Multi-Object Tracking with Threat Analysis System

Executive Summary

This platform integrates computer vision and natural language processing to provide an end-to-end object tracking solution with generation and threat analysis capability. The solution leverages the newest deep learning models for detection, tracking, description generation, translation, and threat analysis to provide a robust platform for security and surveillance solutions.

System Architecture

Core Components

Detection & Tracking Module

- YOLOv8 object detection with BOT-SORT tracking algorithm
- Handles occlusions of up to 60 frames
- Prevents ID switching through IoU matching
- Supports continuous tracking mode

Multilingual Captioning

- BLIP model generates English descriptions
- Helsinki-NLP translation model translates to Arabic
- Timestamped data stored in tracking history

Threat Analysis

- BERT-based classifier examines descriptions
- Multi-category assessment: violence, genocide, hate speech
- Computes cumulative risk indices with sensitivity weights
- Identifies disturbing words/phrases in material

Web API

- Flask-based REST API
- Ngrok for remote access
- Asynchronous model loading
- Supports video upload and frame-by-frame analysis

Fine-tuned Model

Overview

The goal of the fine-tuning procedure is to modify a language model that has already been trained, like BERT, for a particular task. Here, the model is being improved to categorise violent acts, genocide, and hate speech in text. This is accomplished by adding a custom classification head (a linear layer) that generates probabilities for various categories to a pre-trained BERT model.

Model Architecture

PyTorch and the transformers library are used in the construction of the AggressiveBertClassifier class. The BertModel.from_pretrained() function loads the pre-trained BERT model (bert-base-uncased), which forms the basis of the model. A linear classification layer (self.classifier) is then applied to the BERT model's output, mapping the 768-dimensional BERT representation to the

Training Model

The fine-tuning procedure involves training the classifier on a labeled dataset that contains instances of aggressive behavior categories such as violence, genocide, and hate speech. Key steps in fine-tuning include:

- **Optimizer**: Typically uses Adam or AdamW for optimization.
- **Learning Rate**: Fine-tuning is done with a low learning rate, often in the range of 1e-5 to 5e-5.
- **Training Epochs**: Fine-tuned for 3-4 epochs with early stopping to prevent overfitting.
- **Loss Function**: Cross-entropy loss is used, with softmax applied to the output logits to generate probabilities.

Categories and Thresholds

The classifier predicts multiple categories:

- **Violence**: 7 classes.
- **Genocide**: 5 classes.
- **Hatespeech**: 3 classes.

Each category uses thresholds to determine the severity level, with higher weights assigned to more severe predictions.

Model Evaluation

- **Accuracy**: The overall prediction accuracy is measured.
- **Confusion Matrix**: Used to assess the model's performance across all categories.

- **Precision, Recall, F1-Score**: These metrics are calculated for each category to evaluate the model's ability to distinguish aggressive behaviors.
- **Risk Scoring**: The model's ability to assign an appropriate risk score based on the severity of predictions is evaluated.

Classification on Dataset

The system implements a multi-label, multi-category classification approach specifically designed for content moderation and threat assessment.

Dataset Overview

The model is trained on a dataset of labeled text samples categorized into various types of aggressive behavior such as violence, hate speech, and genocide rhetoric. These datasets can include:

- **Violence**: Texts that describe different levels of violence, ranging from mild threats to explicit violent speech.
- **Hatespeech**: Texts containing discriminatory language, including hate speech, racial slurs, and derogatory remarks.
- **Genocide**: Texts advocating or describing genocidal actions or ideologies.

The dataset is structured as:

- **Text**: The actual text containing the speech or language to be classified.
- **Labels**: Each text is associated with one or more labels, corresponding to the categories defined in the model (e.g., violence, hate speech).

Data Preprocessing

- **Text Tokenization**: Text is tokenized using BERT's tokenizer (BertTokenizer), which splits the text into subword units and converts them into token IDs compatible with BERT.
- **Padding and Truncation**: Inputs are padded to a maximum length (typically 256 tokens), and truncation is applied to ensure uniform input size across all samples.
- **Attention Masks**: Attention masks are generated to indicate which tokens are real (1) and which are padding (0).

Technical Implementation Highlights

Object Tracking with Occlusion Handling

```
def handle_occlusion(track_id, detections, frame_time):
    # Maintains object identity when briefly not visible
    # Records occlusion events in the activity history
```

The system sustains object position and identity even through short-term occlusions, thus enabling continuous tracking in the midst of cluttered scenes containing many objects.

Multilingual Description Generation

```
def generate_description(frame, obj_id):
   # Extracts object from frame
   # English caption generated using BLIP
   # Translated into Arabic by neural machine translation
   # Returns timestamped bilingual descriptions
```

Objects are automatically described in both English and Arabic, with descriptions updated periodically to capture changes in appearance or context.

Continuous Tracking and Summary Generation

```
def generate_tracking_summary(track_id):
   # Aggregates temporal descriptions
   # Divides activities by time
   # Reports object behavior over the whole tracking period
```

The system preserves a sequential account of object descriptions and systematically produces detailed summaries for the monitored objects.

Threat Analysis and Risk Assessment

```
def _calculate_risk(self, predictions: Dict) -> Dict:
   # Weights given to contributions from different threat categories
   # Calculates the standardized risk score
   # Provides a risk rating: MINIMAL, LOW, MODERATE, HIGH, CRITICAL
```

Threat analysis evaluates descriptions across various dimensions for troubling content, generating risk predictions and flagging possible security threats.

Performance Metrics

Current performance metrics show the system approaches real-time processing on modern hardware:

Processing Stage	Average Time
Object Detection	33ms
Image Captioning	210ms
Translation	21ms
Threat Analysis	11ms
Total Pipeline	275ms

Table 1: Performance Metrics for Processing Stages

API Functionality

The system provides REST API endpoints for:

- Video upload and processing
- Frame-by-frame analysis
- Object tracking initiation/termination
- Threat report generation

Implemented Optimizations

- **Asynchronous Model Loading**: Models are loaded in background threads without blocking application startup
- **Effective Occlusion Handling**: Maintains object identity without reprocessing in occlusions
- **Description Caching**: Avoids redundant captioning of stationary objects
- **Annotated Frame Management**: Disk/memory overflow avoided through automatic cleanup

Recommended Enhancements

- **Frame Skipping**: Process subset of frames when there is low motion
- **Parallelized Pipeline**: Run stages in parallel for reduced latency
- **Adaptive Occlusion Handling**: Adjust parameters based on scene complexity
- **ReID Integration**: Incorporate person re-identification for improved long-term tracking

Conclusion

The system effectively integrates multiple AI modules to create a comprehensive object tracking and analysis system. The modular design makes it easy to swap out components with better models as they are found. The current implementation provides a solid foundation for video-based threat detection applications requiring object tracking, description, and threat assessment.