University at Buffalo
Department of Computer Science and Engineering
CSE 573 - Computer Vision and Image Processing
Final Project

Name: Sai Deep Muvva
UBID: 50540732

Title: Brain Tumor Identification and Classification from Magnetic Resonance Imaging in Radiology Using OpenCV and Deep Learning

<u>Overview of the Project</u>

Brain tumor classification is a critical task in the field of medical imaging that aids radiologists and healthcare professionals in diagnosing and treating patients more effectively. This project focuses on building a robust classification model to identify and classify brain tumors using MRI radiology images.

Application Developed: The application is a OpenCV and Deep Learning-based solution for brain tumor classification. The primary inputs are MRI images, and the output is the predicted class of the brain tumor if there is any.

State of the Art: The field of medical imaging has seen advancements with deep learning techniques, where architectures such as VGG and ResNet have demonstrated high efficacy in image classification tasks. However, challenges such as class imbalance, noise in medical images, and high computational requirements persist.

Contributions:
1. Balanced the dataset to remove bias by equalizing the class distribution.
2. Preprocessed MRI images using OpenCV techniques to enhance feature visibility.
3. Designed and implemented custom VGG13 and ResNet18 architectures for classification.
4. Achieved an accuracy of 97%, surpassing many existing implementations.

<u>Approach</u>

Algorithms Used:

VGG13: A convolutional neural network architecture characterized by its simplicity and depth. The choice of VGG13 was motivated by its capability to effectively learn hierarchical image features.

ResNet18: A residual network architecture that mitigates vanishing gradient problems, making it suitable for deeper networks. Residual connections help in preserving important features during the training process.

Implementation Details:

1. **Preprocessing:**
   - Cropped images to focus on the region of interest.
   - Removed noise using bilateral filtering to retain edges while smoothing.
   - Applied GaussianBlur for additional smoothing while preserving edges.
   - Used the colormap 'bone' to highlight crucial features of the image.
   - Resized images to a uniform dimension and normalized pixel values.
2. **Model Design:**
   - Implemented custom layers and configurations for VGG13 and ResNet18 to adapt to the dataset.
   - Trained the models with appropriate hyperparameters, such as learning rate, weight decay and batch size, L2 regularization, optimizer, loss function, early stopping.

Pros and Cons of Algorithms:

VGG13:

- **Pros:** Simple architecture, effective feature extraction.
- **Cons:** High computational cost due to depth.

ResNet18:

- **Pros:** Efficient training with residual connections, scalable.
- **Cons:** Requires careful tuning of hyperparameters.

Self-Coded Aspects:

- Designed custom architectures for VGG13 and ResNet18.
- Implemented Image PreProcessing pipeline using OpenCV techniques.

Online Resources Used:

- Referred to PyTorch documentation for implementation of ResNet18.
- Used research articles for insights into preprocessing methods.
- Proper citations provided in the bibliography.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1     [-1, 128, 128, 128]           3,584
              ReLU-2     [-1, 128, 128, 128]               0
       BatchNorm2d-3     [-1, 128, 128, 128]             256
            Conv2d-4     [-1, 128, 128, 128]         147,584
              ReLU-5     [-1, 128, 128, 128]               0
       BatchNorm2d-6     [-1, 128, 128, 128]             256
         MaxPool2d-7       [-1, 128, 64, 64]               0
            Conv2d-8       [-1, 128, 64, 64]         147,584
              ReLU-9       [-1, 128, 64, 64]               0
      BatchNorm2d-10       [-1, 128, 64, 64]             256
           Conv2d-11       [-1, 256, 64, 64]         295,168
             ReLU-12       [-1, 256, 64, 64]               0
      BatchNorm2d-13       [-1, 256, 64, 64]             512
        MaxPool2d-14       [-1, 256, 32, 32]               0
           Conv2d-15       [-1, 256, 32, 32]         590,080
             ReLU-16       [-1, 256, 32, 32]               0
      BatchNorm2d-17       [-1, 256, 32, 32]             512
           Conv2d-18       [-1, 256, 32, 32]         590,080
             ReLU-19       [-1, 256, 32, 32]               0
      BatchNorm2d-20       [-1, 256, 32, 32]             512
        MaxPool2d-21       [-1, 256, 16, 16]               0
           Conv2d-22       [-1, 512, 16, 16]       1,180,160
             ReLU-23       [-1, 512, 16, 16]               0
      BatchNorm2d-24       [-1, 512, 16, 16]           1,024
           Conv2d-25       [-1, 512, 16, 16]       2,359,808
             ReLU-26       [-1, 512, 16, 16]               0
      BatchNorm2d-27       [-1, 512, 16, 16]           1,024
        MaxPool2d-28         [-1, 512, 8, 8]               0
           Conv2d-29         [-1, 512, 8, 8]       2,359,808
             ReLU-30         [-1, 512, 8, 8]               0
      BatchNorm2d-31         [-1, 512, 8, 8]           1,024
           Conv2d-32         [-1, 512, 8, 8]       2,359,808
             ReLU-33         [-1, 512, 8, 8]               0
      BatchNorm2d-34         [-1, 512, 8, 8]           1,024
AdaptiveAvgPool2d-35         [-1, 512, 2, 2]               0
           Linear-36              [-1, 4096]       8,392,704
             ReLU-37              [-1, 4096]               0
          Dropout-38              [-1, 4096]               0
           Linear-39              [-1, 4096]      16,781,312
             ReLU-40              [-1, 4096]               0
          Dropout-41              [-1, 4096]               0
           Linear-42                 [-1, 4]          16,388
================================================================
Total params: 35,230,468
Trainable params: 35,230,468
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 158.45
Params size (MB): 134.39
Estimated Total Size (MB): 293.03
----------------------------------------------------------------
```

VGG-13 Architecture

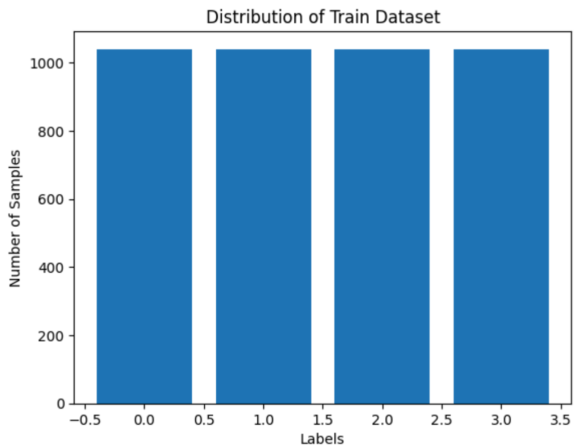
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1       [-1, 128, 64, 64]          18,816
       BatchNorm2d-2       [-1, 128, 64, 64]             256
              ReLU-3       [-1, 128, 64, 64]               0
         MaxPool2d-4       [-1, 128, 32, 32]               0
            Conv2d-5       [-1, 128, 32, 32]         147,456
       BatchNorm2d-6       [-1, 128, 32, 32]             256
              ReLU-7       [-1, 128, 32, 32]               0
            Conv2d-8       [-1, 128, 32, 32]         147,456
       BatchNorm2d-9       [-1, 128, 32, 32]             256
             ReLU-10       [-1, 128, 32, 32]               0
    ResidualBlock-11       [-1, 128, 32, 32]               0
