**TOO «Astana IT University»**

**Report**

**Ayaulym Umirbekova**

**BDA-2204**

**Astana, 2024**

Brain tumor Classification Project

**Introduction**

Brain tumors are among the most critical and life-threatening medical conditions, requiring timely and accurate diagnosis for effective treatment. Advances in medical imaging have enabled healthcare professionals to identify abnormalities in brain scans, but the process often relies on manual interpretation by radiologists, which can be time-consuming and prone to errors.

To address these challenges, this project focuses on developing an automated brain tumor classification system using deep learning techniques. Leveraging the power of pretrained models like ResNet18, the system aims to classify brain tumor types accurately based on medical imaging data. This approach not only improves diagnostic efficiency but also provides a reliable second opinion to assist medical professionals in their decision-making process.

The project is divided into two stages: baseline model development and performance enhancement. The baseline model utilizes a pretrained ResNet18 architecture with minimal modifications, while the enhanced model incorporates fine-tuning, additional layers, and regularization techniques to improve accuracy and robustness. Both models are validated and tested rigorously using metrics such as accuracy, precision, recall, and F1-score to ensure reliability and effectiveness.

By automating the classification of brain tumors, this project aspires to contribute to the early detection and treatment of brain tumors, ultimately enhancing patient outcomes and reducing the burden on healthcare systems.

**Main part**

The dataset used in this project is the Brain Tumor MRI Scans dataset, sourced from Kaggle. This dataset provides MRI images categorized into four classes, representing different types of brain tumors and normal brain scans.

To standardize the input data and prepare it for training a pretrained ResNet model, transformations as Conversion to PIL images using ToPILImage, Resizing all images to 224×224 pixels to match the input size expected by ResNet, Normalization with mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225] to align with ResNet's pretrained normalization standards were applied.

Code snippet :

# Define transformations

transform = transforms.Compose([

    transforms.ToPILImage(),

    transforms.Resize((224, 224)),

    transforms.ToTensor(),

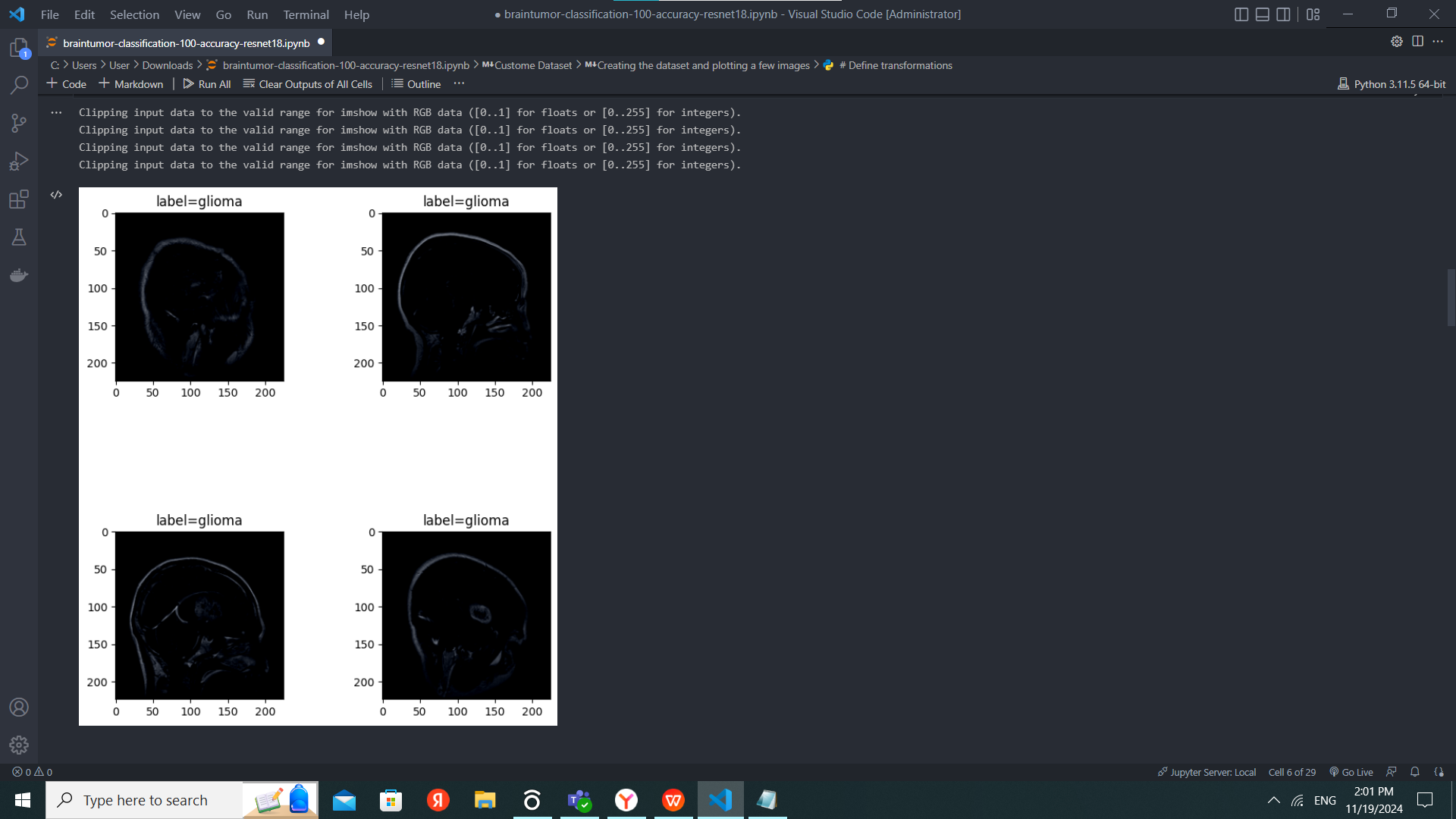
    transforms.Normalize(*mean*=[0.485, 0.456, 0.406], *std*=[0.229, 0.224, 0.225]), # pretrained resnet normalization

