Capstone Project Documentation

Authors: Laura Madrid, Lucas Noritomi-Hartwig, Keshav Worathur

Project Repository: https://github.com/kworathur/CV-Capstone/tree/main

Problem

Brain tumours affect approximately 50,000 Canadians every year, according to the Brain Tumour Registry of Canada (BTRC) [1]. Manual diagnosis of brain tumours is time-intensive and requires specialized knowledge of the brain [2]. We seek to develop automatic methods for diagnosis of glioma, meningioma, and pituitary tumors from Magnetic Resonance Imaging (MRI) scans. This entails a multi-class classification problem to which we hope to apply machine learning techniques in new ways.

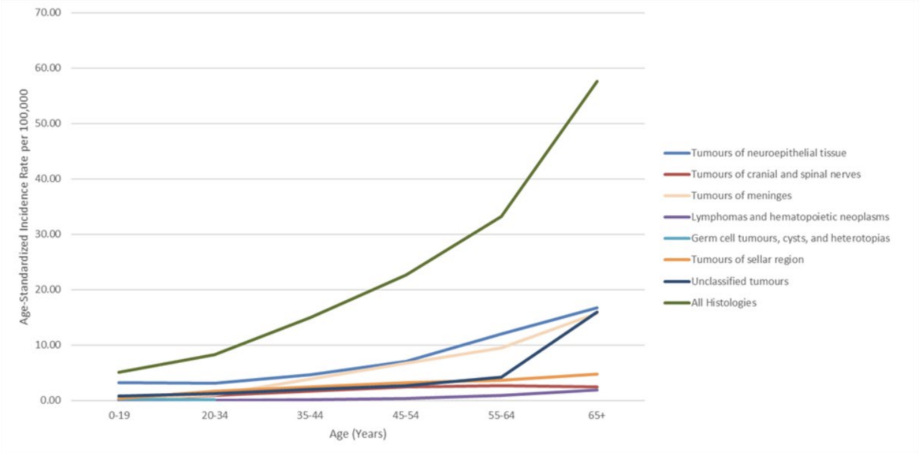

Figure 1: Tumor Incidence Rates

Dataset

Our chosen dataset is the figshare brain tumour dataset [3]. This dataset is available at kaggle. The dataset contains 3,064 slices of MRI scans from patients at Nanfang Hospital, Guangzhou, China. The slices were taken in the sagittal, axial, and coronal planes.

Examples of the tumour scans are shown below:

The input to the model is a MRI slice resized to 256×256 pixels. The labels are 1 for Meningioma, 2 for Glioma, and 3 for Pituitary.

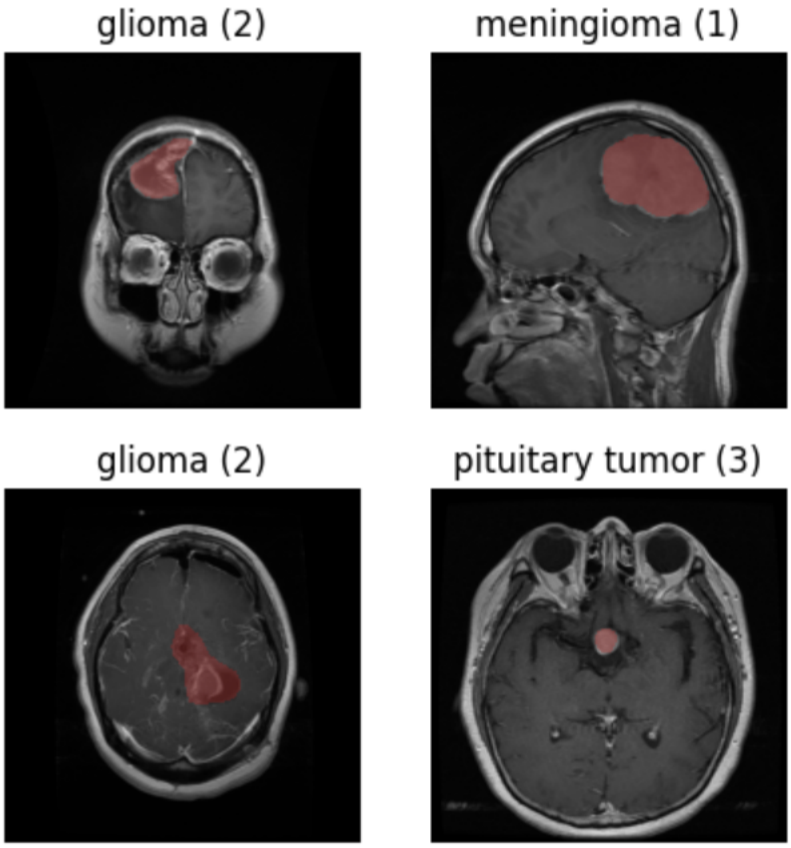

Figure 2: Examples of MRI slices

The dataset notably includes patient IDs, which we will use during evaluation of the model to prevent the model from being evaluated on patients on which it was trained. # Implementation Details

