Problem-2 Report

27 October 2024 15:35

Report on Diffusion Model and CNN Pipeline for Feature Extraction and Reconstruction Error Detection

1. Introduction

In this problem, we aim to design and implement a machine learning pipeline that utilizes a **Diffusion Model** for identifying reconstruction errors in data. The extracted features from this model will then be analyzed through a **Convolutional Neural Network (CNN)** combined with **Long Short-Term Memory (LSTM)** architecture.
The primary objective of this given problem statement is to utilize a diffusion model to capture reconstruction errors in video frames, which serve as valuable features for subsequent analysis. These extracted features are then fed into a CNN to perform classification tasks, such as anomaly detection or video categorization.

2. Architecture and Design

a) Diffusion Model

The diffusion model is a generative model that learns to gradually add noise to an image and then reverse the process to reconstruct the original image. By analysing the discrepancies between the original and reconstructed images, we can extract valuable features.
The diffusion model consists of two main components:

- Encoder: A series of convolutional layers with ReLU activations and Batch Normalization to extract features from the noisy input image.
- Decoder: A series of transposed convolutional layers to reconstruct the original image from the latent representation.

b) CNN+LSTM Model

- CNN Architecture**:** The CNN processes the features extracted from the diffusion model. It employs multiple convolutional and pooling layers, culminating in a global average pooling layer, which condenses the features into a fixed size.
- LSTM Integration**:** While the current implementation focuses solely on CNN, integrating LSTM can enhance temporal feature extraction, particularly useful when processing sequences of video frames.

The CNN architecture typically consists of:

- Convolutional Layers: Extract spatial features from the input data.
- Pooling Layers: Reduce the spatial dimensions of the feature maps.
- Fully Connected Layers: Classify the extracted features into different categories

c) Data Handling

The video dataset is loaded, pre processed, and divided into sequences of frames. Each frame is resized and normalized to ensure consistent input to the models.

Dataset : https://drive.google.com/drive/folders/1Im5ahsdOA1xltS0bvNDCIWjz3GTnzddu?usp=sharing

d) Pipeline Integration

The pipeline integrates the diffusion model and the CNN+LSTM as follows:

- Feature Extraction: The diffusion model processes each video frame to extract features, which are essentially the reconstruction errors.
- Feature Input to CNN+LSTM: The extracted features are fed into the CNN+LSTM model as input.
- Classification: The CNN+LSTM processes the features and outputs a classification prediction.

3.Justification for Model Selection

- Diffusion Model

The diffusion model is chosen for its unique capability to learn representations by focusing on reconstruction errors. By adding noise to the input data, the model can discern between normal and anomalous data effectively. This method is particularly beneficial for detecting subtle changes or anomalies within the video frames, which might not be readily apparent.

- CNN+LSTM Architecture

The combination of CNN and LSTM allows for powerful feature extraction from spatial and temporal data. CNNs excel at processing images and extracting spatial hierarchies, while LSTMs are designed to capture temporal dependencies, making this architecture suitable for video analysis tasks, such as anomaly detection and classification.

<u>4.Process Overview</u>

A. Data Loading and Pre-processing
    - Video Loading: The pipeline begins by loading the video data, where each video is treated as a series of sequential frames. OpenCV is used for loading frames, which allows for efficient handling of video data.
    - Frame Extraction: Frames are extracted at a set interval to capture essential information without overwhelming the model with unnecessary frames. The frames are resized to a standard dimension (e.g., 224x224 pixels), which standardizes input size for the model and reduces computational load.
    - Normalization: Each frame is normalized by scaling pixel values to the [0, 1] range, a common preprocessing step for neural networks, particularly in computer vision. This enhances the stability of the model training and inference phases, ensuring that all features are within a similar range.
    - Sequence Construction: Sequences of frames are constructed to represent temporal information. For instance, a set sequence length of 16 frames captures movement patterns across a small time window, enabling the model to understand temporal dependencies.