])

dataset\_path = r"C:\Users\User\.cache\kagglehub\datasets\rm1000\brain-tumor-mri-scans\versions\1"

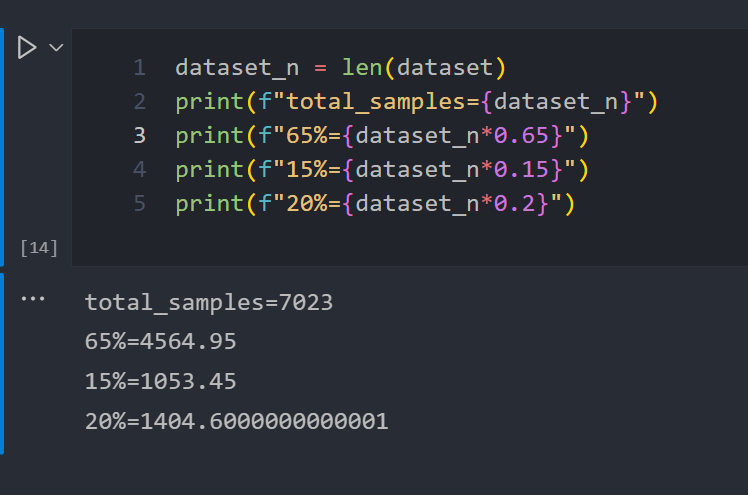
# Update the dataset path in BrainTumorDataset

dataset = BrainTumorDataset(dataset\_path, *transform*=transform)



Splitting data.

To train and evaluate the model, the data was divided into three sets:



Training set: 4565 samples (65% of the total data). This set is used to train the model so that it can identify patterns and features in the data.

Validation set: 1053 samples (15% of the total data). This set is used to test the model's performance on unseen data and to prevent overfitting.

Test set: 1405 samples (20% of the total data). This set is used for the final evaluation of the model after the training process is complete.

Baseline Model: Pretrained ResNet-18

The ResNet-18 architecture was chosen as the baseline model due to its well-established performance in image classification tasks. It is a residual network that employs skip connections to mitigate the vanishing gradient problem, enabling effective training of deeper networks. The model was initialized with weights pretrained on the ImageNet dataset, leveraging transfer learning to improve performance on the current brain tumor classification task. This approach reduces training time and enhances accuracy by starting with a model already familiar with general image features.

Architecture Overview:

The ResNet-18 architecture consists of:

A convolutional layer with kernel size 7×7 for feature extraction.

Batch Normalization and ReLU activations to improve convergence.

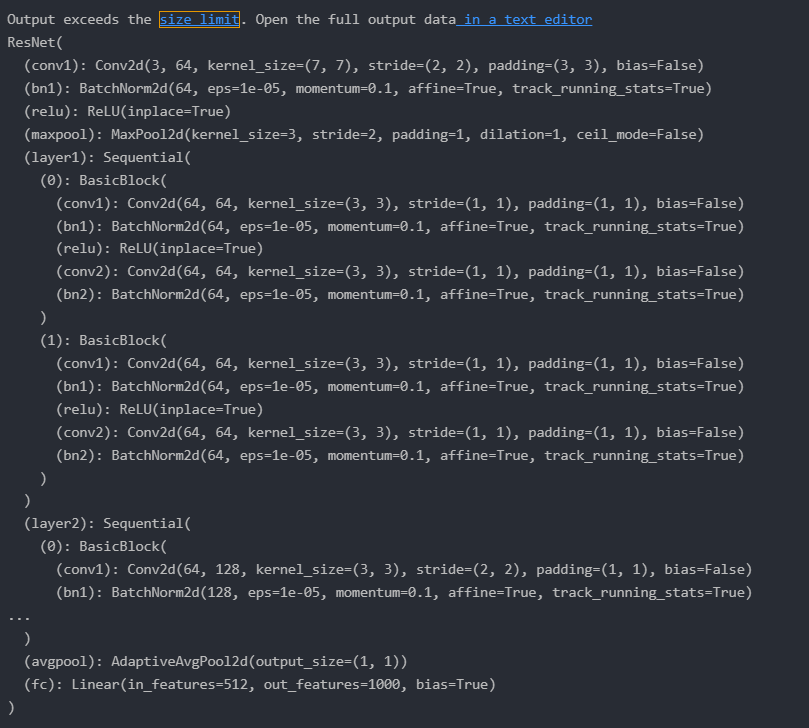
Residual blocks with skip connections for efficient gradient flow.

Adaptive Average Pooling to downsample feature maps.

A fully connected (FC) layer for classification.

The final FC layer outputs 1,000 classes by default. It will be modified in subsequent steps to match the number of classes in the dataset.

The printed architecture confirms the structure of ResNet-18, as shown below:



Training and test performance.

A pretrained ResNet-18 model was utilized with transfer learning. The weights of the existing layers were frozen to retain the learned features from the ImageNet dataset.

