INFO8010: Brain Cancer Detection Project Report

Hiba Qouiqa,[1] **Ramzan Arsanov**,[2] and **Ayoub Assaoud**[3]

[1] *hiba.qouiqa@student.uliege.be (s2304031)*
[2] *r.arsanov@student.uliege.be (s194447)*
[3] *ayoub.assaoud@student.uliege.be (s207227)*

I. ABSTRACT

In the context of our Deep Learning course and to apply and reinforce theoretical concepts, we selected the detection of brain cancer as our project topic. We explored and compared two state-of-the-art deep learning architectures a custom Convolutional Neural Network (CNN) and a Vision Transformer (ViT)—for automated classification of glioma, menin, and brain tumor MRI scans of the Bangladesh Brain Cancer MRI Dataset [1]. of 6,056 scans. Our pipeline included ..... This study not only reinforces practical understanding of deep learning techniques but also demonstrates their potential to support rapid, reliable brain tumor diagnosis in clinical settings.

II. INTRODUCTION

We wanted to tackle the problem of detecting Brain cancers like gliomas, meningiomas, and tumors that can have serious consequences if they aren't caught early, yet reading dozens of MRI scans by hand is both time-consuming and prone to variation between radiologists.

That's why we turned to convolutional neural networks and transformers—two powerful tools we've been studying—so we could build a system that learns directly from MRI images. Our goal was simple: compare a custom-built CNN with a ViT to see which architecture is best suited for this classification task.

Through this project, we:

- Collected and preprocessed over 6,000 MRI scans, applying normalization, resizing, and various augmentations to make the data robust.
- Designed and trained two distinct deep learning models, experimenting with .....
- Evaluated performance using ....

In the following sections, we'll walk through related work, our data pipeline, model details, and results.

I think i will make a pipeline scheme and put it here !!!!

III. RELATED WORK

IV. METHODS

A. Data

1. Dataset Description

We use the Bangladesh Brain Cancer MRI Dataset[1] is a comprehensive collection of MRI images categorized into three distinct classes:

- Brain_Glioma: 2004 images
- Brain_Menin: 2004 images
- Brain_Tumor: 2048 images

The dataset includes a total of 6056 images, uniformly resized to 512x512 pixels. These images were collected from various hospitals across Bangladesh with the direct involvement of experienced medical professionals to ensure accuracy and relevance.

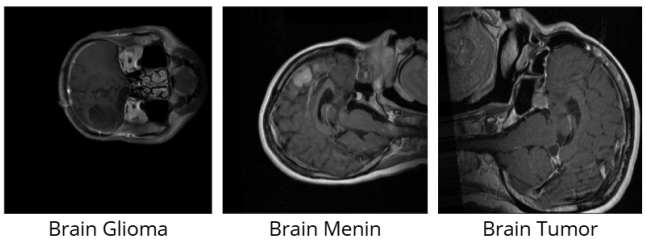

FIG. 1. A sample of MRI images from the brain tumor dataset.

2. Data Exploration:

Before going through preprocessing, we visualized the raw pixel-intensity distributions for each tumor class and discovered a clear brightness bias: glioma images were generally darker, menin images intermediate, and tumor images noticeably brighter (Figure 2). This problem risks the model learning to guess classes solely and stupidly by overall brightness rather than meaningful anatomical

patterns. To address this, we applied Contrast-Limited Adaptive Histogram Equalization (CLAHE) to flatten large-scale brightness differences while preserving local tissue to force the network to focus on true morphological features.

