Proje Dokümanı: Gerçek Zamanlı AI Tabanlı Silah/Bıçak ve Saldırgan Davranış Tespiti Sistemi

Genel Amaç:

Bu sistem, videodan gelen kareleri (frame) yapay zeka modelleriyle analiz ederek silah/bıçak veya saldırgan davranış tespiti yaptıktan sonra anlamlı video segmentlerine ayırmayı amaçlamaktadır. Gerçek zamanlı olarak Redis, Kafka ve Docker altyapısı kullanılarak dağıtık bir mimari kurulmuştur. Bu sayede yüksek performans, ölçeklenebilirlik ve sürdürülebilirlik sağlanmıştır.

Mimari Bileşenler ve Detaylı İşleyiş:

1. Frame Splitter

- **Görev:** Video dosyasını karelerine böler.
- Örneğin, 30 FPS (saniyede 30 kare) hızında çalışan sistem, bir dakikalık videoyu 1800 ayrı kareye böler.
- **Her kare için:**
    - Benzersiz bir frame_id oluşturulur.
    - Frame'in alındığı zaman (timestamp) kaydedilir.
    - Frame görüntüsü Base64 formatında encode edilir.
    - İlk aşamada isDetected bilgisi False olarak işaretlenir.
- Oluşturulan kare bilgileri Redis veritabanına frame:{frame_id} anahtarıyla kaydedilir ve 5 dakika süreyle saklanır (TTL).
- Ayrıca Kafka mesaj kuyruğuna AI Unit için gerekli metadatalar (frame_id, timestamp) gönderilir.

2. Redis (In-Memory Data Store)

- Redis hızlı ve geçici veri depolama için kullanılır.
- Her frame verisi frame:{frame_id} anahtarıyla saklanır.
- Redis'teki veriler geçici olup, TTL ile otomatik silinir.
- Frame sıralamasını korumak için Redis üzerinde ayrıca bir listede (frame_queue) tutulur.

3. AI Unit (Kafka Consumer / Redis Reader)

- Kafka’dan frame_id bilgisini dinleyerek Redis’ten ilgili kare görüntüsünü çeker.

- Bu görüntüyü yapay zeka modeliyle analiz ederek silah/bıçak veya saldırgan davranış olup olmadığını belirler.
- Belirlene THRESHOLD_CONFIDENCE'a göre frameler etiketlenir.
- Tespit edilmiş durumunun kare çerçeve bilgisi metadataya eklenir.
- Elde edilen sonuçları Kafka üzerinden metadata olarak gönderir.
- **Örnek mesaj formatı böyle olabilir:**

{

"frame_id": 1234,

"timestamp": "00:00:41.133",

"prediction": true,

"confidence": 0.92

}

4. Kafka (Message Broker)

- Kafka, hızlı ve gerçek zamanlı mesajlaşma sağlar.
- AI Unit'ten gelen tespit sonuçlarını Decision Unit'e taşır.
- Böylece modüller arasında kopmadan iletişim sağlanır ve her modül bağımsız şekilde ölçeklenebilir.

5. Decision Unit (Kafka Consumer + Redis Reader)

- Kafka'dan gelen AI tespit sonuçlarını dinler ve Redis'ten ilgili karelerin verilerini çeker.
- Bu karelerden gelen tespit sonuçlarını analiz ederek video segmentlerini oluşturur.
- Tespit edilen ardışık karelerde (DETECTION_THRESHOLD) silah/bıçak veya saldırgan davranış bulunması halinde, bu karelerden önceki 3 saniyeyi başlangıç olarak kabul ederek segment oluşturur.
- Yoğunluk analizine göre, 2 saniyelik pencere içinde tespit sıklığı belirlenen eşik değerin altına düşerse (SPARSITY_THRESHOLD), segment sona erer.
- **Örnek segment çıktısı böyle olabilir:**

{

"start_frame": 720,

"end_frame": 1035,

"start_timestamp": "00:00:24.000",

"end_timestamp": "00:00:34.500",

"reason": "Weapon Detected"

}

6. Web Arayüz

- Kullanıcı dostu bir arayüz sağlar:
    - Videolar yüklenebilir ve işlenebilir.
    - İşlenen videolar ve belirlenen segmentler listelenir.
    - Segmentlerin görüntülenmesi ve indirilmesi sağlanır.
- Backend olarak **FastAPI**, frontend olarak **Vue** veya **React** kullanılabilir.