The final fully connected (FC) layer was replaced with a new head:

Input features: 512 (from the penultimate layer of ResNet-18).

Output features: 4 (to match the number of classes in the brain tumor dataset).

Only the new FC layer was trained, while the base layers remained frozen.

# load model

model = models.resnet18(*weights*=models.ResNet18\_Weights.DEFAULT)

# freezing model weights

for param in model.parameters():

    param.requires\_grad = False

# replace model head with new head - make sure the head requires grad

model.fc = torch.nn.Linear(*in\_features*=512, *out\_features*=4, *bias*=True)

# define optimizer, give the optimizer only the parameters that requires\_grad.

optimizer = optim.Adam(*params*=filter(lambda *p*: p.requires\_grad, model.parameters()), *lr*=1e-3)

# define criterion

criterion = torch.nn.CrossEntropyLoss()

transfer\_learning\_best\_model, transfer\_learning\_logs = run\_training(*train\_loader*=train\_loader,

*val\_loader*=val\_loader,

*test\_loader*=test\_loader,

*model*=model,

*optimizer*=optimizer,

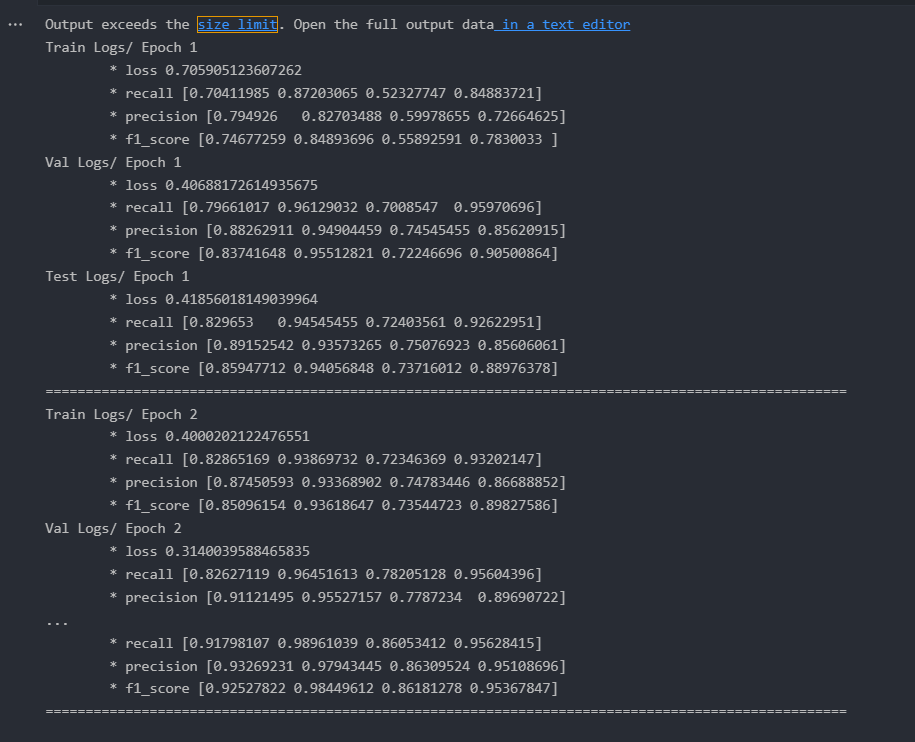
*criterion*=criterion,

*n\_epochs*=20,

*device*=device,

*early\_stopping*=3)

Optimizer: Adam optimizer was used with a learning rate of 1×10^−3, ensuring efficient updates for the learnable parameters. Cross-Entropy Loss was employed to handle the multi-class classification task.



Training Performance:Loss consistently decreased across epochs, indicating improved model fitting.F1 scores showed a balanced improvement across all classes.

Validation Performance: Early stopping was employed to halt training after 3 epochs without improvement in validation loss, preventing overfitting. Class-specific precision and recall highlighted areas where the model struggled, such as "pituitary" samples.

Test Performance: The model achieved strong F1 scores on the test set, reflecting good generalization to unseen data.

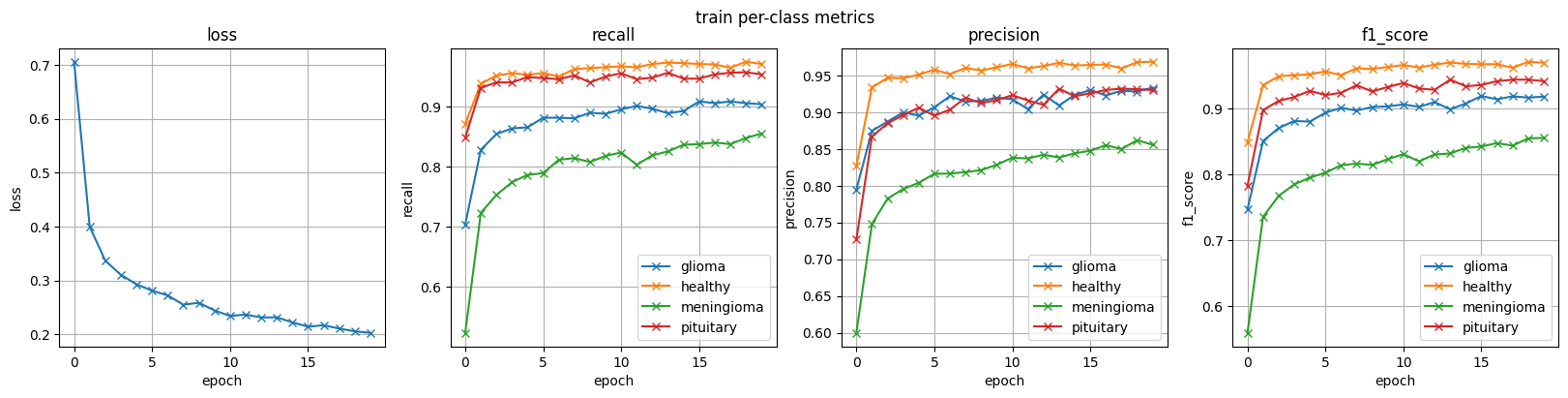
The dataset contains four classes, with the following sample counts:

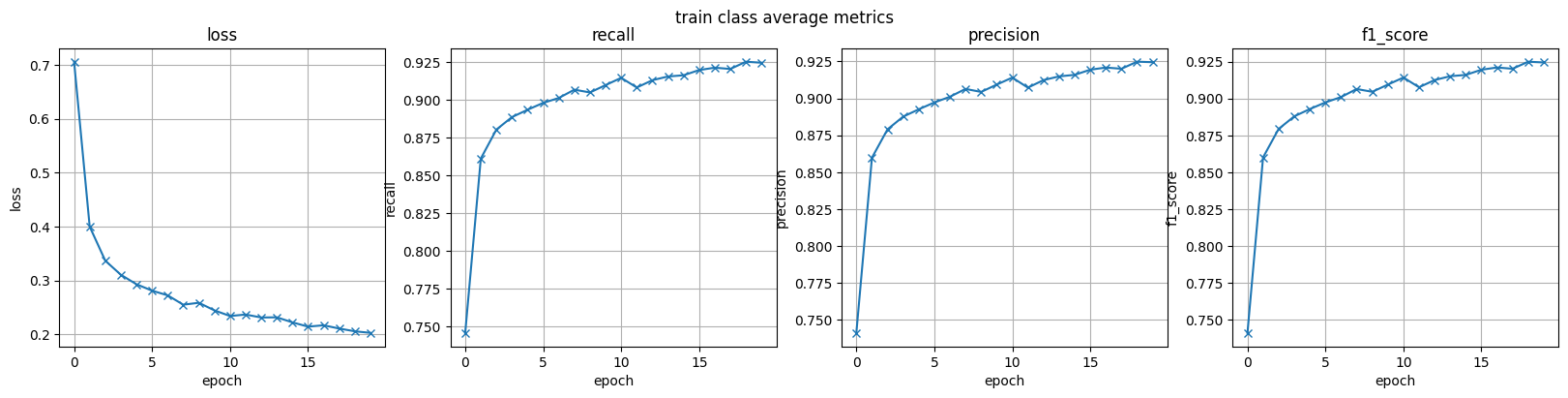
Healthy: 2,000 samples

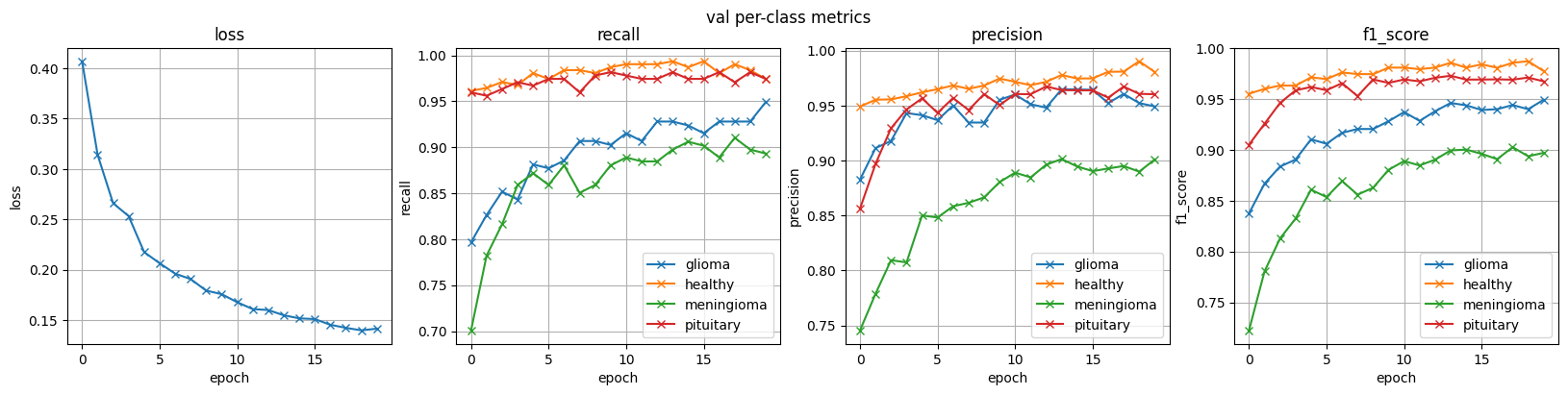
Glioma: 1,621 samples

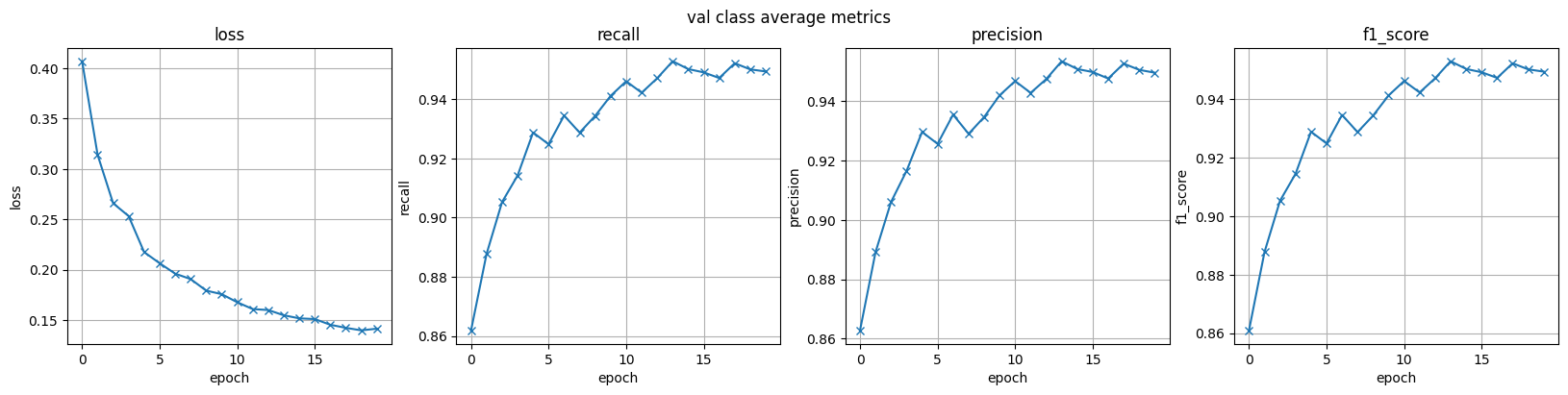
Meningioma: 1,645 samples

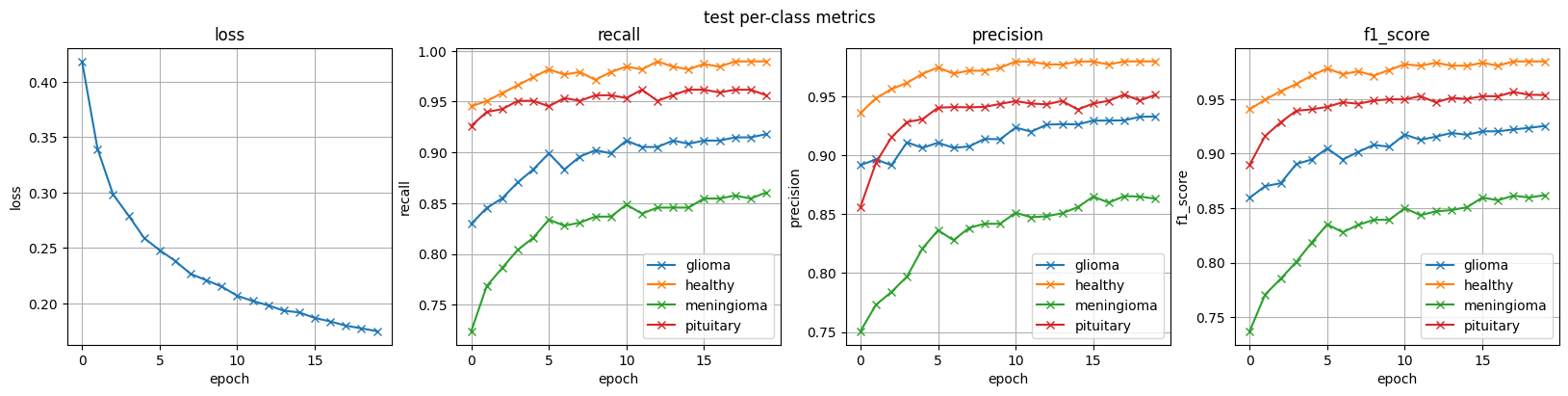
Pituitary: 1,757 samples

