Udacity AWS ML Engineer: Final project proposal

Domain background

In this project I'll be working on a problem from the medical imaging domain. In particular, I will look into automatically identifying lower-grade gliomas (a form of brain tumour) from MRI scans. Glial cells are non-neuronal cells in the brain and spinal cord that provide support and protection for neurons. Brain tumours that start in these glial cells are known as Gliomas. The segmentation of gliomas is an important task for the early diagnosis of brain tumours. According to Buda et al. (2019), several studies have indicated that that tumour shape may be indicative of patient prognosis. However, obtaining tumour features requires manual segmentation of brain MRI images, which is a time-consuming and tedious process. Deep learning offers the potential to automate this process which provides important pre-operation information and may assist with the identification of tumour genomic subtype.

Problem statement

The objective of this study is to train and deploy a segmentation model to automatically identify lower-grade gliomas from input MRI images. Examples of MRI scan slices are shown in Figure 1, along with overlaid segmentation masks which indicate the location of the tumour. This will be performed using AWS SageMaker. Here the main goal is to develop an end-to-end solution rather than achieving the best model performance.

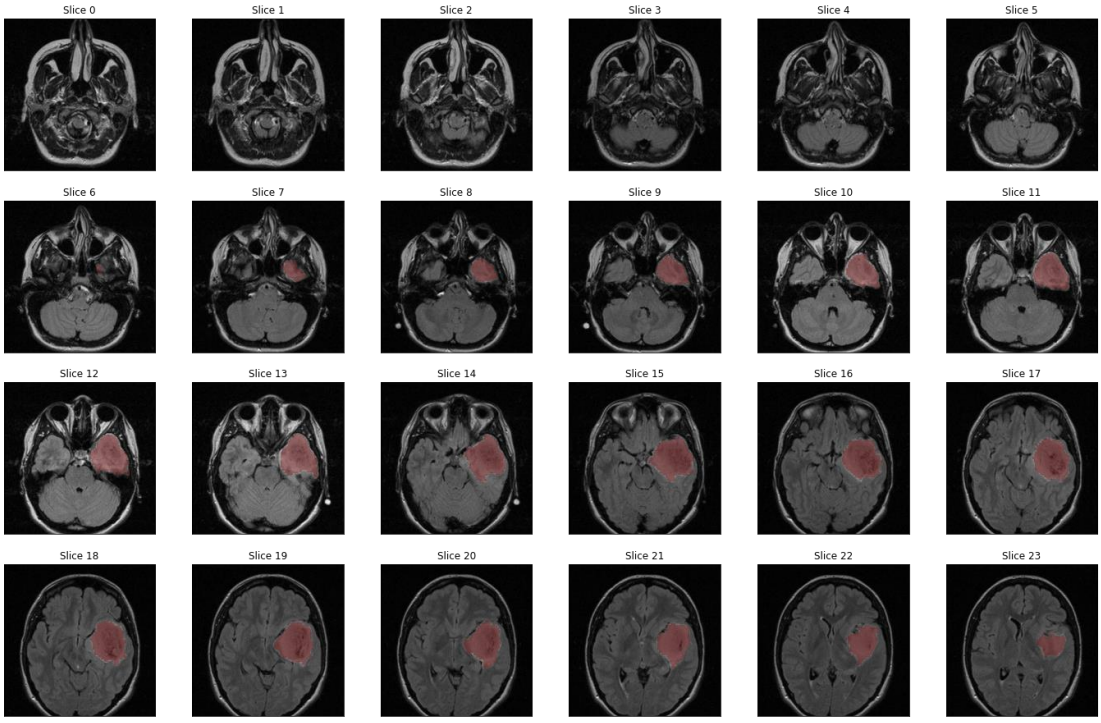


Figure 1 – Example patient MRI dataset and overlaid segmentation mask (red). Data from Kaggle Brain MRI segmentation dataset.

Datasets and inputs

In this project I'll be working with the Brain MRI Segmentation dataset from Kaggle. This includes brain MRI images from 110 patients and corresponding segmentation masks obtained from The Cancer Imaging Archive (TCIA). The original 3D MRI volumes were split into 2D slices, each 256 x 256 pixels in size. Each patient has a different number of slices, ranging from 20 to 88 slices.

