Brain Tumor Classification Using Deep Learning and Meta-Learning

Improving Medical Imaging With AI

Sogand Ghasemi

1. Problem Definition

- Brain tumors are one of the most common and dangerous conditions that need early detection for better treatment and prognosis.
- Early detection through imaging techniques (e.g., MRI, CT scans) can significantly improve outcomes.

- ***Challenges:*** *Tumor classification from images can be complex due to high variability in tumor appearances, image quality, and overlapping features between tumor and non-tumor regions.*

2. Approach

To address these challenges, we propose a two-phase solution:

- **Phase 1**: ***CNN for Binary Classification*** – In this phase, we use a Convolutional Neural Network (CNN) to classify brain tumor images into two categories: Tumor vs No Tumor.
- **Phase 2**: ***Meta-Learning with MAML*** – In the second phase, we extend the CNN model using Model-Agnostic Meta-Learning (MAML) to classify different types of brain tumors, such as glioma and meningioma, making the model more adaptable and efficient.

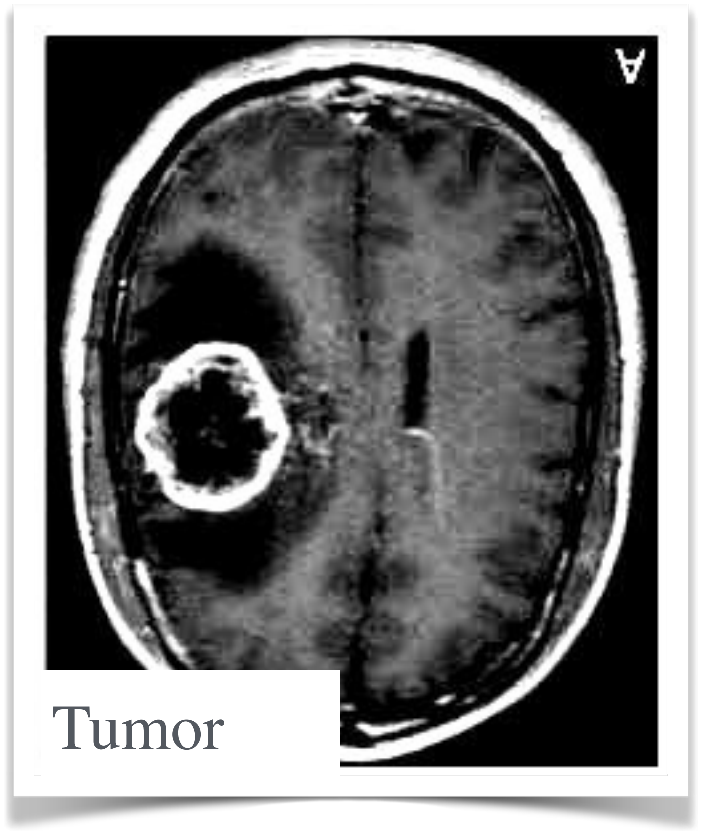
3. Phase 1

3.1)Dataset and Preprocessing

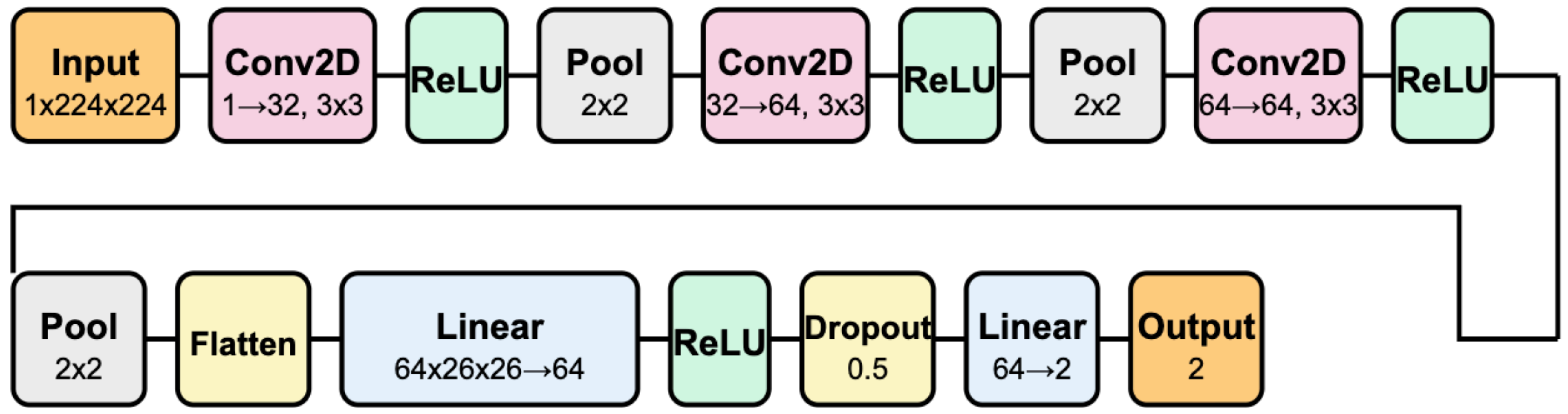
I used the **BR35H dataset** from Kaggle, which contains MRI images of brain tumors. This dataset consists of 3000 images and it is particularly useful for training models to classify brain tumor images into two categories: **Tumor(1)** vs **No Tumor(0)**.

- The images were resized to 224x224 pixels and converted to grayscale.
- The dataset was split into: **Training Set**(1800 images), **Validation Set**(600 images) and **Test Set**(600 images).
- The data was then normalized and converted into PyTorch tensors with shapes:
    - ***Training Set:*** *torch.Size([1800, 1, 224, 224])*
    - ***Validation Set:*** *torch.Size([600, 1, 224, 224])*
    - ***Test Set:*** *torch.Size([600, 1, 224, 224])*
- Finally, the data was loaded into DataLoader with a batch size of 32 for training.

No Tumor


Tumor

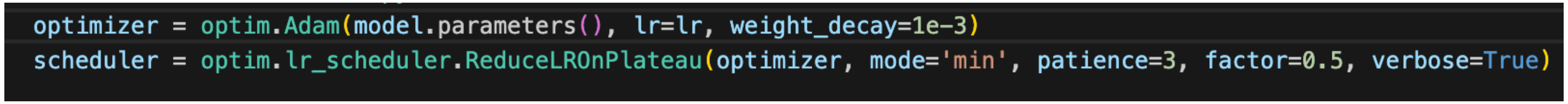
3.2)CNN Model Architecture



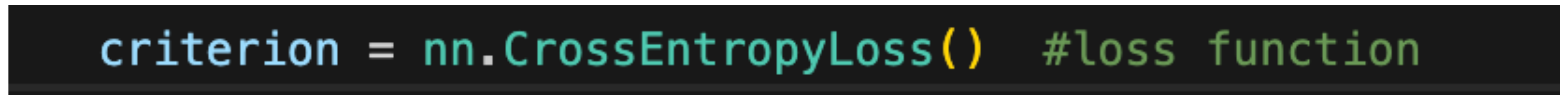
I implemented a custom Convolutional Neural Network (CNN) for binary brain tumor classification. The model consists of three convolutional layers (32, 64, 64 filters) for feature extraction, followed by a fully connected layer with 64 neurons, ReLU activation, and a Dropout (0.5) to prevent overfitting. The final output layer has two neurons for classification. We initially experimented with a deeper model (4 layers), but it increased the risk of overfitting, so we chose to stick with the 3-layer architecture for better performance.