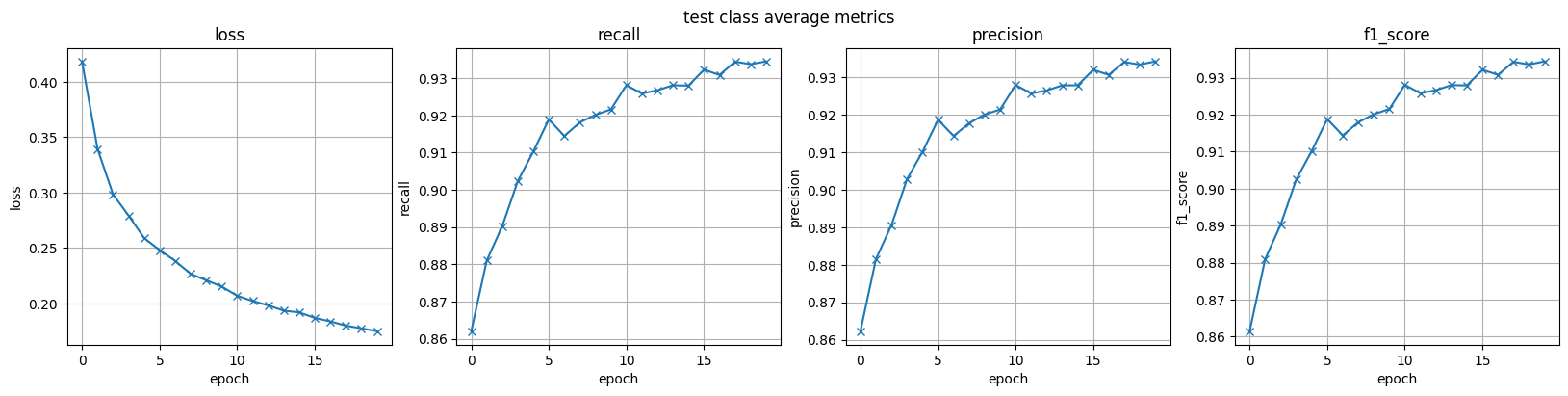












**Loss** Across Training, Validation, and Test Sets: The loss decreases steadily across all epochs for training, validation, and test sets, suggesting that the model is successfully learning the patterns in the data.