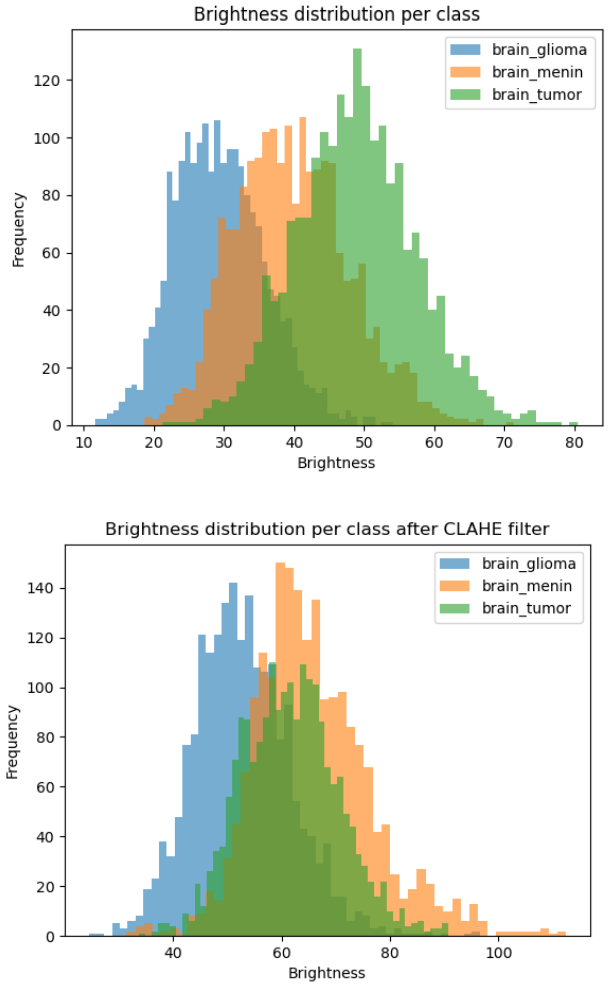

FIG. 2. Brightness histograms for glioma (blue), menin (orange), and tumor (green) scans before and after contrast enhancement.

3. *Data Preprocessing*

Our preprocessing pipeline was implemented in Python using OpenCV, TorchVision, and scikit-learn, and included the following steps:

- Contrast Enhancement and Brightness equalization: We applied Contrast Limited Adaptive Histogram Equalization (CLAHE) on grayscale MRI images to enhance local contrast and highlight tissue structures.
- Foreground Cropping: we generated a binary mask to distinguish brain tissue from the background. We computed the bounding box of nonzero mask regions, cropped the image accordingly, and reduced background noise.
- Resizing: images were resized to 256×256 pixels.
- Normalization : Pixel intensities were scaled to the [0,1] range by dividing by 255.
- Data Augmentation: To increase dataset diversity and reduce overfitting, we generated augmented tensors by applying horizontal flips, vertical flips, and rotations (90° and 270°) to each image, no translating was applied as the image is already cropped to the limit of ROI.
- Dataset Splitting: The processed images were stacked on PyTorch tensors and split using train_test_split: 80% for training, 10% for validation, and 10% for testing.
- Saving Processed Images: Processed tensor images were saved as JPG files organized into train, val, and test directories by class labels.

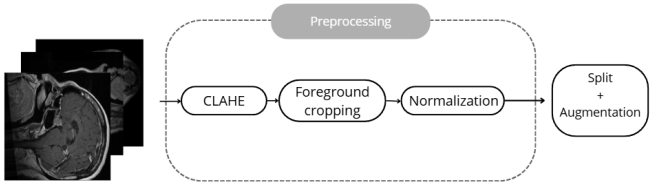

FIG. 3. Preprocessing pipeline

B. Architecture

1. *CNN*

Our CNN architecture is composed of four convolutional blocks, each consisting of a convolutional layer, batch normalization, ReLU activation, dropout, and max pooling. The number of channels increases in deeper layers (64, 128, 256, 512). After the convolutional feature extractor, the output is flattened and passed through two fully connected layers with dropout and ReLU, ending with a final classification layer. This design enables the model to extract hierarchical features from MRI images and perform robust classification. The architecture is implemented in the `CNNModel` class and is optimized for small medical datasets, with aggressive data augmentation to improve generalization.

$fc\_dropout$ is used to prevent overfitting at dense layer, and the model is trained using the AdamW optimizer with a learning rate of $1e-4$. The batch size is set to 16, and the model is trained for 25 epochs.

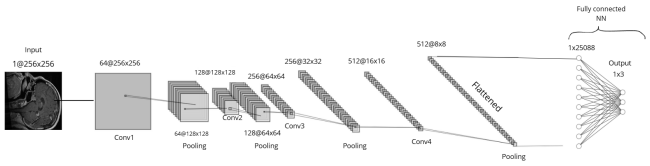

FIG. 4. Architecture of the CNN model.