3.2.1)Training

The model is trained for up to 100 epochs with a learning rate of 0.0001. We use the Adam optimizer to efficiently update the model's weights by adapting the learning rate based on the gradients, and CrossEntropyLoss is used as the loss function for classification. A learning rate scheduler reduces the learning rate by half if the validation loss doesn't improve for 3 epochs. The model's weights are updated using backpropagation, and early stopping is applied with a patience of 20 epochs to prevent overfitting. If the validation loss improves, the model is saved. Training progress is tracked with accuracy and loss curves for both training and validation.



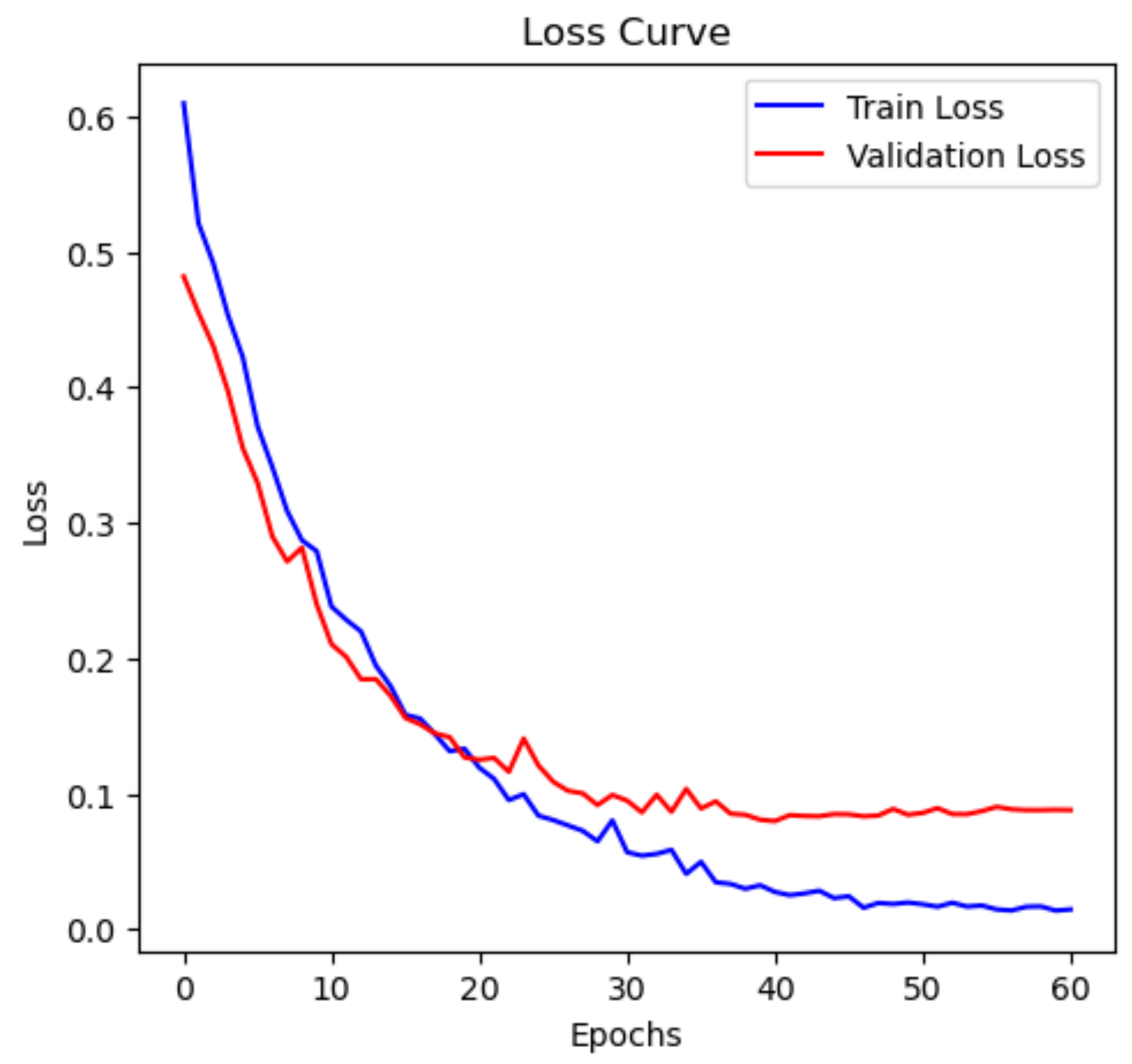
```
optimizer = optim.Adam(model.parameters(), lr=lr, weight_decay=1e-3)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer, mode='min', patience=3, factor=0.5, verbose=True)
```

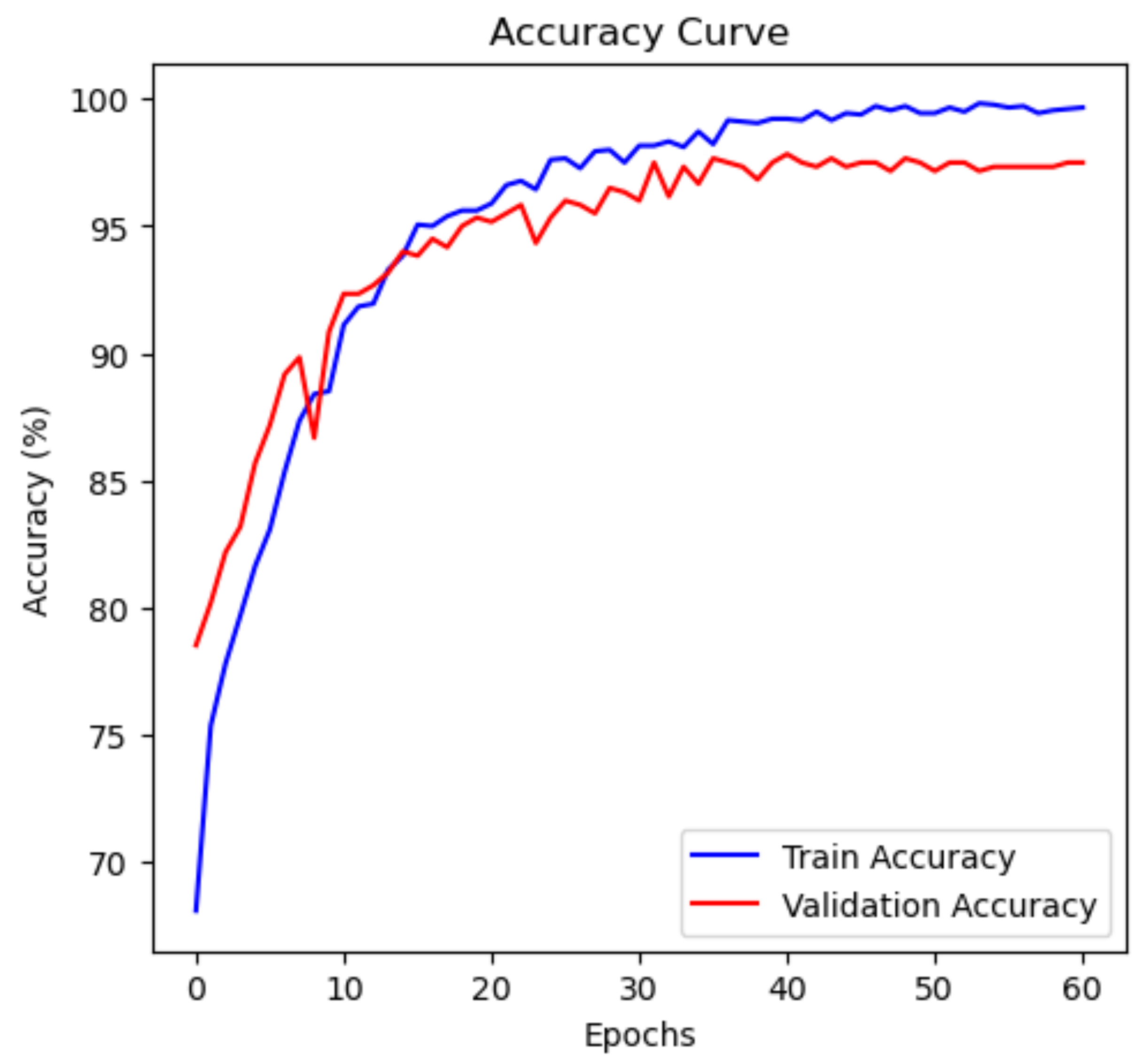


```
criterion = nn.CrossEntropyLoss()  #loss function
```

