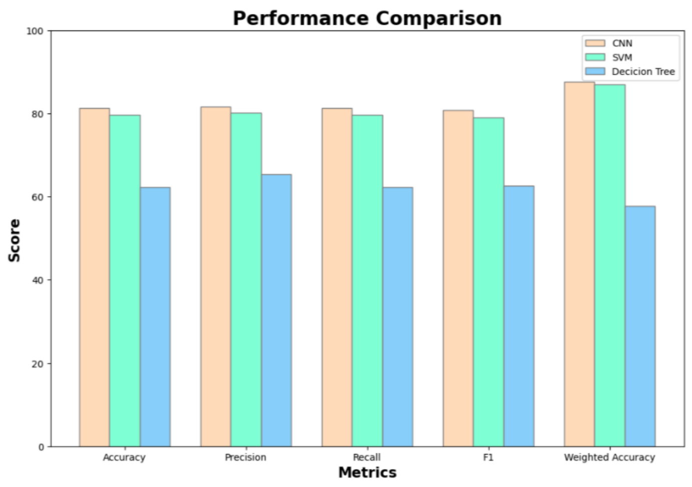


```
Decision Tree Evaluation Report
              precision    recall  f1-score   support

           0       0.53      0.71      0.60        59
           1       0.74      0.56      0.64        87

    accuracy                           0.62       146
   macro avg       0.63      0.64      0.62       146
weighted avg       0.65      0.62      0.63       146

Weighted Accuracy is 57.742 %
```

Appendix C: Performance of Decision Tree



Appendix D: The Visualisation of the Decision Tree with max_depth = 2



Appendix E: Performance Comparison of the 3 Models