2. *ViT - Reference Book*

The reference Vision Transformer (ViT) that was explained by 'Dive into deep learning [2] follows the standard design:

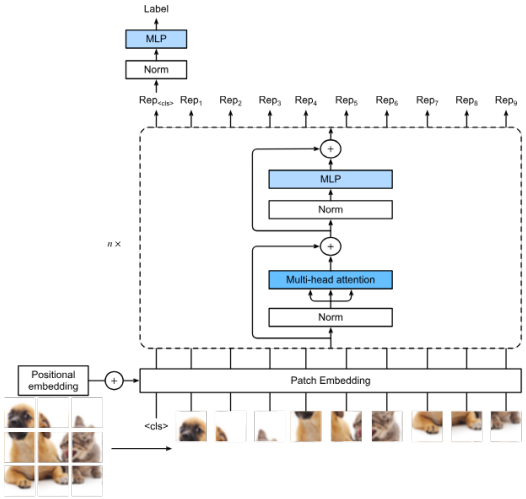

FIG. 5. The vision Transformer architecture from reference book

- **Positional Encoding:** Since the Transformer architecture does not have any inherent notion of the order of the input sequence, positional encodings are added to the patch embeddings to provide information about the position of each patch in the original image.

- **[CLS] Token:** A learnable `[CLS]` token is prepended to the sequence of patch embeddings. This token is used for classification tasks, and its final representation after processing through the Transformer blocks is used as the input to the classification head.

- **Patch Embedding:** The input image is divided into non-overlapping patches, which are then linearly embedded into a sequence of vectors. This is done using the same convolutional layer, to prevent the model from making assemptions about the real order of patches even with data augmentation, the kernel size is equal to the patch size and stride is equal to the patch size. The final shape:

  $$(\#batches, \#patches + 1, \#output\_channels)$$

  with: $\#patches = \frac{img\_H}{patch\_H} \frac{img\_W}{patch\_W} + 1$

- **Transformer Encoder Blocks:** The core of the ViT consists of sequence of Transformer encoder blocks, called **ViTBlock**, each containing parallel Multi-Head Masked Self-Attention (MHMSA) and feed-forward neural network (MLP) layers.[3] [4] [5]

  Residual connections and layer normalization are applied around each sub-layer.

- **Classification Head:** The classification is performed using the output corresponding to the `[CLS]` token final representation. This implementation closely follows Dosovitskiy et al. (2021) [4].

3. *ViT - different FCs for each head*

What was a bit confusing is the fact that reference book, did not mention the fact that the same FCs projections (W^Q, W^K, W^V) are used for all heads, which is not the case in the original paper [4].

Thus, we modified the structure of heads so that each attention $head_i$ has its own set of fully connected projection weights (W_i^Q, W_i^K, W_i^V) for queries, keys, and values, instead of sharing weights across heads.

This increases the representational capacity of the model and allows each head to learn distinct features. The rest of the architecture remains similar to the reference ViT, with patch embedding, positional encoding, and stacked Transformer blocks.

4. *ViT - no cls*

the additional 'token' $< cls >$ in this ViT was subject to many discusions for AI community, its utility is indeed not that obvious.

Thus we removed the `[CLS]` token. Instead, the model uses only the patch representations, and the output for classification is taken from the first patch token after the Transformer encoder.

The approach aims to test the impact of the `[CLS]` token for our case, as well as exploring possible generalization.

5. *ViT - cls + MLP over Reps*

To leverage information from all patches, this variant concatenates all token representations (including `[CLS]`) after the Transformer encoder and passes the resulting vector through a two-layer MLP for classification This allows the model to aggregate information from the entire

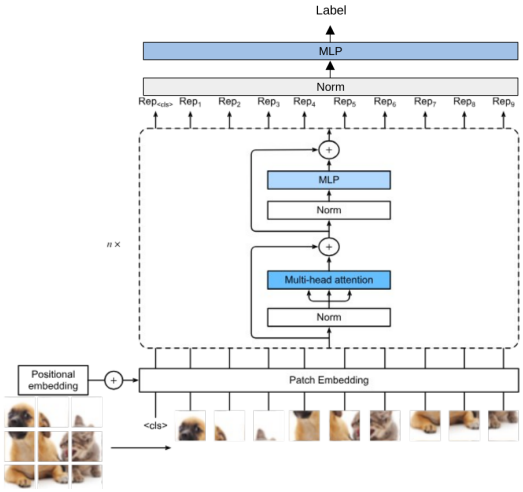

FIG. 6. The architecture of ViT with MLP over representations

image, potentially capturing more global context than using only the [CLS] token.

By taking only one representation to infer the class, we may lose some useful information, therefore we connected all representations to a MLP of two layers, first one contains $num_hiddens$ neurons and the second one is the output layer with $num_classes$ neurons. the choice of the number of neurons in the first layer is arbitrary, but we found that it works well for our case.