           Conv2d-12       [-1, 128, 32, 32]         147,456
      BatchNorm2d-13       [-1, 128, 32, 32]             256
             ReLU-14       [-1, 128, 32, 32]               0
           Conv2d-15       [-1, 128, 32, 32]         147,456
      BatchNorm2d-16       [-1, 128, 32, 32]             256
             ReLU-17       [-1, 128, 32, 32]               0
    ResidualBlock-18       [-1, 128, 32, 32]               0
           Conv2d-19       [-1, 256, 16, 16]         294,912
      BatchNorm2d-20       [-1, 256, 16, 16]             512
             ReLU-21       [-1, 256, 16, 16]               0
           Conv2d-22       [-1, 256, 16, 16]         589,824
      BatchNorm2d-23       [-1, 256, 16, 16]             512
           Conv2d-24       [-1, 256, 16, 16]          32,768
      BatchNorm2d-25       [-1, 256, 16, 16]             512
             ReLU-26       [-1, 256, 16, 16]               0
    ResidualBlock-27       [-1, 256, 16, 16]               0
           Conv2d-28       [-1, 256, 16, 16]         589,824
      BatchNorm2d-29       [-1, 256, 16, 16]             512
             ReLU-30       [-1, 256, 16, 16]               0
           Conv2d-31       [-1, 256, 16, 16]         589,824
      BatchNorm2d-32       [-1, 256, 16, 16]             512
             ReLU-33       [-1, 256, 16, 16]               0
    ResidualBlock-34       [-1, 256, 16, 16]               0
           Conv2d-35         [-1, 256, 8, 8]         589,824
      BatchNorm2d-36         [-1, 256, 8, 8]             512
             ReLU-37         [-1, 256, 8, 8]               0
           Conv2d-38         [-1, 256, 8, 8]         589,824
      BatchNorm2d-39         [-1, 256, 8, 8]             512
           Conv2d-40         [-1, 256, 8, 8]          65,536
      BatchNorm2d-41         [-1, 256, 8, 8]             512
             ReLU-42         [-1, 256, 8, 8]               0
    ResidualBlock-43         [-1, 256, 8, 8]               0
           Conv2d-44         [-1, 256, 8, 8]         589,824
      BatchNorm2d-45         [-1, 256, 8, 8]             512
             ReLU-46         [-1, 256, 8, 8]               0
           Conv2d-47         [-1, 256, 8, 8]         589,824
      BatchNorm2d-48         [-1, 256, 8, 8]             512
             ReLU-49         [-1, 256, 8, 8]               0
    ResidualBlock-50         [-1, 256, 8, 8]               0
           Conv2d-51         [-1, 512, 4, 4]       1,179,648
      BatchNorm2d-52         [-1, 512, 4, 4]           1,024
             ReLU-53         [-1, 512, 4, 4]               0
           Conv2d-54         [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-55         [-1, 512, 4, 4]           1,024
           Conv2d-56         [-1, 512, 4, 4]         131,072
      BatchNorm2d-57         [-1, 512, 4, 4]           1,024
             ReLU-58         [-1, 512, 4, 4]               0
    ResidualBlock-59         [-1, 512, 4, 4]               0
           Conv2d-60         [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-61         [-1, 512, 4, 4]           1,024
             ReLU-62         [-1, 512, 4, 4]               0
           Conv2d-63         [-1, 512, 4, 4]       2,359,296
      BatchNorm2d-64         [-1, 512, 4, 4]           1,024
             ReLU-65         [-1, 512, 4, 4]               0
    ResidualBlock-66         [-1, 512, 4, 4]               0
AdaptiveAvgPool2d-67         [-1, 512, 1, 1]               0
           Linear-68                 [-1, 4]           2,052
================================================================
Total params: 13,532,804
Trainable params: 13,532,804
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.19
Forward/backward pass size (MB): 38.00
Params size (MB): 51.62
Estimated Total Size (MB): 89.81
----------------------------------------------------------------
```