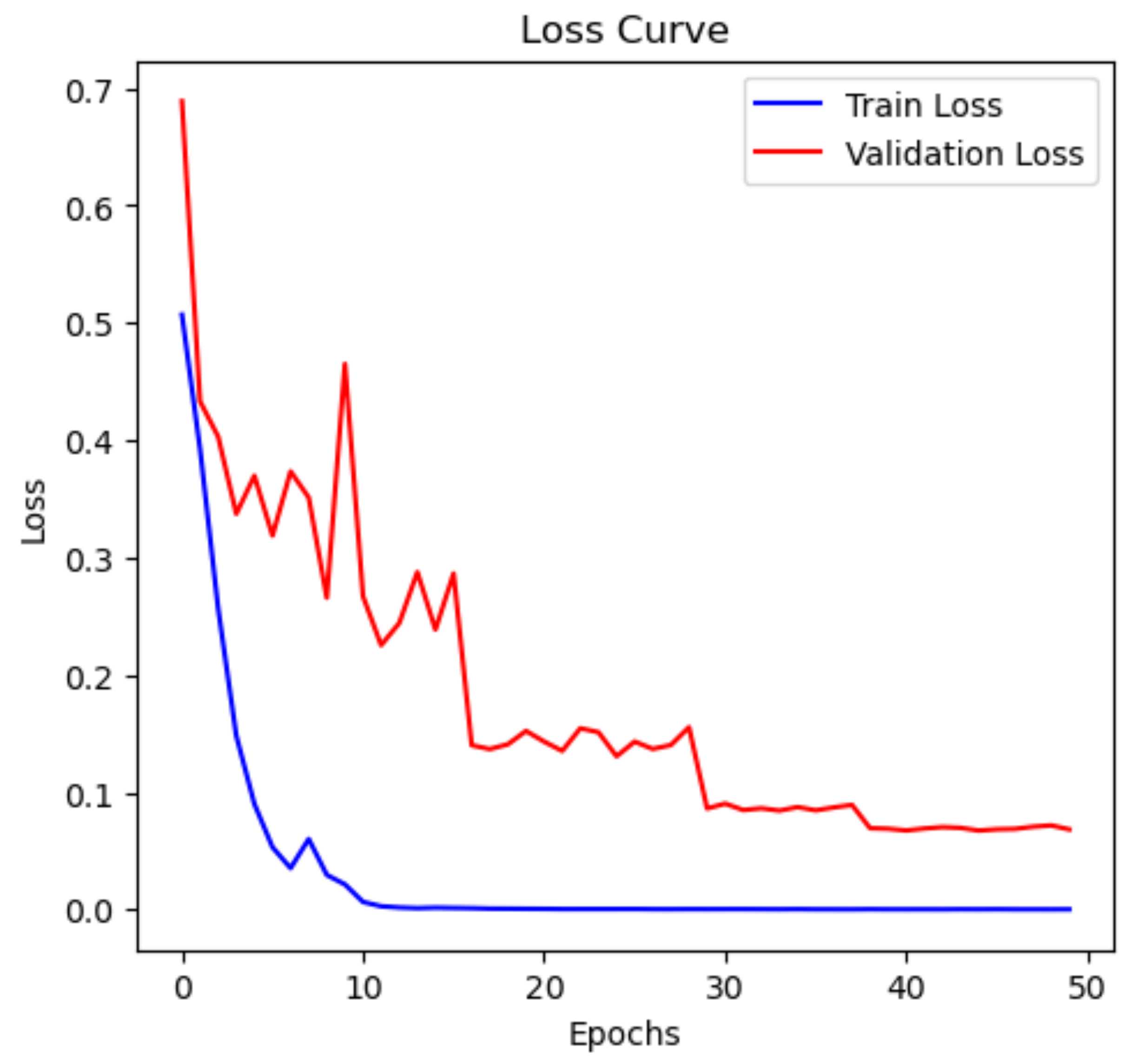
3.3)Results

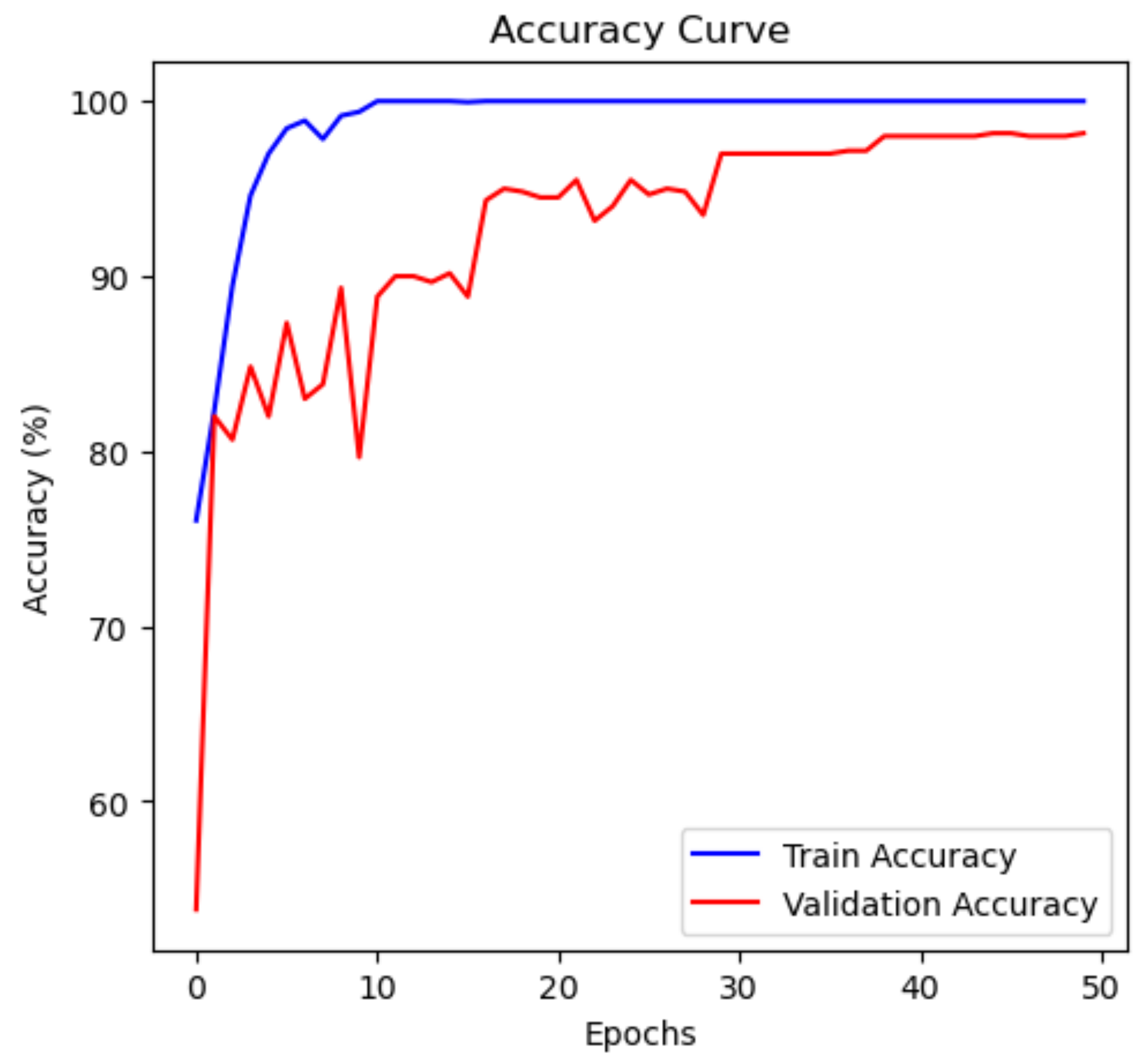
1.CNN with 3 Layers



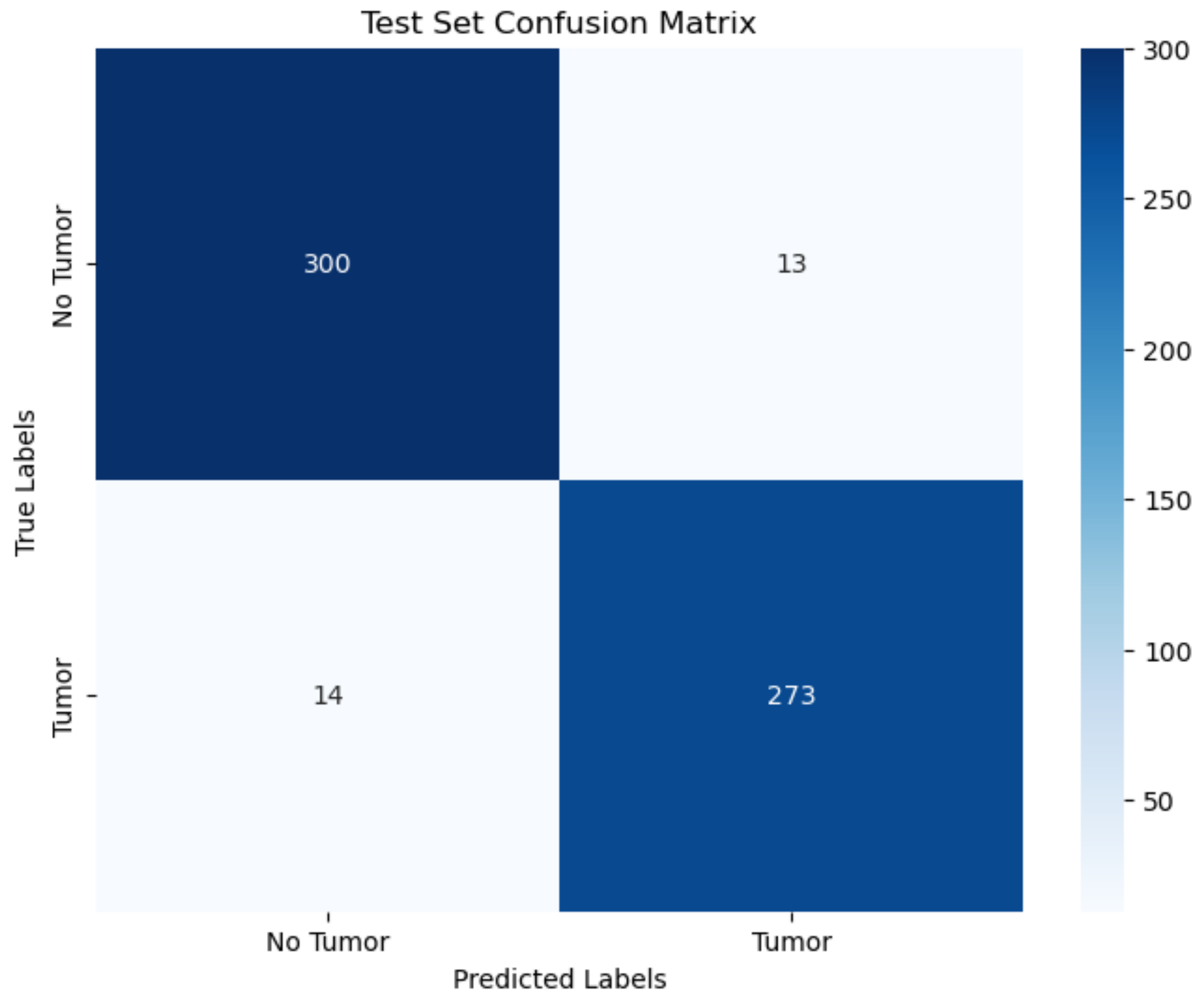


2.CNN with 4 Layers

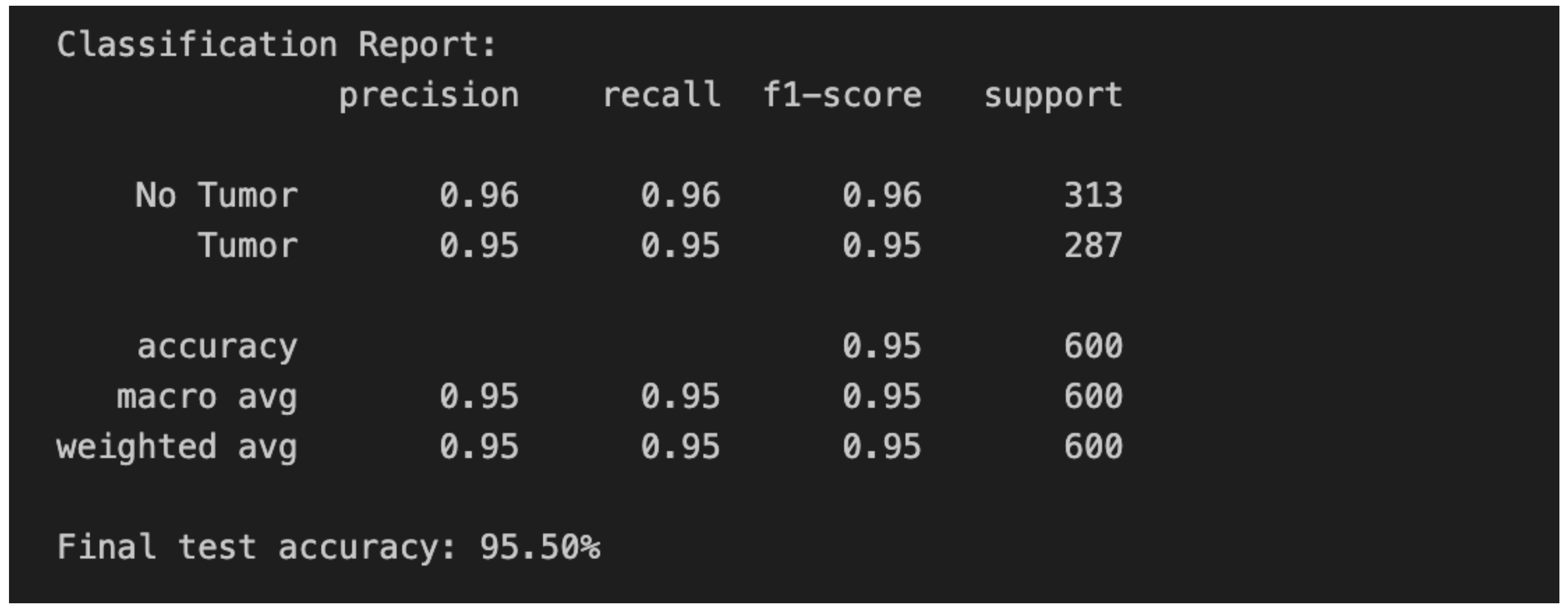




3.Classification Report



3.1) Confusion Matrix of 3-Layer CNN



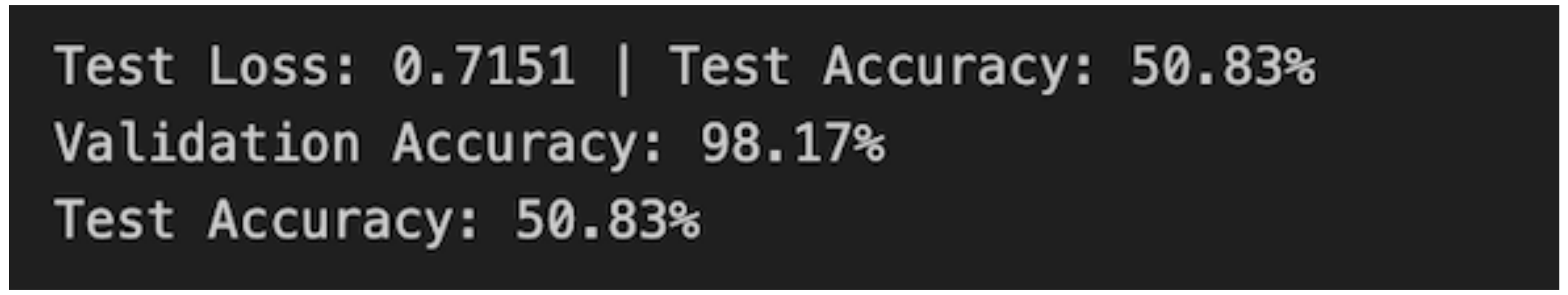
```
Classification Report:
              precision    recall  f1-score   support

    No Tumor       0.96      0.96      0.96       313