C. Training

1. Parameters

TABLE I. Shared parameters between all models.

Parameter	Value
Learning rate	1e-4
Optimizer	AdamW
Batch size	16
Image size	256
Loss function	CrossEntropy
Epochs	25
Patch size (ViT)	16
#hiddens (ViT)	516
MLP hidden dim	156
Weights decay	1e-4
Use bias	false
FC_Dropout	0.3
Dropout	0.2
Embedding Dropout	0.2

for ViT - cls MLP over Reps, we choosed to use 12 heads and 6 blocks, because the additional MLP on top could lead to overfitting, we found that it works well for our case which sounds a good trade-off between performance and computational cost as well.

TABLE II. Parameters used for each model.

Parameter	ViT (reference book)	ViT (CLS head)	ViT (no CLS head)	ViT (MLP over rep)
#heads	8	8	8	12
#Blocks	12	12	12	6

2. Training Procedure

In order to get close to appropriate learning parameters, as well as some reasonable hyper-parameters, we performed a grid search for the basic architecture in section IV B 2 over the following parameters: Learning rate: $[0.0001, 0.0005, 0.0012]$, Batch size: $[16, 32]$, no. of hidden: $[126, 516]$, no. of heads: $[4, 8, 12]$ and blocks: $[6, 12]$. then after some additional trials, we decided to use the parameters in the table II

3. Optimizer

We used the AdamW optimize, an extension of the Adam optimizer with weight decay. AdamW helps to prevent overfitting by adding a weight decay term to the loss function, which encourages the model to learn simpler representations.

The decision to use AdamW was based on its effectiveness in training deep learning models, especially in the context of image classification tasks. we kept the default parameters for AdamW, which are:

β_1	β_2	ϵ	weight decay	learning rate
0.9	0.999	$1e-8$	$1e-4$	$1e-4$

TABLE III. Optimizer Parameters

4. Loss Function

the loss function used for training the models is the Cross-Entropy loss, which is commonly used for multi-class classification tasks. The Cross-Entropy loss is defined as:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{3} y_{i,j}\log(p_{i,j})$$

$y_{i,j}$ is 1 if sample i belongs to class j, otherwise 0
$p_{i,j}$ is the predicted probability for class j.

Thus even if the model predicts a probability of 0.8 for the correct class and 0.2 for the other classes, it will be penalized for this low confidence, but the loss will be low accordingly.

V. RESULTS

As can be seen in the figure 7, the CNN model outperforms the ViT models in terms of both training and validation accuracies, and converges faster. The CNN achieves a maximum

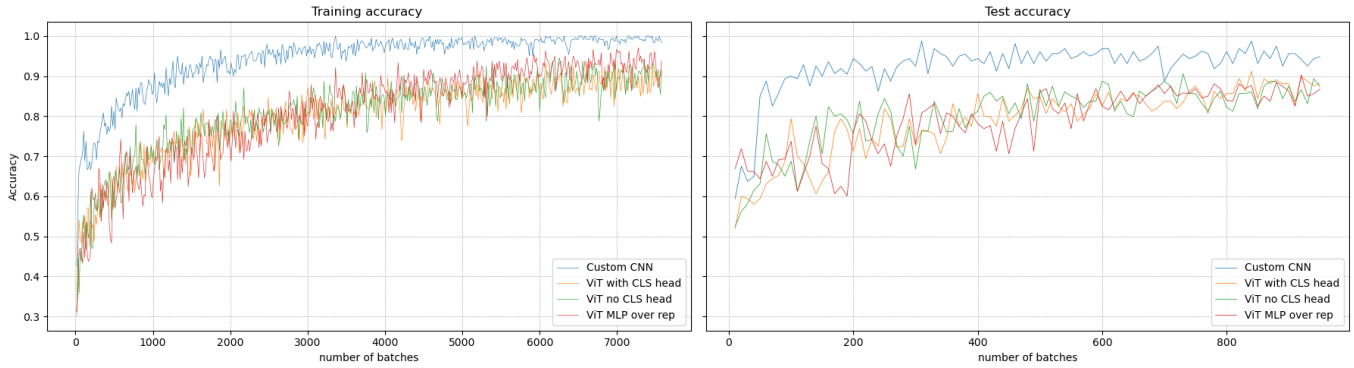

FIG. 7. Evolution of Training and test accuracies for each model.

accuracy of 95.5% in both, before 25 epochs, while the ViT models reaches a maximum accuracy of around 90% after 25 epochs.

the ViT models show a more gradual increase in accuracy, with the MLP over representations model achieving the highest validation accuracy of 90.5% after epoch 35 (in other test), it was the slowest model, but confidently surpassing other ViT variante, which may indicate a room for improvement for larger datasets . The no CLS head model performs slightly worse, with a maximum validation accuracy of 89.5%. The CLS head model achieves a maximum validation accuracy of 89% at epoch 25.