We implemented a custom Convolutional Neural Network (CNN) in tensorflow. The architecture is detailed in the paper "Classification of Brain Tumors using a Convolutional Neural Network"[3]

Loading the Data

To load in the dataset, we created a custom pre-processing pipeline:

1. Resize the image from 512×512 pixels to 256×256 pixels.
2. Augment the image by performing a $90°$ rotation or be performing a flip over the horizontal axis.
3. Standardizing the pixels values in the input image.

We load the images into memory in batches of 16, to avoid exceeding the RAM quota in Google Colab. We trained our model using the NVIDIA T4 GPU available in Colab.

Model Implementation

Our model consists of four classification blocks, arranged in a sequence to downsample the input image into a summary vector of size 2048. A diagram containing two such classification blocks is shown below:

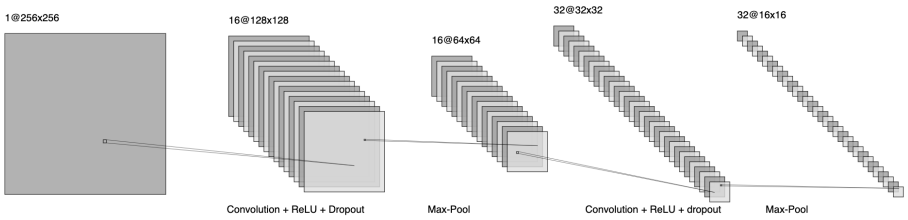

Figure 3: Diagram Depicting Classification Blocks

The model outputs a **softmax** vector of probabilities, where each probabilitity represents the likelihood that a tumor class is present in the MRI scan. We take the class corresponding to the largest probability as the model's prediction.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resizing_1 (Resizing)       (None, 256, 256, 1)       0

 normalization (Normalizati  (None, 256, 256, 1)       131073
 on)

 conv2d (Conv2D)             (None, 128, 128, 16)      416

 dropout (Dropout)           (None, 128, 128, 16)      0

 max_pooling2d (MaxPooling2  (None, 64, 64, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        4640

 dropout_1 (Dropout)         (None, 32, 32, 32)        0

 max_pooling2d_1 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 dropout_2 (Dropout)         (None, 16, 16, 64)        0

 max_pooling2d_2 (MaxPoolin  (None, 8, 8, 64)          0
 g2D)

 conv2d_3 (Conv2D)           (None, 8, 8, 128)         73856

 dropout_3 (Dropout)         (None, 8, 8, 128)         0

 max_pooling2d_3 (MaxPoolin  (None, 4, 4, 128)         0
 g2D)