**Recall**

Per-Class Trends:The healthy class consistently achieves high recall, reflecting the model's ability to correctly identify healthy samples.The glioma and pituitary classes show steady improvement in recall, though glioma has slightly lower recall in earlier epochs.The meningioma class has the lowest recall, suggesting challenges in identifying this class.

Class Average: Recall improves significantly across epochs, stabilizing around 93% on the test set

**Precision**

Per-Class Trends: Precision for the healthy class is consistently high, reflecting the model's confidence in predictions for this class. The pituitary class also shows strong precision, indicating fewer false positives. The meningioma class initially struggles but improves steadily, aligning with recall trends.

Class Average: Precision improves across epochs, exceeding 93% by the final epoch in the test set.

**F1 Score**

Per-Class Trends: The healthy and pituitary classes consistently achieve high F1 scores, indicating a balance between precision and recall. The meningioma class starts with a low F1 score but improves steadily, suggesting the model gradually learns to handle this class.The glioma class shows a stable F1 score improvement, though slightly below the healthy and pituitary classes.

Class Average: F1 scores improve across epochs, reaching over 93% on the test set, demonstrating robust overall performance.

**Performance by Class**

Best Performance: The model performs exceptionally well on the healthy class, with near-perfect precision, recall, and F1 scores by the final epoch.

Most Challenging Class: The meningioma class consistently lags behind in all metrics, suggesting this class may require additional focus (e.g., data augmentation or better feature extraction).

**Consistency Across Training, Validation, and Test Sets**

The trends in performance metrics (loss, precision, recall, F1 score) are consistent across training, validation, and test sets, indicating that the model generalizes well and avoids overfitting.

**Enhanced model**

Fine-Tuning an existing architecture, this was your chosen enhancement approach.

Fine-Tuning Process : Unlike the baseline model, which only retrained the fully connected (FC) head, the enhanced model also fine-tuned the last convolutional block (layer4) of the ResNet-18 architecture. This allowed the model to adapt its higher-level feature representations to the specific characteristics of the brain tumor dataset, resulting in improved performance.

Learning Rate Adjustment: A smaller learning rate (1×10 ^− 5) was applied to the fine-tuned weights to prevent drastic updates to the pretrained features, ensuring controlled adjustments.

Baseline Approach: Uses the pretrained backbone without modifying any convolutional layers, except for replacing the final FC layer.

Enhanced Approach: Modifies the final FC layer and selectively fine-tunes part of the convolutional backbone (layer4), allowing the model to learn more dataset-specific features.

**Performance Metrics**

Fine-tuning led to remarkable improvements across all metrics (loss, recall, precision, and F1 score), achieving 100% accuracy in the final epochs.

**Train Performance:**

Loss: Decreased steadily to nearly zero, indicating excellent convergence.