7. Docker Kompozisyonu

- Tüm bileşenler Docker container'ları içinde ayrı ayrı çalıştırılır.
- Bu, projenin kolay dağıtımını, taşınabilirliğini ve ölçeklenebilirliğini sağlar.
- Örnek Docker Compose Yapısı:

services:

frame_splitter:

redis:

kafka:

kafka-init:

ai_unit:

decision_unit:

web_interface:

Ayrıntılı Veri Akışı Örneği:

- Bir video sisteme yüklenir ve Frame Splitter videoyu karelere bölerek Redis'e yükler ve Kafka'ya metadata gönderir.
- AI Unit, Kafka'dan frame_id alarak Redis'ten frame görüntüsünü çeker, yapay zeka ile işleyip Kafka'ya sonucu gönderir.
- Decision Unit, Kafka'dan AI sonuçlarını alıp Redis'ten kareleri çekerek olay segmentlerini belirler.
- Belirlenen segmentler Web arayüzüne aktarılır, kullanıcılar tarafından izlenebilir veya indirilebilir hale gelir.

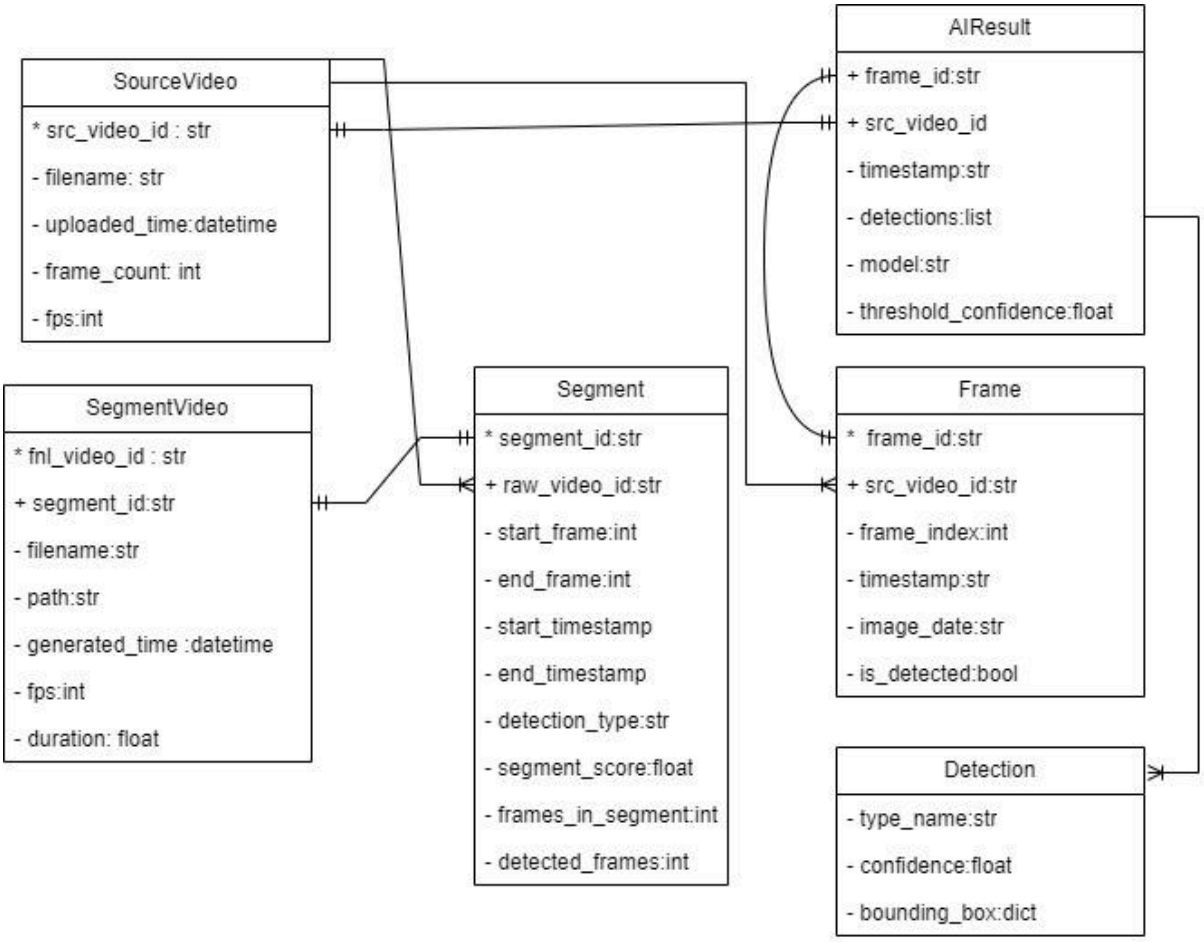
Karar Algoritması Parametreleri:

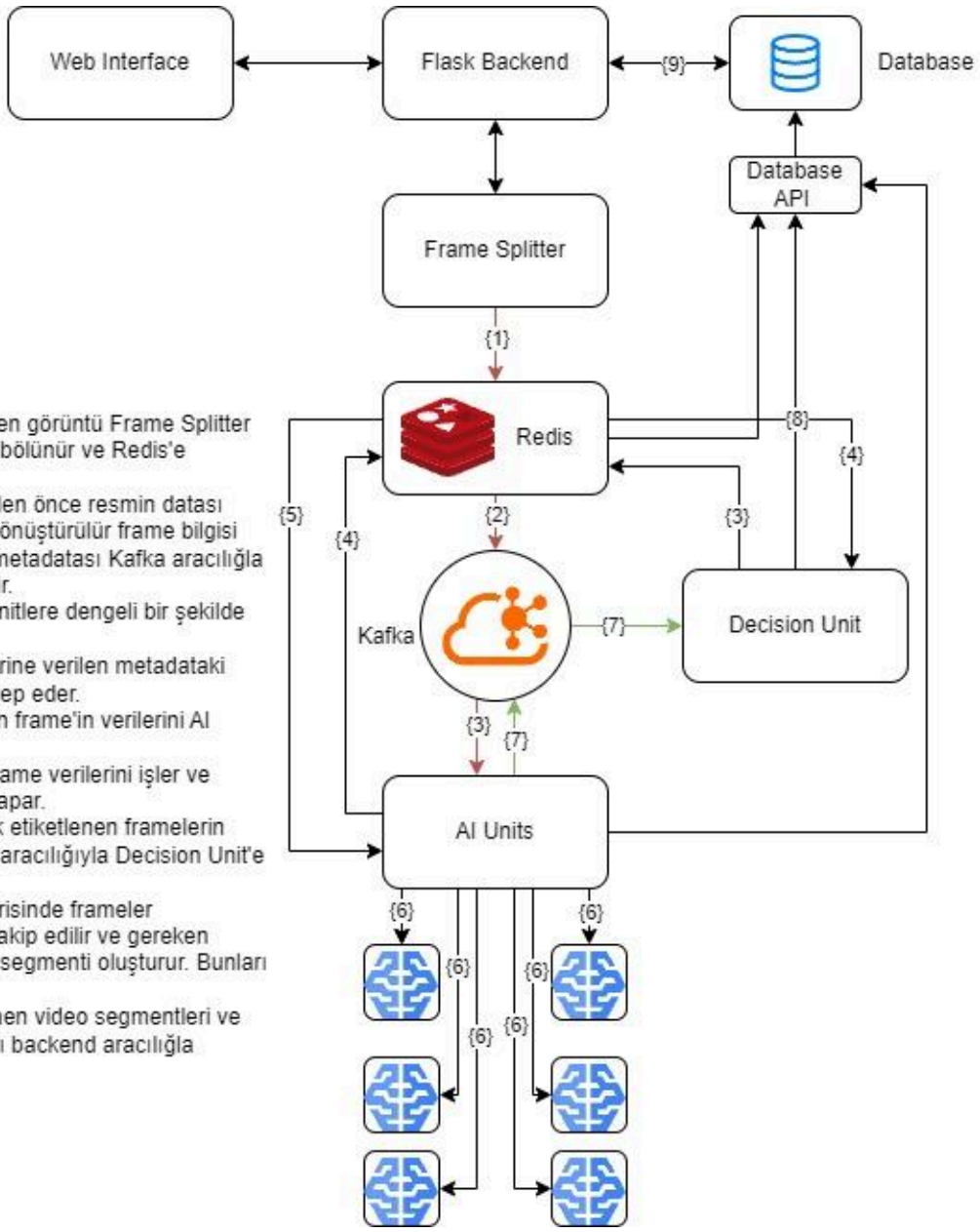
- FPS: Kare hızı (örn. 30)
- DETECTION_THRESHOLD: Olay başlatmak için ardışık minimum pozitif tespit sayısı (örn. 10)
- THRESHOLD_CONFIDENCE: Kare içerisinde tespit edilen görüntünün güven seviyesine göre isDetected, true veya false olarak işaretlemek için kullanılır.
- PRE_EVENT_BUFFER: Olay öncesi ek süre (örn. 3 saniye)
- POST_EVENT_WINDOW: Olay sonrası yoğunluk kontrolü yapılan zaman penceresi (örn. 2 saniye)
- SPARSITY_THRESHOLD: Bu pencere içinde maksimum izin verilen tespit sayısı (örn. 2)