 flatten (Flatten)           (None, 2048)              0

 dense (Dense)               (None, 2048)              4196352

 dense_1 (Dense)             (None, 3)                 6147

=================================================================
Total params: 4430980 (16.90 MB)
Trainable params: 4299907 (16.40 MB)
Non-trainable params: 131073 (512.01 KB)
_________________________________________________________________
```

Figure 4: Listing of Model Layers

Problems we could not solve

When we planned our project, we aimed to explain the decisins of our model by generating **counterfactual** images. Counterfactual images help us answer questions like:

> How would this scan of a patient with a Meningioma tumor be different if the patient instead had a Glioma tumor?

Exploring such hypothetical scenarios gives us insight into how the image can be changed to produce a different prediction from the model. This would in turn allow us to gain more insight into the model's decision making process.

Novel Contributions

When a model and a neurologist differ in their opinions about a scan, how can we reconcile these differences? Our novel contribution is the use of saliency maps for interpreting our classifier's decisions. Our saliency maps are computed by taking partial derivates of the predicted class with respect to each pixel of the original image.

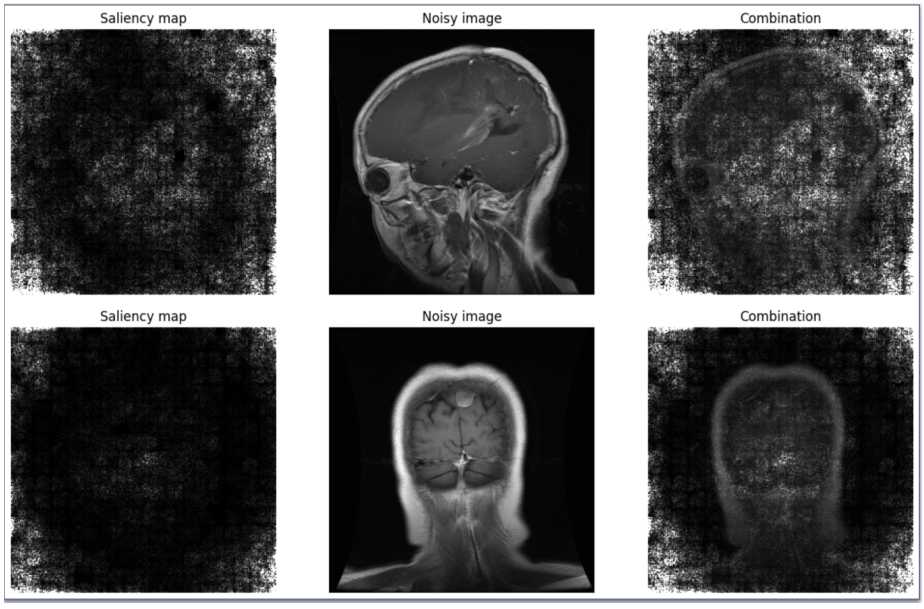

Figure 5: Saliency Maps

Evaluation Results

When evaluating our model on a single test set, we obtained a test accuracy of 93.4%.

We also used subject-wise cross validation for evaluating the model:

- Split the dataset into 10 subsets, called **folds**, where each patient can appear in only one fold. The number of distinct patients in each fold is roughly equal.
- Set aside two folds for the test set, two folds for the validation set, and using the remaining six folds as the training dataset.
- Train the model and record the test accuracy.
- Repeat this process until every example has appeared in the test set. Average the test accuracies to produce an estimate of the model's performance on unseen data.

We obtained an average test accuracy of 91.3%. This shows the model can differentiate between the different tumor classes. However, there is strong evidence the model may misclassify tumors when deployed in the real world.

Individual Contributions

Throughout the semester, we worked as a group to develop our project and address issues as they arised. All members contributed equally to the project, with individual contributions listed below:

Laura Madrid explored novel extensions to our project, such as GANs for producing counterfactual images and the vanilla gradient method of producing saliency maps.

Lucas Noritomi-Hartwig selected the dataset and performed data preprocessing. He also researched novel extensions to our project and wrote the code for saliency maps.

Keshav Worathur researched related works pertaining to our problem and set up the project repository. He wrote code for the data pre-processing pipeline and trained the model.

References

[1]Brain Tumour Registry of Canada. https://braintumourregistry.ca/, 2019. Accessed: 2023-10-01.

[2] E. S. Biratu, F. Schwenker, Y. M. Ayano, and T. G. Debelee, "A survey of brain tumor seg- mentation and classification algorithms," J Imaging, vol. 7, Sept. 2021.

[3] Milica M Badža and Marko Č Barjaktarović. Classification of brain tumors from mri images using a convolutional neural network. Applied Sciences, 10(6):1999, 2020.

Workflow

The notebook `brain_tumor_classifcation.ipynb` contains code for:

- Downloading the dataset from Kaggle.
- Extract images from .mat files and save to .png files.
- Visualizing MRI slices with tumours highlighted.
- Data pre-processing and augmentation.
- Training the model saving model checkpoints to `best_model.h5`.
- Producing saliency maps from the trained model.

To open the notebook in Google Colab:

1. File > Open Notebook
2. Open Notebook > From Github
3. Paste the repository url () in the search bar and click `brain_tumor_classification.ipynb` to open the notebook.
4. Runtime > Change Runtime Type > T4 GPU
5. Runtime > Run All
6. The second cell contains the following code:

   ```
   from google.colab import files
   files.upload() # Upload your kaggle API Key
   ```

   This waits until the user uploads their kaggle API key, named kaggle.json. The notebook runs without user input, first visualizing the data then training the model. The saved model is saved to best_model.h5
7. If you would only like to view the saliency maps, locate the section titles "Load the Best Model", and upload the pretrained model `best_model.h5` (found in this repository). Then, run the remaining cells to view the saliency maps.