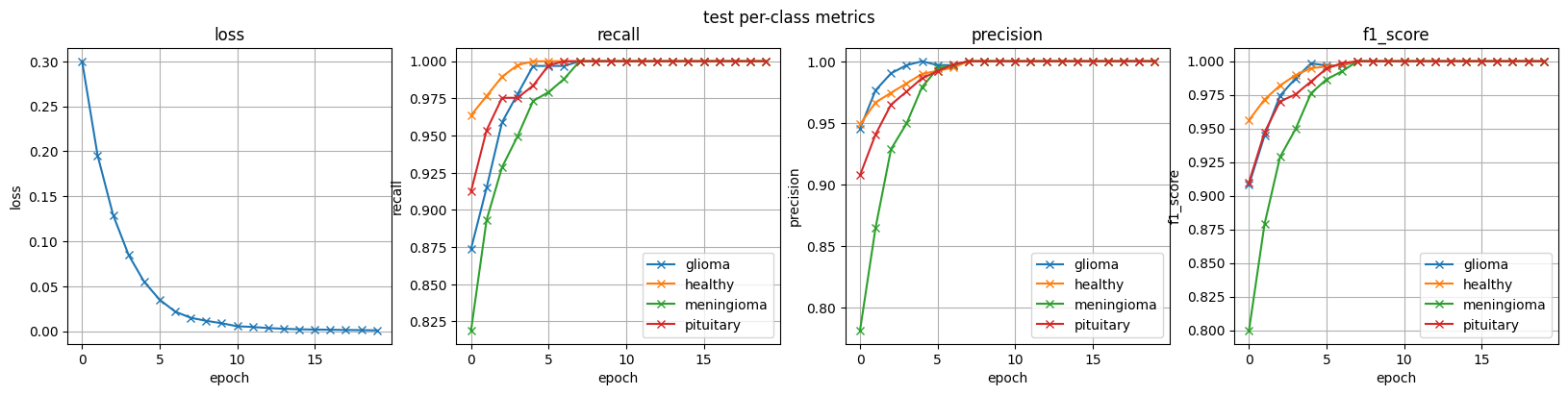
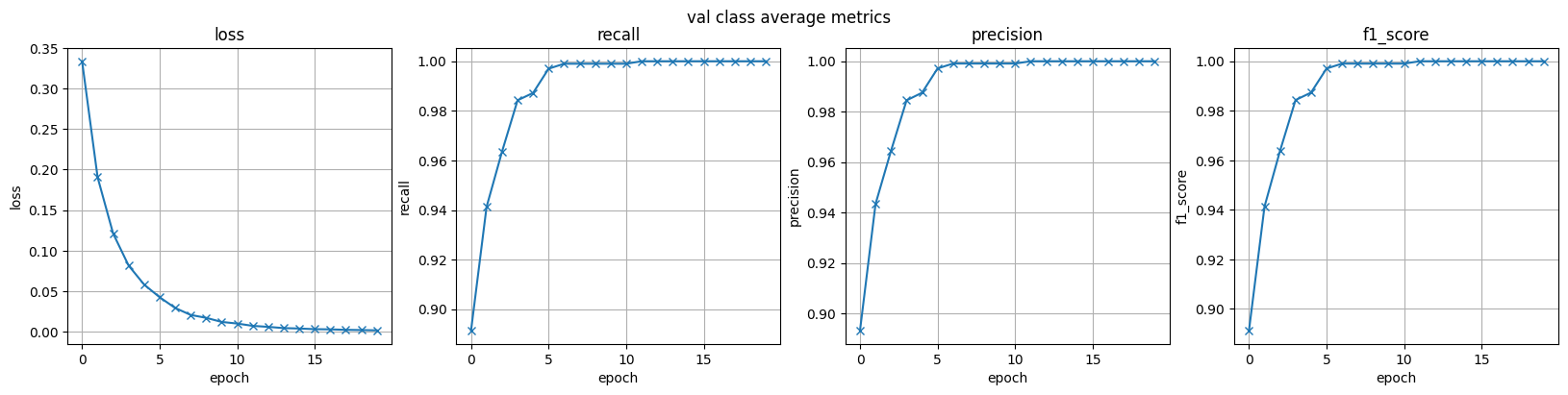
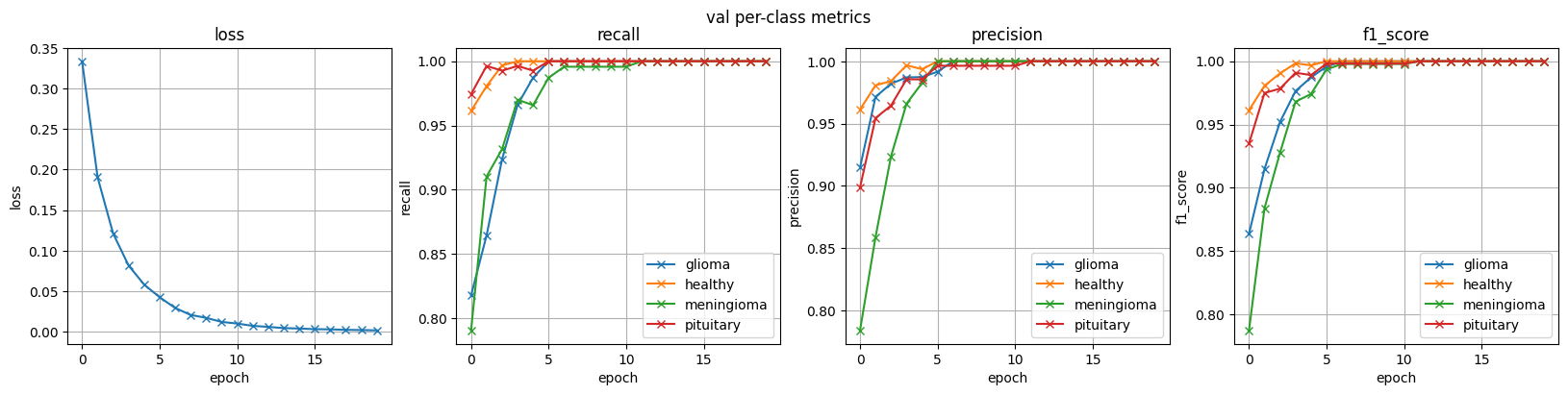
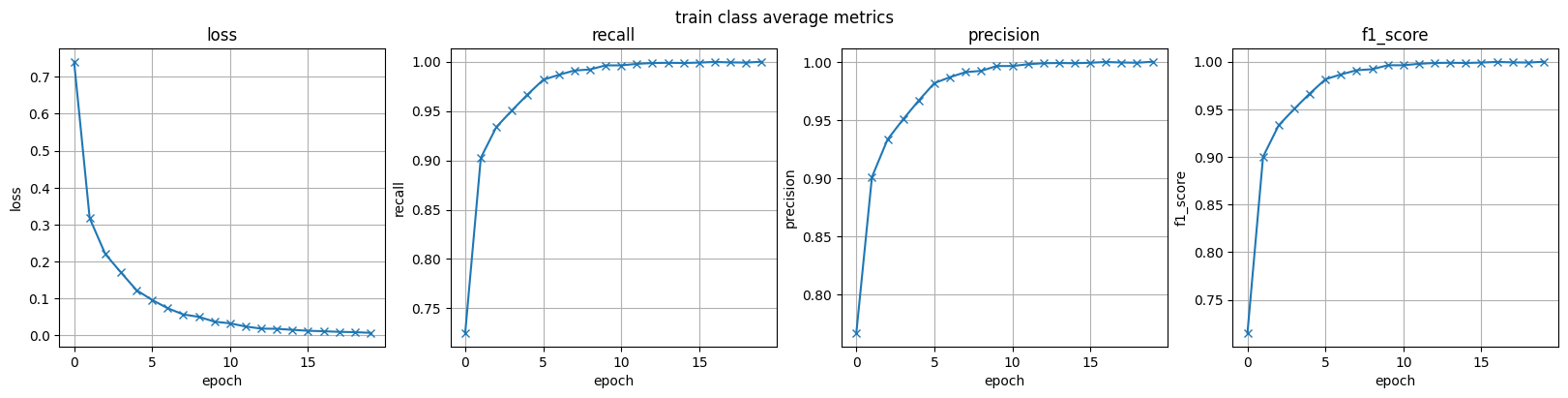
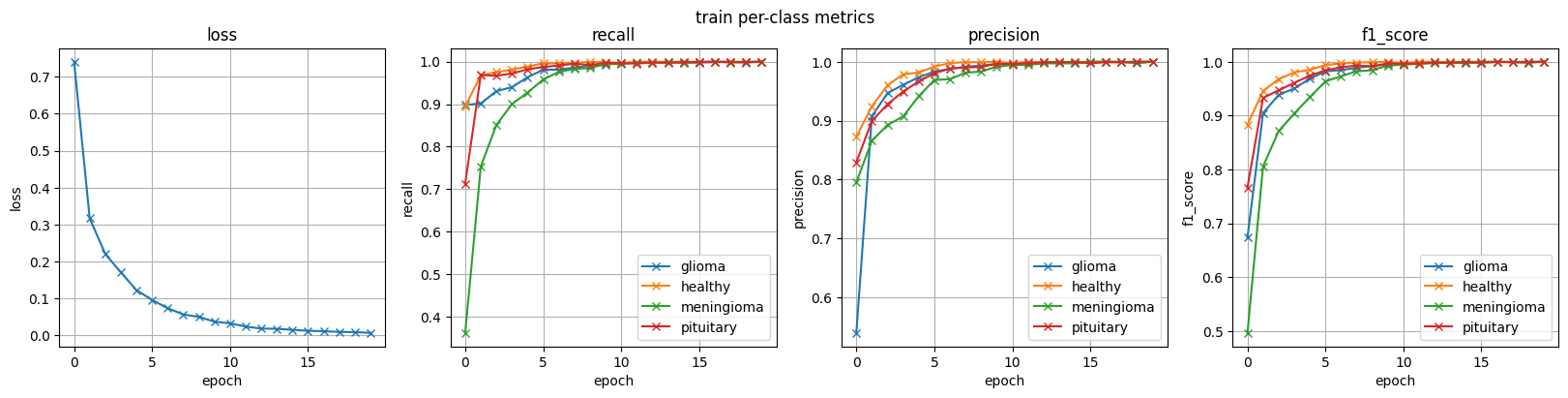
Per-Class Metrics: All classes (glioma, healthy, meningioma, pituitary) reached perfect scores for recall, precision, and F1 score, demonstrating the model's ability to correctly classify every sample in the training set.

