Assignment 2:Deployment Instructions

By: Pradyumna Seethamraju

Deployment Instructions for Code 2

1. Prerequisites

1. **Python Installation**: Ensure Python 3.7 or above is installed.
2. **Install Required Packages**: Using `pip`, install the necessary libraries as listed below:

```
pip install torch torchvision numpy opencv-python-headless scikit-learn
```

3. **Hardware Requirements**:

- **GPU** (optional but recommended): The code is optimized for CUDA-enabled GPUs for faster computation. Ensure the appropriate CUDA toolkit is installed if running on a GPU.

2. Project Structure

Organize the code and assets as follows:

```
project-root/
├── main.py            # Main execution file (contains the `main()` function)
├── models/
│   ├── diffusion_model.pth  # Trained Diffusion Model (optional pre-trained weights)
│   └── cnn_lstm_model.pth # Trained CNN+LSTM Model (optional pre-trained weights)
├── data/
│   └── video.mp4            # Video file for prediction or testing
└── output/
        └── best_model.pth  # Checkpoint file for the best
```

3. Training the Model

3.1 Data Preparation

1. Place the video file(s) to be used for training in the `/data` directory.
2. Update `VIDEO_PATH` in `main.py` to the path of the video file for training. Optionally, adjust configurations for `BATCH_SIZE`, `NUM_EPOCHS`, `LEARNING_RATE`, and `MAX_FRAMES` based on hardware capabilities.

3.2 Running Training

1. **Execute the Training Script**: Run the following command to start training both models:

```
python main.py
```

2. **Output Checkpoints**: During training, model checkpoints are saved in the `/output` directory with the best-performing model (based on F1 score) stored as `best_model.pth`.

3.3 Monitor Training

- Training logs will display epoch-wise loss and metrics (Accuracy, F1 Score, AUC). Adjust hyperparameters if metrics indicate poor performance.

4. Running Inference with Trained Models

1. **Ensure Checkpoints are Available**:
   - The `cnn_lstm_model.pth` and `diffusion_model.pth` (if available) should be placed in `/models/`.
   - If pre-trained models are not available, train them as outlined in Step 3.
2. **Load Trained Model**:
   - The `ModelInference` class in `main.py` will automatically load the best model checkpoint from `/output/best_model.pth` and the Diffusion Model from `/models/diffusion_model.pth` if provided.
3. **Execute Inference Script**:
   - To run inference on a new video, execute the `predict()` function in the `ModelInference` class.
   - Use the following command to run inference:

     ```
     python main.py
     ```

   - **Prediction Output**: Predictions will be printed to the console or saved based on your `predict()` method configuration. Modify `ModelInference.predict()` if you wish to save predictions to a file.

5. Configuration and Hyperparameter Adjustments

The following configurations in `main.py` can be customized for deployment:

- **Device**: The code automatically detects and uses a GPU if available. Adjust `device` manually if needed.
- **Batch Size, Epochs, and Learning Rate**: Update values like `BATCH_SIZE`, `NUM_EPOCHS`, and `LEARNING_RATE` in the configuration section.
- **Logging**: The `logging` library is used to monitor progress. Adjust the logging level as desired.

6. Optional: Containerizing the Deployment

To package this deployment with Docker, follow these steps:

1. **Dockerfile Creation**:
    - Create a `Dockerfile` in the project root:

```
FROM pytorch/pytorch:latest

# Set working directory

WORKDIR /app

# Copy project files

COPY . .

# Install dependencies

RUN pip install -r requirements.txt

# Run the application

CMD ["python", "main.py"]
```

Build the Docker Image:

```
docker build -t video-anomaly-detection .
```

Run the Docker Container:

```
docker run --gpus all -v $(pwd)/output:/app/output -v $(pwd)/data:/app/data video-anomaly-detection
```

This will run the `main.py` script inside a container, utilizing any available GPU resources if your environment supports it.