       Tumor       0.95      0.95      0.95       287

    accuracy                           0.95       600
   macro avg       0.95      0.95      0.95       600
weighted avg       0.95      0.95      0.95       600

Final test accuracy: 95.50%
```

3.2) Report of 3-Layer CNN



```
Test Loss: 0.7151 | Test Accuracy: 50.83%
Validation Accuracy: 98.17%
Test Accuracy: 50.83%
```

3.3) Report of 4-Layer CNN

4. Phase 2

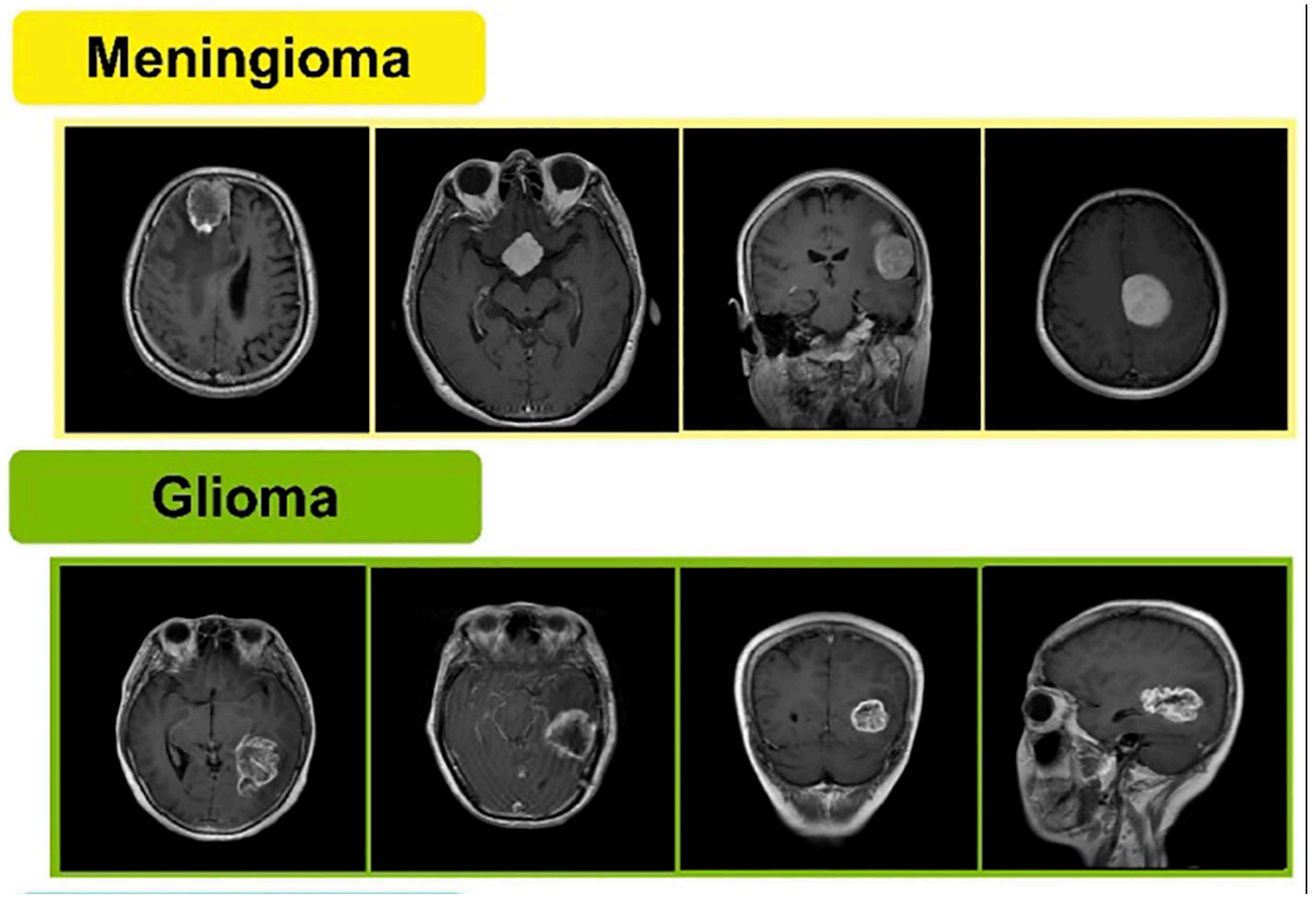
In Phase 2, I extend the CNN model from Phase 1 by applying **Model-Agnostic Meta-Learning (MAML)** for classifying brain tumor types, such as **glioma** and **meningioma**. MAML enables the model to quickly adapt to new tasks with minimal data by learning generalizable features. This allows the same CNN architecture to efficiently perform classification with few examples, improving its robustness for tumor type classification.