VI. DISCUSSION

A. The failure of ViT models to beat CNN

Althought the tremendous potential of Attenstion mechanisms, CNN remains competitive for relatively small datasets, as can be seen in the losses plot, CNN converges faster that ViT and achieves better accuracies.

this can be due to the fact that the complexity of the problem is not that high, and the dataset that remains small against some starndard datasets (that can exceed 300 million images), thus, ViT transformer was not able to learn meaningful representations from all patches, or haven't enough training data to generalize, even with image augmentation. It also shows a more gradual increase in accuracy, with the MLP over representations model achieving the highest validation accuracy of 90.5% after epoch 35 (in other test), so it was the slowest model, but confidently surpassing other ViT variants, which may indicate a room for improvements on larger datasets .

B. The importance of cls token

The question of the importance of cls token has been repeated many times as its role was not that obvious, a question from github repository from google-research titled "Is the extra class embedding important to predict the results, why not simply use feature maps to predict?", for more details see Google-search github issues

An answer to that question was from the belgian reasearcher Lucas Beyer from Google lab, has confirmed that it is not really important from the performance point of view, but it is important from the design point of view, as they wanted the model to be "exactly Transformer, but on image patches", so they kept this design from Transformer, where a token is always used.

Indeed, in our results, the model with no cls head performed almost in the same way, in terms of accuracy and learning speed with the one with cls head.

C. The importance of

We couldn't demonstrate the full potential of Vit - MLP over representations, in our case, as it performed almost in the same way with the one with cls head, which was not not expected. This may be due to the fact that the dataset is relatively small or the complexity of the problem, and the model is not able to learn meaningful representations from all patches.

D. The importance of cropping images

the impact of cropping surrounding black space: as many other works didn't considered cropping images, such as in kaggle, we found that this step increased the test accuracy by five points to hit 95.5% for CNN as an example, which was expected as the model was able to focus on the region of interest (ROI) and ignore the background noise. This is particularly important in medical imaging, where the ROI may be small compared to the overall image size, thus reducing unnecessary artifacts.

E. About possible improvements

While our models achived relativly high accuracies, there are still improvments to be made in both performance and generalization.

- The ViT models were trained from scratch. However, as previously mentioned, these models require a larger dataset to converge well. We likely could have obtain

better performance using pretrained weights from models trained on large medical datasets.

- We applied basic augmentations such as flips and rotations. However, more advanced techniques such as random deformations or noise injections could help the models generalize better.
- Our dataset is limited because there's no metadata such as age, gender, location of the tumor, etc. Those metadada integreted with images could improve accuracy.
- We could have also used ensemble methods by combining predictions from our models by voting. It would likely improve generalization, especially for ambiguous cases.
- One simple improvement is to add a segmentation step : first run a U-Net to get a tumor mask, crop the MRI to that region, then feed only the tumor patch into our CNN/ViT. This way the classifier sees just the lesion—no distracting background—which usually boosts accuracy. Plus, the segmentation maps themselves give a clear visual explanation for why the model called it glioma, meningioma or pituitary tumor

VII. REFERENCES

[1] Orvile, "Brain cancer mri dataset," https://www.kaggle.com/datasets/orvile/brain-cancer-mri-dataset, 2020.

[2] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning, section 11.8. transformers for vision," https://d2l.ai/chapter_attention-mechanisms-and-transformers/vision-transformer.html, 2020.

[3] A. Baevski and M. Auli, "Adaptive input representations for neural language modeling," *International Conference on Learning Representations*, 2018. [Online]. Available: https://arxiv.org/abs/1809.10853

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, and et al., "An image is worth 16 x 16 words: transformers for image recognition at scale," https://arxiv.org/abs/2010.11929, 2021.

[5] Q. Wang, B. Li, T. Xiao, J. Zhu, C. Li, D. Wong, and L. Chao, "Learning deep transformer models for machine translation," *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1810–1822, 2019. [Online]. Available: https://arxiv.org/abs/1906.05237