1. Overview

The **SqueezeNet++ Evo Transformer** is an advanced deep learning model designed for **retinal OCT image classification**, integrating SqueezeNet's efficient architecture with the Transformer's attention mechanism. This guide will help you set up, train, evaluate, and fine-tune the model.

Features:

- **High classification accuracy** for multiclass and binary retinal image classification.
- Integrated **Feature Pyramid Networks (FPN)** for multi-scale feature extraction.
- **Enhanced attention mechanisms** to focus on spatial and channel-wise features.
- Incorporation of **Graph Convolutional Networks (GCNs)** to model spatial relationships within OCT images.

2. Installation

Prerequisites

To run this project, ensure the following dependencies are installed:

- Python 3.7+
- PyTorch 1.8.0+
- CUDA 10.1 or above (optional for GPU)
- Required Python packages

Install Required Libraries

Use the following commands to set up the environment:

```
# Clone the repository

git clone https://github.com/Pavithra-Mani94/SqueezeNet-Evo-Transformer.git

cd SqueezeNet-Evo-Transformer



# Create a virtual environment (optional)

python3 -m venv squeeze-env

source squeeze-env/bin/activate



# Install dependencies

pip install -r requirements.txt
```

3. Dataset Preparation

Supported Datasets

1. **OCT2017**: Available here
2. **Duke OCT Dataset**: Available on request.
3. **Real-time dataset**: This is a proprietary dataset sourced from Mahatma Eye Hospital Private Limited.

Dataset Structure

Ensure the datasets are arranged in the following structure:

```
/data
  /OCT2017
    /CNV
    /DME
    /Drusen
    /Normal
  /Duke
    /AMD
    /Normal
  /Real-time
    /class1
    /class2
    ...
```

Preprocessing

Resize the images to **227x227** pixels before training:

```
from PIL import Image
import os

# Resize images to 227x227
def resize_images(folder):
    for img_file in os.listdir(folder):
        img_path = os.path.join(folder, img_file)
        with Image.open(img_path) as img:
            img = img.resize((227, 227))
            img.save(img_path)
```

4. Training the Model

Training Parameters

Key training parameters include:

- **Batch size**: 64
- **Learning rate**: 0.0001
- **Optimizer**: Adam
- **Epochs**: 50
- **Loss function**: Categorical cross-entropy

Training Command

To start training the model, use the following command:

```
python train.py --dataset-path ./data/OCT2017 --model SqueezeNetEvo --batch-size 64 --
epochs 50 --lr 0.0001
```

Explanation:

- --dataset-path: Path to the dataset directory.
- --model: The model to be used, which is **SqueezeNetEvo** in this case.
- --batch-size: Size of each batch of data.
- --epochs: Number of training epochs.
- --lr: Learning rate.

The model will be saved automatically in the ./saved_models directory after training.

Example Command for Real-time Dataset

```
python train.py --dataset-path ./data/Real-time --model SqueezeNetEvo --batch-size 64 --
epochs 50 --lr 0.0001
```

5. Evaluation and Testing

Evaluation Metrics:

The model supports the following metrics for evaluation:

- **Accuracy**
- **Precision**
- **Recall (Sensitivity)**
- **Specificity**
- **F1-Score**
- **Matthews Correlation Coefficient (MCC)**
- **Area Under the Curve (AUC-ROC)**

Running Evaluation

To evaluate the model on a test dataset:

```
python evaluate.py --model-path ./saved_models/squeezenet_evo_best.pth --dataset-path ./data/OCT2017 --batch-size 64
```

6. Evaluation and Testing

Evaluation Metrics:

The model supports the following metrics for evaluation:

- **Accuracy**
- **Precision**
- **Recall (Sensitivity)**
- **Specificity**
- **F1-Score**
- **Matthews Correlation Coefficient (MCC)**
- **Area Under the Curve (AUC-ROC)**

Running Evaluation

To evaluate the model on a test dataset:

```
python evaluate.py --model-path ./saved_models/squeezenet_evo_best.pth --dataset-path ./data/OCT2017 --batch-size 64
```


Example Evaluation Results:

```
Accuracy: 99.23%
Precision: 98.76%
Recall: 99.69%
F1-Score: 99.20%
AUC-ROC: 0.987
```

6. Fine-Tuning the Model

Fine-tuning helps to improve the model's performance on smaller or specialized datasets.

Fine-Tuning Command

```
python fine_tune.py --model-path ./saved_models/squeezenet_evo_best.pth --new-dataset-path ./data/Real-time --batch-size 32 --epochs 20 --lr 0.00001
```

Explanation:

- **--model-path**: The pre-trained model you want to fine-tune.
- **--new-dataset-path**: Path to the new dataset for fine-tuning.
- **--batch-size**: Batch size for fine-tuning.
- **--epochs**: Number of fine-tuning epochs.
- **--lr**: Fine-tuning learning rate.

7. **Model Architecture**

Here's an overview of the key components of **SqueezeNet++ Evo Transformer**:

- **SqueezeNet++ Layers**: Lightweight feature extraction with Fire modules.
- **Feature Pyramid Integration (FPI)**: For multi-scale feature extraction.
- **Attention Mechanism**: Enhanced spatial and channel-wise attention.
- **Graph Convolutional Networks (GCN)**: To model spatial relationships.
- **Transformer Encoder**: Handles sequential information in the image patches.

  The following diagram shows the overall architecture:

  ```
  Input Image -> SqueezeNet++ -> Feature Pyramid Integration -> Attention
  Mechanisms -> Graph Convolutional Networks -> Transformer Encoder -> Output
  Class Prediction
  ```

8. Visualizing Results

Use the **Grad-CAM** tool to visualize the regions of the image that the model focused on while making its predictions.

Grad-CAM Command

```
python grad_cam.py --model-path ./saved_models/squeezenet_evo_best.pth --image-path
./data/OCT2017/test/CNV/img1.jpg
```

Output:

The command generates a heatmap showing areas of focus and overlays it on the input image.

9. **Using Pre-trained Models**

Pre-trained models can be loaded and used for prediction:

```
python predict.py --model-path ./saved_models/squeezenet_evo_best.pth --image-path
./data/OCT2017/test/CNV/img1.jpg
```

10. Permanent Links

All the data and code are hosted on stable platforms for long-term access:

- **Code Repository**: GitHub - SqueezeNet++ Evo Transformer
- **Datasets**:
  - OCT2017: Mendeley Link
  - Duke: Available on request.

This documentation covers all the necessary steps to set up, train, evaluate, and fine-tune the **SqueezeNet++ Evo Transformer** model.