4.1)Dataset and Preprocessing

I obtained the dataset from Kaggle, which contains images of two types of brain tumors: glioma and meningioma. Glioma is a type of tumor that originates in the glial cells of the brain and spinal cord, which can affect both adults and children. It often grows quickly and can be highly invasive. Meningioma, on the other hand, is a tumor that arises from the meninges, the protective layers surrounding the brain and spinal cord, and is typically slower growing and less invasive than glioma. The dataset includes 1321 glioma images and 1339 meningioma images, bringing the total to 2660 images after filtering. I have assigned 0 to glioma and 1 to meningioma for classification purposes. For preprocessing, the images were resized to 64x64 pixels, converted to grayscale (1 channel), and then transformed into PyTorch tensors with pixel values normalized to the range [0, 1].

4.1.1)How CNN Classifies Meningioma and Glioma:

CNN differentiates Meningioma and Glioma by learning key MRI features. Meningiomas are well-defined, round, and extra-axial, with a homogeneous texture and strong contrast enhancement. Gliomas are irregular, infiltrative, intra-axial, and heterogeneous due to edema and necrosis. The CNN extracts these patterns through convolutional layers, enabling accurate classification.



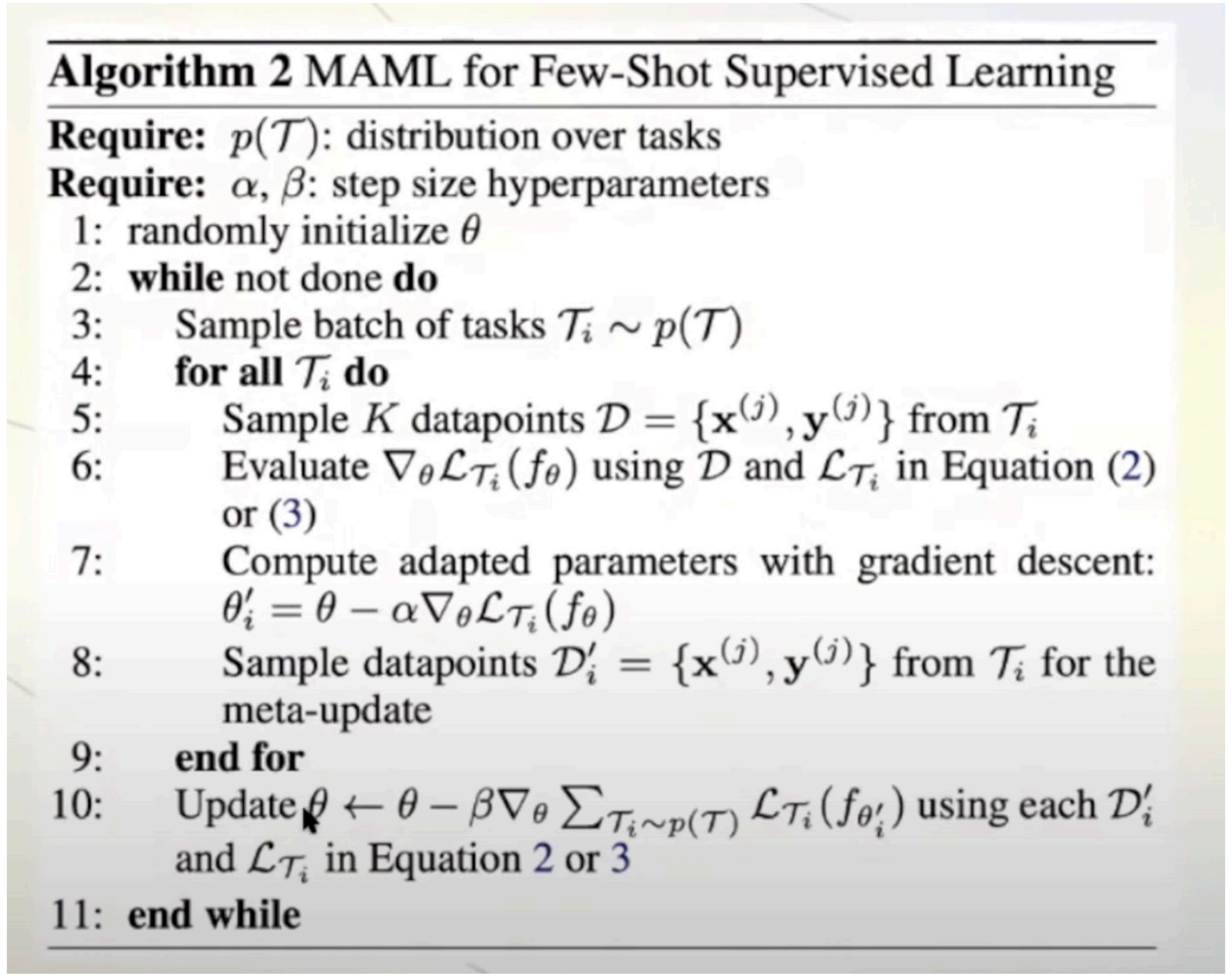
2.Meningioma and Glioma

4.2)Meta-Learning with MAML

In the meta-learning phase using Model-Agnostic Meta-Learning (MAML), we aim to train the CNN model to quickly adapt to new brain tumor classification tasks, such as classifying glioma and meningioma. MAML works by first training the model on multiple tasks, where each task is divided into a support set (for training) and a query set (for evaluation).