Class-Average Metrics: Recall, precision, and F1 score reached a perfect score of 1.0 by the final epoch.

**Validation and Test Performance**

Metrics on unseen data also showed significant improvement compared to the baseline model. Validation and test losses approached zero, with perfect recall, precision, and F1 scores across all classes, reflecting robust generalization.

**Loss**: Train, validation, and test loss graphs show a rapid decrease to near-zero levels, confirming strong convergence and absence of overfitting.



**Per-Class Metrics**: Recall, precision, and F1 scores quickly approach 1.0 for all classes, with the most significant improvement observed in the meningioma class (which struggled in the baseline model).

**Class-Average Metrics**: Weighted averages for all metrics consistently improve, reaching perfect scores by epoch 10–15, indicating excellent overall performance.

**Comparison with Baseline**

Baseline Model: Struggled with the meningioma and glioma classes, particularly in recall and F1 score. Metrics plateaued at lower levels compared to the enhanced model.

Fine-Tuned Model: Resolved the issues with underperforming classes. Achieved perfect classification across all classes, highlighting the benefits of fine-tuning.

**Visualization of Predictions on Test Set**

fine\_tuning\_best\_model.eval()

# Get a batch of images from the test set

dataiter = iter(test\_loader)

images, labels = next(dataiter)

images, labels = images.to(device), labels.to(device)

# Select 6 random images

indices = np.random.choice(*a*=np.arange(len(images)), *size*=6, *replace*=False).astype(np.int64)

images = images[indices]

labels = labels[indices]

# Get model predictions

outputs = fine\_tuning\_best\_model(images)

\_, preds = torch.max(outputs, 1)  # get the predicted class indices

plot\_images(images.cpu(), dataset.labels\_inverse\_transform(labels.cpu().numpy()), *predictions*=dataset.labels\_inverse\_transform(preds.cpu().numpy()),

*denormalize*=transforms.Normalize(*mean*=[-0.485, -0.456, -0.406], *std*=[1/0.229, 1/0.224, 1/0.225]), *hspace*=0.3)