B. Feature Extraction
    - Diffusion Model for Reconstruction and Feature Extraction:
        - The diffusion model takes each frame as input and generates two outputs: (1) a reconstructed version of the frame and (2) a high-level feature map of the frame.
        - The reconstruction error is leveraged here as an indicator of irregularities, such as unusual patterns or anomalies, within the data. Frames with higher reconstruction errors may correspond to abnormalities or deviations, which the CNN-LSTM model can analyze further.
    - Noise Injection: A diffusion model uses a noise-injection mechanism during the feature extraction process, which simulates slight perturbations to understand the stability of each frame’s reconstruction. This also enhances the model’s robustness by providing it with varied perspectives of each frame, allowing for better discrimination of abnormal patterns.
    - Encoder-Decoder Architecture:
        - The diffusion model uses an encoder-decoder structure, where the encoder compresses input frames into a latent feature representation (feature map), and the decoder attempts to reconstruct the original frame.
        - By analyzing the difference between the input and reconstructed frames (i.e., the reconstruction error), the model detects discrepancies that signify abnormal or unexpected patterns within the frame.
    - Feature Map Generation: The encoded representations (feature maps) are then collected across the sequence, representing the essential features extracted from each frame for further processing by the CNN-LSTM model.

C. Prediction
    - CNN for Feature Interpretation:
        - The CNN component processes the feature maps output by the diffusion model. It captures the spatial patterns within each frame’s features, effectively identifying complex visual characteristics associated with anomalies or normal patterns.
        - This CNN architecture is specifically designed with several convolutional layers, batch normalization, and pooling operations to extract high-level spatial patterns while reducing spatial dimensions.
    - LSTM for Temporal Dependency Modeling:
        - Following the CNN’s extraction of spatial features, an LSTM layer processes the sequential features, analyzing dependencies across frames. This step captures temporal correlations, allowing the model to discern anomalies based on both spatial and temporal patterns.
        - The LSTM's recurrent nature makes it well-suited for detecting gradual changes, abrupt transitions, or sustained anomalies across time, all of which are essential in applications like video anomaly detection.
    - Binary Classification Output:
        - The final layer of the CNN-LSTM model is a binary classifier that outputs either a 1 or 0 for each sequence. Here, 1 represents the presence of an anomaly (irregular or abnormal pattern detected), while 0 represents normal sequences.
        - This binary output simplifies the decision-making process for downstream applications, allowing the model to act as a flagging mechanism in real-time anomaly detection systems.

D. Prediction Visualization
    - Prediction Visualization: For interpretability, predictions I plotted over time to visualize the model's decisions. This visualization helps assess the frequency and distribution of anomalies within a given video, aiding in further tuning and validation of the pipeline's effectiveness.

5.Steps to replicate the training process

A. Environment Setup:
- Ensured all required libraries are installed and configured in the working environment.

B. Data Preparation:
- Objective: Prepare the video data required for anomaly detection testing.
- Steps:
    1. Downloaded a video file.
    2. Stored the video file in a local directory. The video path is specified as:
       (video_path = "/content/Hyderabad City _ Dallas Center Road and more@Hitech City.mp4")
    3. Ensured that video file is accessible and compatible with OpenCV for frame extraction.

C. Model Initialization
- Objective: Set up the model pipeline that will handle feature extraction and prediction.
- Steps:
    1. Instantiated the Pipeline class by passing the path to your video file. This step initializes the video processing pipeline, including the diffusion model and CNN components.
       (pipeline = Pipeline(video_path))
    2. Now, the model is ready to process frames from the video and generate predictions.

D. Inference Execution
- Objective: Perform inference on the video to detect potential anomalies.
- Steps:
    1. Used the inference_on_video method of the Pipeline class to analyze the video and generate predictions for each frame.
    2. The predictions are binary outputs (0 or 1) indicating the presence or absence of anomalies.
       (predictions = pipeline.inference_on_video())
    3. This method returns a list of predictions, where each entry corresponds to a frame or sequence of frames in the video.

6.Conclusion & Results

The code successfully integrates a diffusion model for reconstruction error detection with a CNN+LSTM architecture for video analysis tasks.
Result : Since We are not training the model so, there will be no true samples. Hence the Accuracy: 0.0000, Precision: 0.0000, Recall: 0.0000, F1 Score: 0.0000

Outputs : Problem-2 - Output.pdf

Source Code : https://colab.research.google.com/drive/1jzQyJl-wxer88R0XbEhAegp3VojUkVLP?usp=sharing