Assignment 2: Design Document

By: Pradyumna Seethamraju

---

1. Model Architecture and Flow

The code consists of two primary models: a **Diffusion Model** for feature extraction and anomaly detection (reconstruction error) and a **CNN+LSTM Model** for sequence-based classification. Here's a detailed overview of each component:

- **Video Dataset**:
  - **Purpose**: Loads video frames from a specified file, applies preprocessing transforms, and makes frames available for further model processing.
  - **Flow**:
    1. Video frames are read using OpenCV.
    2. Frames are converted to RGB format.
    3. Each frame is optionally transformed and resized before use.
- **Diffusion Model**:
  - **Architecture**:
    1. **Encoder**: Three convolutional layers with increasing depth (64, 128, 256 channels), using `BatchNorm2d` and `ReLU` activations to extract features.
    2. **Decoder**: Three deconvolutional (transpose convolution) layers to reconstruct input frames and calculate reconstruction error.
  - **Flow**:
    1. Input frames are passed through the encoder to obtain compressed feature representations.
    2. The decoder reconstructs the frames from the compressed representation.
    3. Reconstruction error (difference between input and reconstructed frames) is computed, which helps in detecting anomalies.
- **CNN+LSTM Model**:
  - **Architecture**:
    1. **CNN Block**: Two convolutional layers (128 and 64 channels) with `BatchNorm2d` and `ReLU`, followed by dropout for regularization.
    2. **LSTM Block**: Processes sequential data from the CNN output. A 2-layer LSTM with a hidden size of 64 captures temporal dependencies across frames.
    3. **Output Layer**: Fully connected layer with a sigmoid activation for binary classification (probabilistic output).
  - **Flow**:

1. Encoded features (from the Diffusion Model) are processed frame by frame by the CNN.
2. The reshaped CNN output is fed to the LSTM layer to capture temporal dependencies.
3. The final output is generated by a fully connected layer, which performs binary classification.

2. Justification for Using a Diffusion Model

The **Diffusion Model** serves as a feature extractor and a mechanism for detecting reconstruction errors, often indicative of anomalies. Here's why it's appropriate:

- **Anomaly Detection via Reconstruction Error**: Diffusion models, designed to model data distributions, can learn high-quality representations of typical data. If input deviates from expected patterns, the reconstruction error will be significant, signaling a possible anomaly.
- **High-Quality Feature Extraction**: By training an encoder-decoder network, the model extracts compressed, informative features capturing patterns within frames, which are later leveraged by the CNN+LSTM for classification.
- **Complementarity with CNN+LSTM**: The encoded features from the Diffusion Model reduce input dimensionality for the CNN+LSTM, helping it focus on essential aspects rather than noise. This setup leverages both spatial features from frames and temporal dependencies across sequences.

3. Process, Assumptions, and Configurations

3.1 Process Flow

1. **Data Preprocessing**:
    - Video frames are loaded, resized, normalized, and transformed.
2. **Diffusion Model**:
    - Trained to reconstruct each frame and detect anomalies based on reconstruction error.
3. **CNN+LSTM Model**:
    - Extracted features from the Diffusion Model are passed to the CNN+LSTM model to detect temporal anomalies and perform classification.
4. **Evaluation**:
    - Accuracy, F1 score, and AUC are computed on test data to assess model performance.

3.2 Assumptions

- **Consistency in Video Frame Rates**: Assumes that video frames are consistent in rate to ensure temporal data accuracy in the LSTM.

- **Binary Classification**: Assumes a binary classification problem, but the architecture is adjustable for multi-class with minor changes.
- **Anomaly Detection via Reconstruction Error**: Assumes that anomalies result in higher reconstruction error. This aligns with settings where out-of-distribution samples differ structurally or visually from the training data.

3.3 Configurations

- **Device**: Configured for GPU (CUDA) if available, otherwise CPU.
- **Transformations**: Resizing frames to `180x320` for optimal model processing.
- **Batch Sizes**: Configured as `BATCH_SIZE=8` for the CNN+LSTM training, chosen based on memory limits.
- **Learning Rate and Scheduler**: Uses `LEARNING_RATE=0.001` with `ReduceLROnPlateau` to adjust the rate based on performance metrics, enhancing training stability.

4. Steps to Replicate Training and Using Trained Models for Inference

4.1 Training Steps

1. **Data Preparation**:
    - Ensure the video data is in the specified location (`VIDEO_PATH`).
    - Configure preprocessing transformations such as resizing and normalization.
2. **Initialize the Dataset and DataLoader**:
    - Load frames using `VideoDataset`.
    - Define train/test splits, set up DataLoaders with batch processing and optional pinning for GPU.
3. **Train the Diffusion Model**:
    - Train the encoder-decoder structure on the video data to reconstruct frames and calculate reconstruction error.
4. **Extract Features**:
    - Use the trained Diffusion Model to encode frames into a compressed representation and record reconstruction errors.
5. **Train the CNN+LSTM Model**:
    - Prepare the CNN+LSTM with extracted features and reconstruction errors as inputs.
    - Train the model using the specified criterion (`BCEWithLogitsLoss` for binary classification).
    - Regularly evaluate on the validation set and adjust the learning rate using `ReduceLROnPlateau`.
6. **Save the Best Model**:
    - After each epoch, evaluate performance and save the model checkpoint if it achieves a new best F1 score.

4.2 Inference Steps with Trained Model

1. **Load the Trained Model**:
    - Load the CNN+LSTM model weights from the saved checkpoint file (`best_model.pth`).
2. **Run Predictions on New Videos**:
    - Use `VideoDataset` to load frames from a new video.
    - Preprocess and transform frames to the required size and normalization.
    - Extract features using the Diffusion Model, then pass these features to the CNN+LSTM model for prediction.
3. **Output**:
    - Predictions from the CNN+LSTM model indicate the likelihood of each frame sequence falling into the target class (e.g., anomalous or normal).