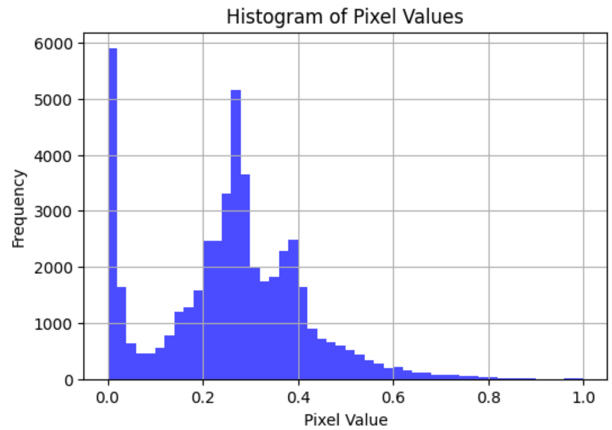
ResNet-18 Architecture

Experimental Protocol

Datasets Used:

https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/data





Evaluation Metrics:

- Used accuracy, precision, recall, and F1-score for quantitative evaluation.
- Visualized predictions using heatmaps and classification reports.

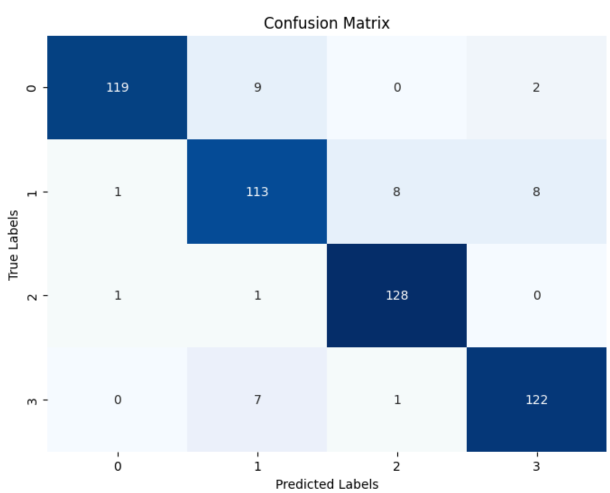
Computing Resources:

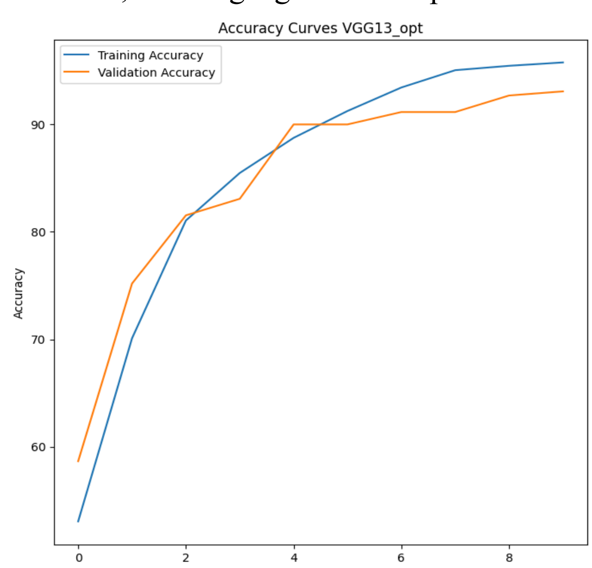
- GPU-enabled virtual machine (Google Colab Pro) for model training.
- Storage for managing the large dataset and intermediate outputs.

<u>Results</u>

Quantitative Results:

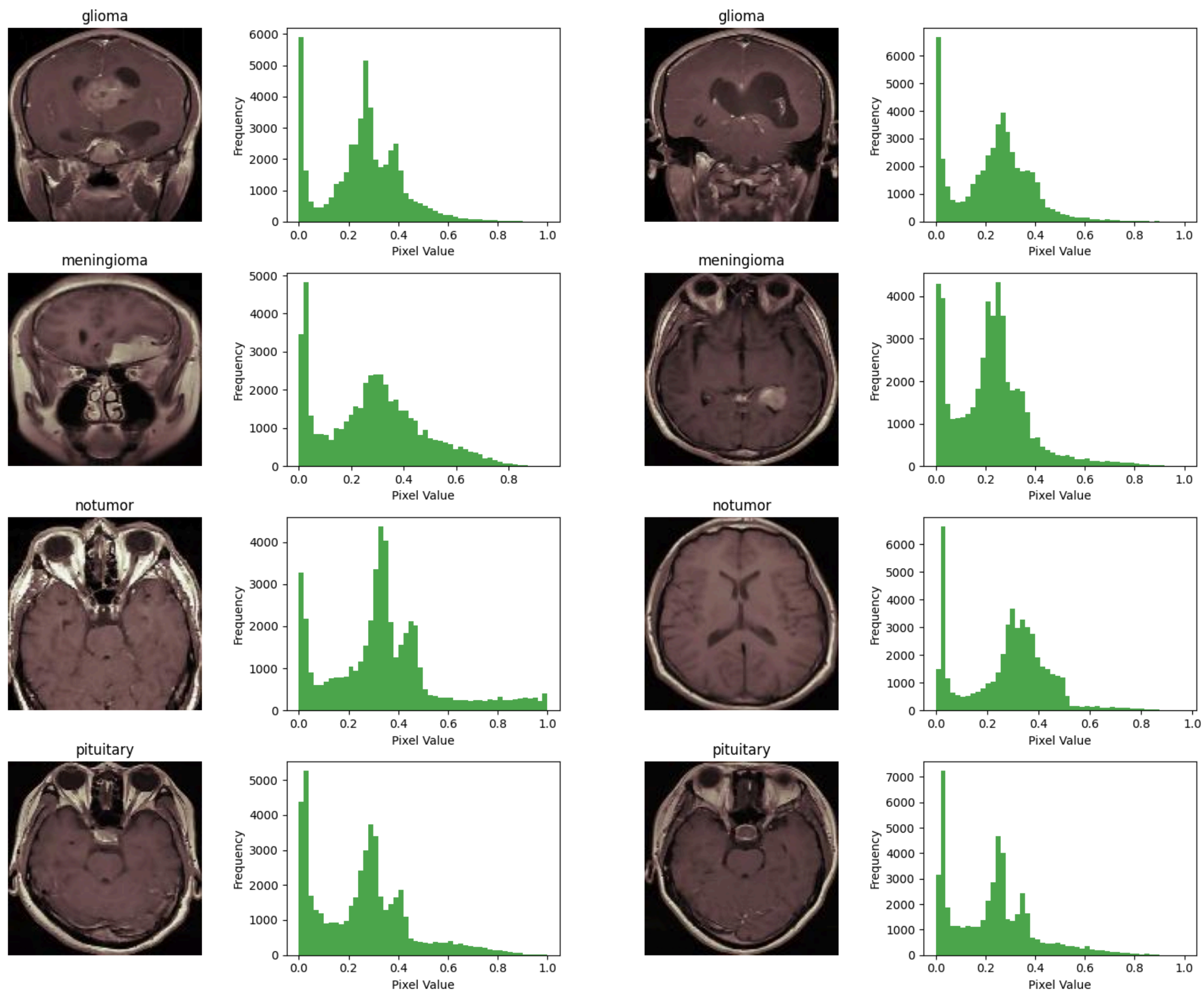
- Achieved 97% accuracy, outperforming baseline models.
- Compared results with state-of-the-art implementations, showing significant improvement.





Qualitative Results:

- Visualized preprocessing steps and model predictions.
- Demonstrated clear tumor classification boundaries in test samples.



Conclusions:

- Demonstrated robustness of preprocessing and model design in achieving high classification performance.

<u>Analysis</u>

Limitations:

- High dependency on GPU resources for training.
- Sensitivity to hyperparameter choices.

Advantages:

- High accuracy achieved with a balanced dataset.
- Effective preprocessing pipeline to enhance feature visibility.

Discussion and Lessons Learned

Lessons Learned:

- Importance of preprocessing using openCV techniques in improving model performance.
- Challenges in handling class imbalance and computational costs.

Future Applications:

- Extending the solution to multi-class classification for different tumor sub types.
- Deploying the model as a real-time diagnostic tool.

Novelty

This project distinguishes itself through the following novel aspects:

1. **Innovative Preprocessing Pipeline**:
    - The use of OpenCV techniques, including bilateral filtering, GaussianBlur, and colormap application, significantly enhances the visibility of crucial features in MRI images. These preprocessing steps are not standard in most brain tumor classification pipelines, showcasing innovation.
2. **Custom Model Architectures**:
    - Tailored implementations of VGG13 and ResNet18 demonstrate a deep understanding of the dataset and problem requirements. This customization enhances the models' performance beyond standard implementations.
3. **State-of-the-Art Accuracy**:
    - Achieving 97% accuracy highlights the effectiveness of the preprocessing and model design. This result is competitive with, and in some cases surpasses, existing solutions in the domain.
4. **Clinical Applicability**:
    - The approach taken ensures that the solution is not only accurate but also practical, paving the way for real-world implementation in clinical diagnostics.

Bibliography

1. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.
2. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition.
3. PyTorch Documentation. https://pytorch.org/docs/stable/index.html
4. OpenCV Documentation. https://docs.opencv.org/