Avantajlar:

- Redis kullanımı ile hızlı ve düşük gecikmeli geçici veri saklama.
- Kafka ile modüller arasında kesintisiz iletişim ve mesaj yönetimi.
- Modüler yapı sayesinde kolay ölçeklendirme ve bakım.
- Docker altyapısı ile hızlı dağıtım ve sürdürülebilirlik.
- Etkili karar algoritması sayesinde yanlış pozitiflerin azaltılması ve gerçek tehditlerin hızlı belirlenmesi.

ER Diagram:


SourceVideo
* src_video_id : str
- filename: str
- uploaded_time:datetime
- frame_count: int
- fps:int
AIResult
+ frame_id:str
+ src_video_id
- timestamp:str
- detections:list
- model:str
- threshold_confidence:float
SegmentVideo
* fnl_video_id : str
+ segment_id:str
- filename:str
- path:str
- generated_time :datetime
- fps:int
- duration: float
Segment
* segment_id:str
+ raw_video_id:str
- start_frame:int
- end_frame:int
- start_timestamp
- end_timestamp
- detection_type:str
- segment_score:float
- frames_in_segment:int
- detected_frames:int
Frame
* frame_id:str
+ src_video_id:str
- frame_index:int
- timestamp:str
- image_date:str
- is_detected:bool
Detection
- type_name:str
- confidence:float
- bounding_box:dict



1: Backend'den gelen görüntü Frame Splitter tarafından karelere bölünür ve Redis'e gönderilir.
2:Redis'e eklenmeden önce resmin datası base64 formatına dönüştürülür frame bilgisi ayarlanır.Frame'in metadatası Kafka aracılığla AI Unitlere gönderilir.
3: Metadatalar AI Unitlere dengeli bir şekilde dağıtılır.
4: AI Unitler kendilerine verilen metadataki bilgileri Redisten talep eder.
5: Redis talep edilen frame'in verilerini AI Unitlere iletir.
6: AI unitler gelen frame verilerini işler ve etiketlendirmesini yapar.
7: "Detected" olarak etiketlenen framelerin metadataları Kafka aracılığıyla Decision Unit'e gönderilir.
8: Decision Unit içerisinde frameler algoritmaya kadar takip edilir ve gereken durumlar için video segmenti oluşturur. Bunları Database'e yazar.
9: Database'e eklenen video segmentleri ve onların metadataları backend aracılığla arayüze gönderilir.