Inner-Loop:

The inner loop in MAML focuses on adapting the model to each specific task by performing multiple gradient updates using the support set (training data for each task). First, a copy of the model's weights is created(fast weights), which will be adapted during the inner loop. For each task, the model is trained on a small number of labeled examples from the support set(batch_size). The model performs gradient descent using these examples, computing the loss (Cross-Entropy) and updating the fast weights by calculating the gradients and applying the inner learning rate (`inner_lr`). This process is repeated for a set number of steps (`num_inner_steps`), which fine-tunes the model's parameters to minimize the task-specific loss. The inner loop allows the model to quickly adjust to each task by leveraging only a small set of training examples, making it ready for the subsequent evaluation on the query set in the outer loop.



Algorithm 2 MAML for Few-Shot Supervised Learning

```
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha$, $\beta$: step size hyperparameters
 1: randomly initialize $\theta$
 2: while not done do
 3:     Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 4:     for all $\mathcal{T}_i$ do
 5:         Sample $K$ datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$
 6:         Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation (2)
            or (3)
 7:         Compute adapted parameters with gradient descent:
            $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
 8:         Sample datapoints $\mathcal{D}'_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from $\mathcal{T}_i$ for the
            meta-update
 9:     end for
10:     Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using each $\mathcal{D}'_i$
        and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 2 or 3
11: end while
```

Outer-Loop

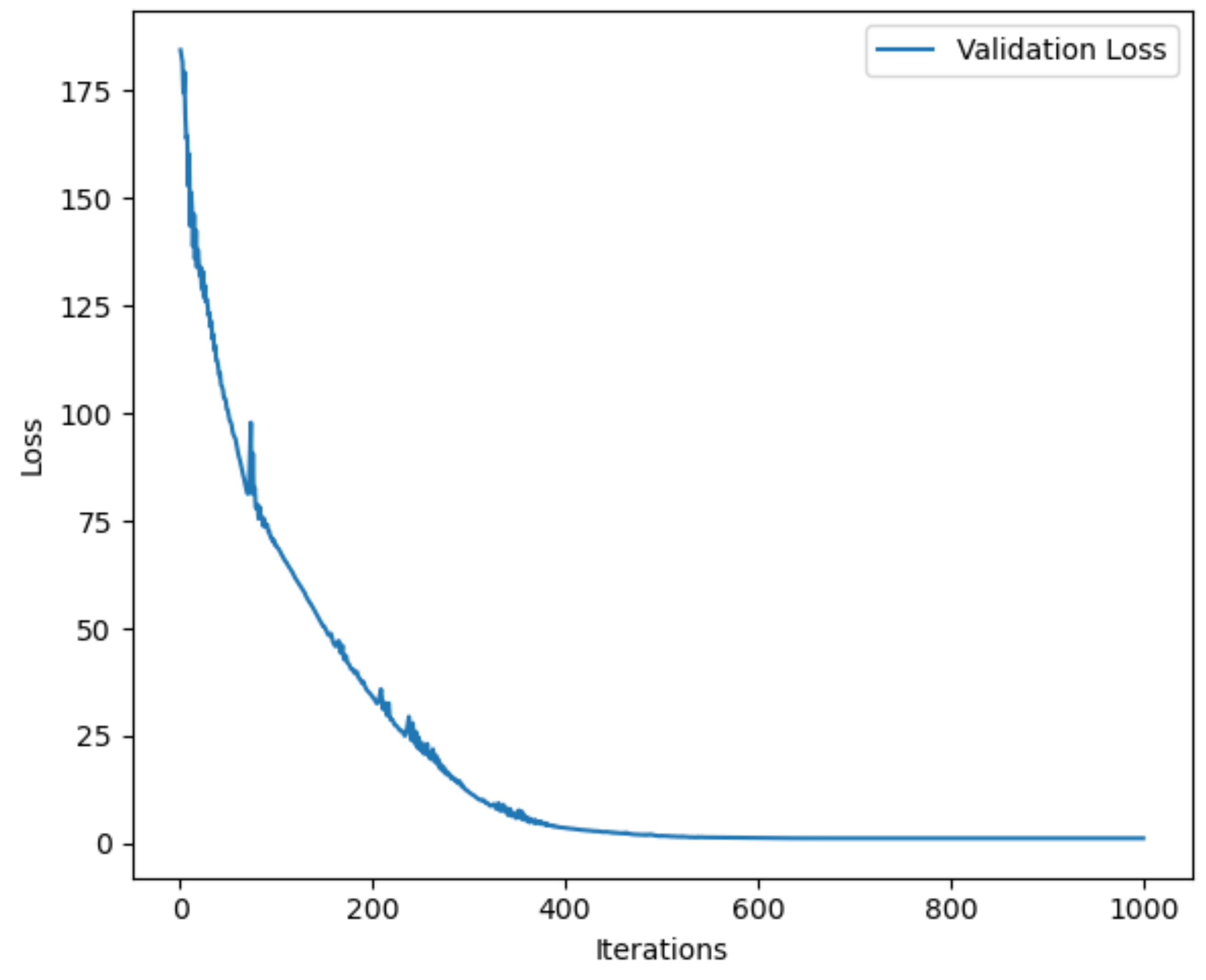
The outer loop in MAML focuses on meta-optimization by updating the model's general parameters based on its performance across multiple tasks. After the inner loop adapts the model to each specific task, the outer loop evaluates the model’s performance on the query set (the test data). The model’s loss on the query set is computed, and this meta-loss is accumulated across all tasks. The model’s meta-parameters (the initial weights) are then updated using the meta-optimizer (Adam) based on the gradients of the meta-loss, which helps the model improve its ability to adapt to new tasks. This process allows the model to learn a shared initialization that works well across tasks, enabling it to adapt quickly to new tasks with only a few gradient updates. The outer loop ensures that the model generalizes effectively, optimizing it for fast adaptation during future task-specific fine-tuning.