The input MRI images consist of three channels, which correspond to pre-contrast, fluid attenuated inversion recovery (FLAIR) and post-contrast channels. An example of several slices from one patient are shown in Figure 2 below. The segmentation mask in this case is binary, indicating whether there is FLAIR abnormality or not. These masks were produced manually by a medical school graduate and verified by an experienced radiologist.

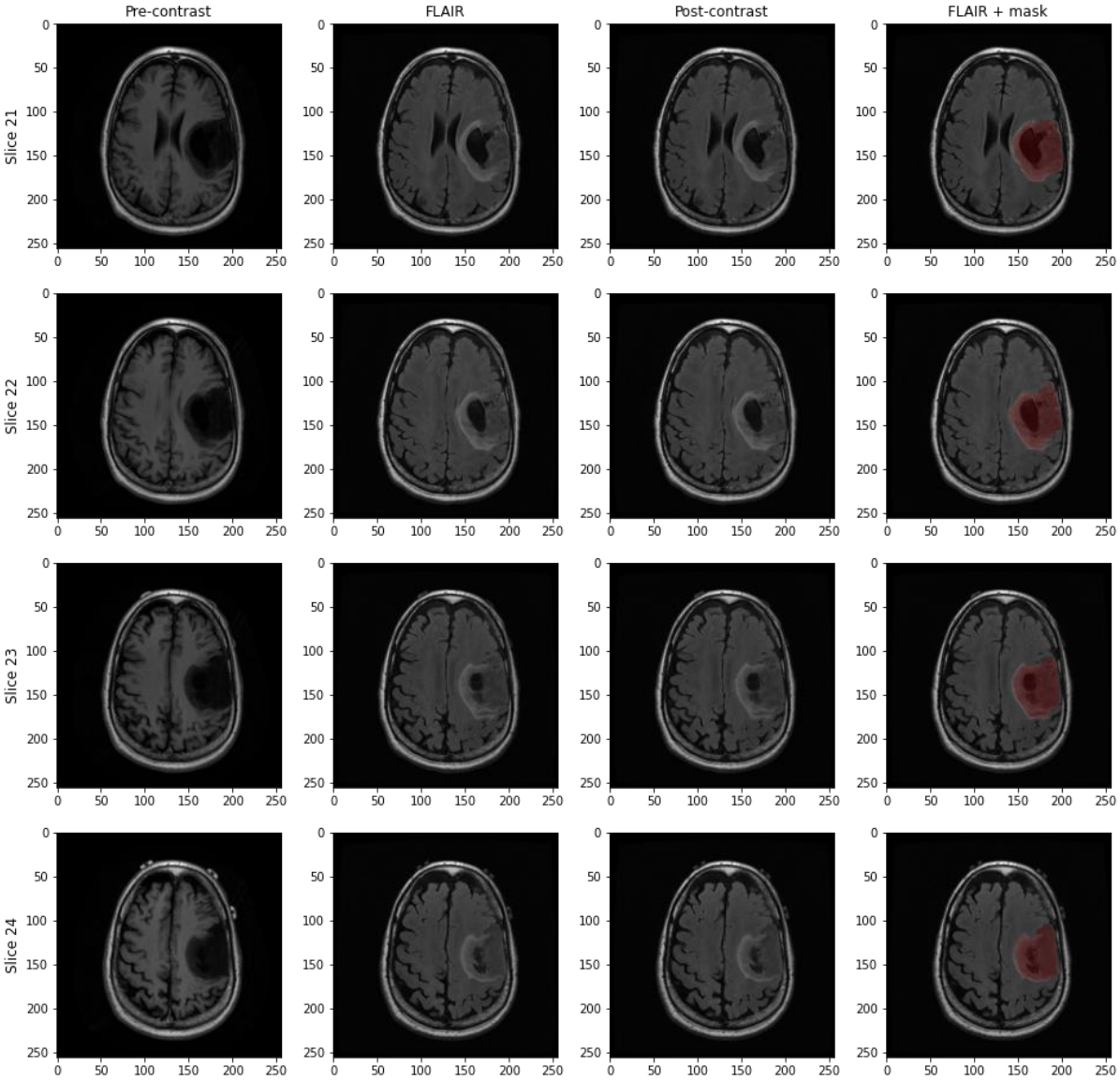


Figure 2 – Example MRI slices consisting of three input channels.

Solution statement

The UNet (Ronneberger et al., 2015) architecture has been successfully utilized for a wide range of segmentation tasks. I will use this as the first network architecture in my tests. A Bayesian hyperparameter search will be run in in SageMaker to determine the optimum hyperparameter configuration (although I will only run a limited search for the sake of time and costs). This will serve as my baseline model. I will then test a newer segmentation network to see if they are able to improve the results. Some options for my second network include:

- Attention UNet: The addition of the attention mechanism to the original UNet architecture can help the network learn where to focus in an image, improving performance.
- Vision Transformer (ViT): Transformer models have produced state-of-the-art results in many areas and are now widely utilized in vision problems. The small dataset size might be more of an issue for this type of model though.

Benchmark model

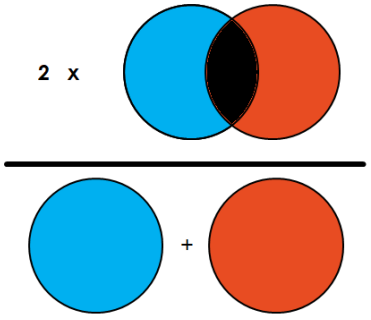
The main objective of this project is to build an end-to-end machine learning solution in AWS SageMaker. While I will attempt to produce the best result possible, it is not the goal of the project to produce a superior model to what others have already found. Once the main workflow is established, a wider/longer hyperparameter search can be run and other model architectures tested. The hard part is developing this pipeline.

Many people on Kaggle report a dice score of > 90%. However, the examples I've reviewed randomly separate split the image slices into training, validation and test dataset. This means that 2D slices from the same patient are included in both the training, validation and test datasets. Adjacent slices from the same patient can be very similar, so I feel this is a form of data leakage. In my solution, I ensure that the data from one patient is only in one dataset.

The standard model that often performs very well on segmentation tasks is the UNet (Ronneberger et al., 2015). I will use this as my first model, and then compare the results to a more recent model architecture, such as Attention UNet or ViT. I'll be trying to make sure that my model achieves a Dice score of at least 0.7, but hopefully it will be higher than this.

Evaluation metrics

The performance of the model will be measured using the Dice score (or coefficient), which measures the similarity between two sets of data (in this case the predicted segmentation mask and the manually labelled mask). The Dice score is illustrated in Figure 3. Put simply, it is two times the area of overlap between the two images divided by the total number of pixels in the images.



$$\frac{2*|X \cap Y|}{|X|+|Y|}$$

Figure 3 – Illustration of the Dice score (Source)

The Dice coefficient is very similar to the Intersection over Union (IoU). Scores ranges from 0 to 1, with 1 meaning perfect prediction. The use of pixel accuracy will not be a good indicator of performance in this case due to the class imbalance of the data (i.e., the model could predict all pixels to be the negative class and it would achieve a very high pixel accuracy score).

Project design

I'll be making use of the following tools for my proposed solution:

- MONAI: Open-source Python package for machine learning with medical datasets. This will be used for image loading, transformations and building deep learning models.
- PyTorch Lightning: Python package to help organize and productionize PyTorch code. It also makes it easier to implement best practices.
- Weights and Biases: This will be used for experiment tracking and possibly for model management.

I'll be making use of tools within MONAI and PyTorch Lightning to try to make the code run as fast as possible on GPU. This includes:

- Using 16-bit floating point training
- Moving all the data to GPU (since the dataset is small)

As part of my solution, I intend to give various options to run the code. This includes:

- Fully local implementation of SageMaker where the model is trained and deployed on a local machine (using Docker)
- Notebook (or script) running from a local machine but submitting training job and deployment to AWS.
- Running fully on AWS SageMaker.

If I have time at the end, I'll try to create my own customized container on AWS Elastic Container Registry (ECR) for this project.