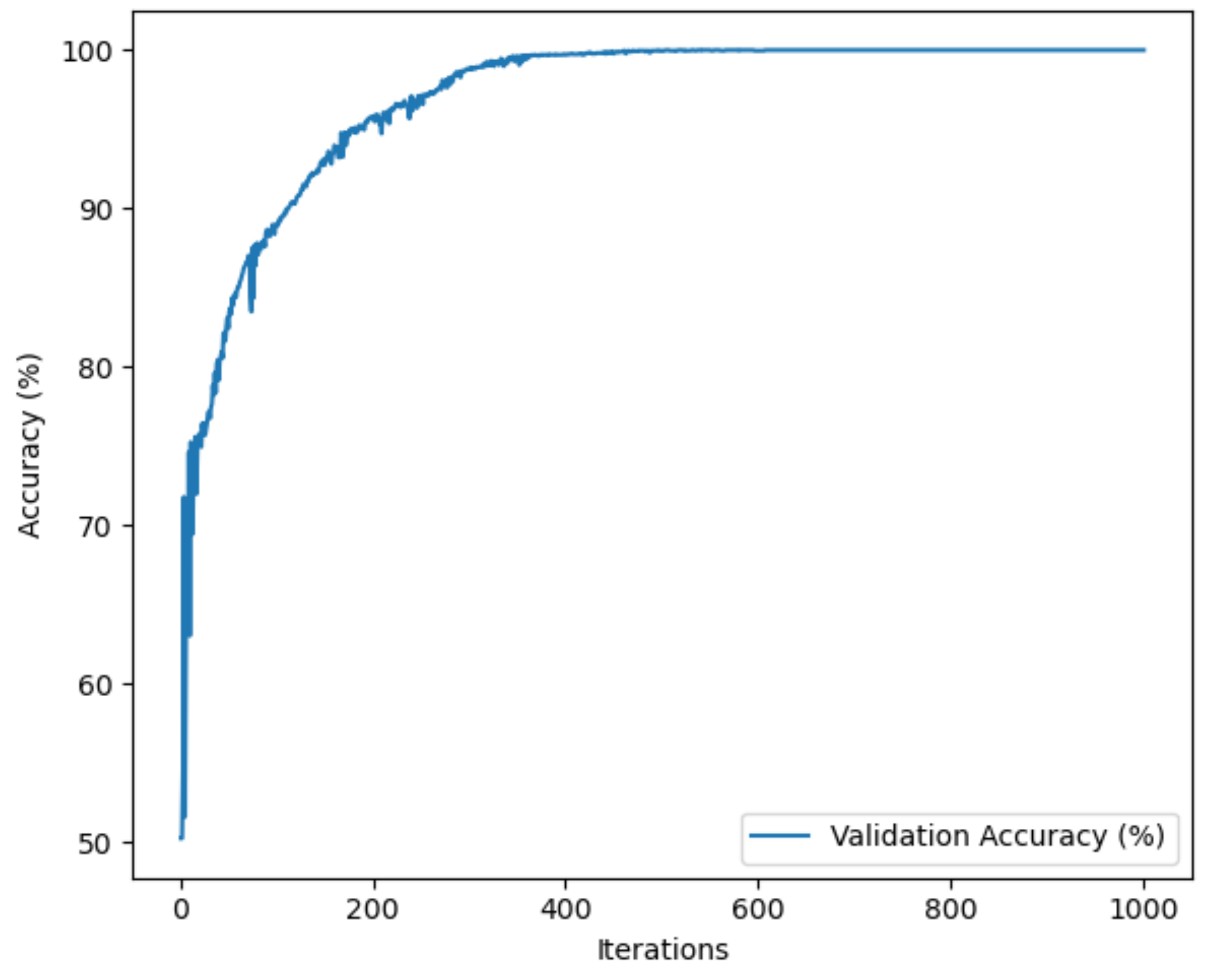
4.3)Training

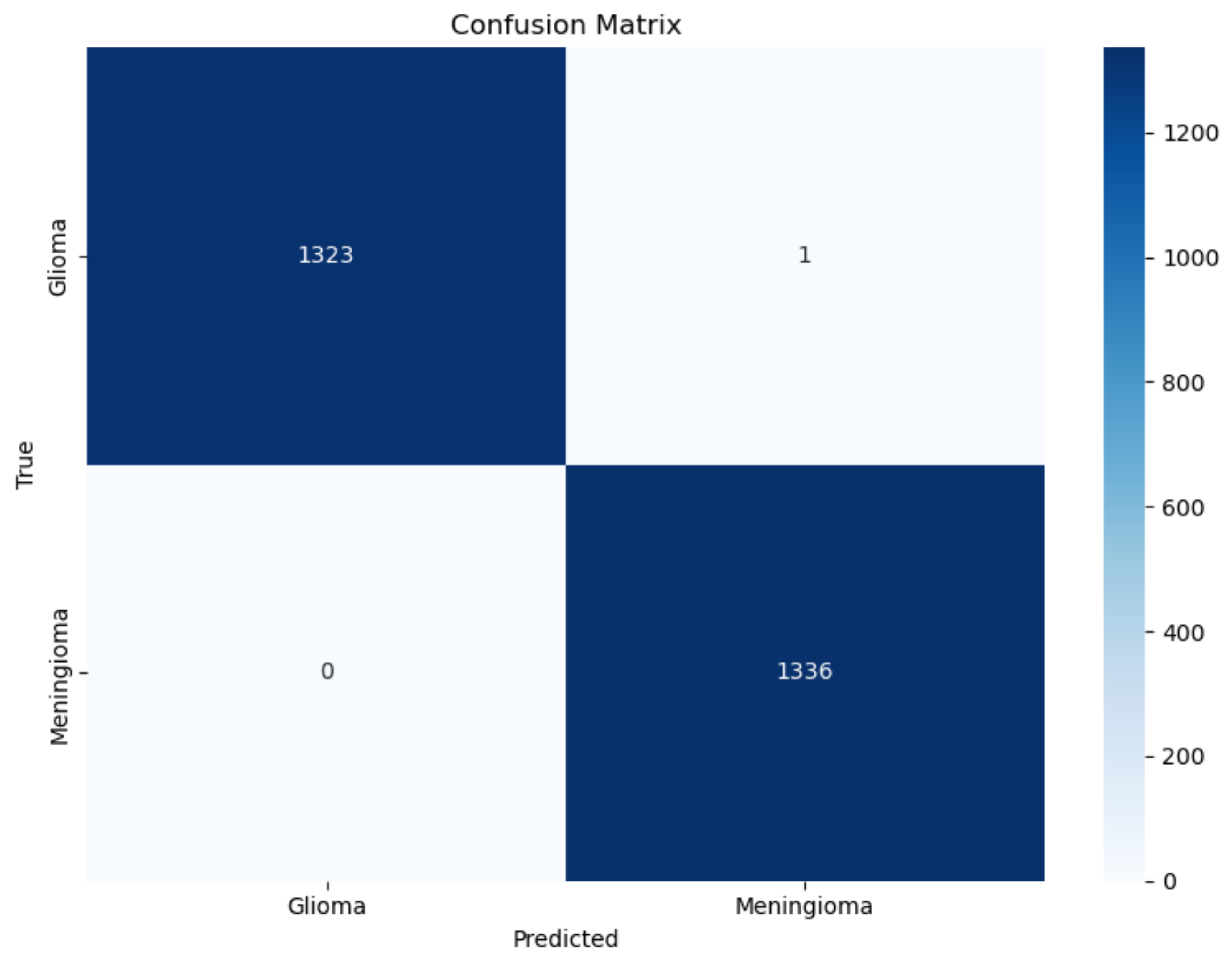
The model is trained for 1000 iterations with a meta-learning rate of 0.002 for the meta-optimizer (Adam). The inner learning rate for task-specific adaptations is set to 0.01. The dataset is split into 2 tasks, with each task divided into a support set (used for training) and a query set (used for evaluation). The task split ratio is 50% for both training and testing within each task. The batch size for both the support and query sets is set to 5, the model processes and evaluates 5 samples at a time during each iteration. The learning rate scheduler (ReduceLROnPlateau) adjusts the learning rate by halving it if the validation loss doesn’t improve after 5 epochs (patience=5). Weight decay is set to 1e-5 to prevent overfitting by penalizing large weights.

4.3)Results

The model shows nearly perfect classification of glioma and meningioma as seen in the confusion matrix. The training curves indicate that after around 600 iterations, both the validation loss and validation accuracy stabilize near 100%, suggesting the model has converged. To improve efficiency, we could implement early stopping to halt training once the model reaches stable performance, preventing unnecessary training beyond this point.







True \ Predicted	Glioma	Meningioma
Glioma	1323	1
Meningioma	0	1336