Pusan National University
Computer Science and Engineering
Technical Report 2025-09

네트워크 상태와 영상 특성 기반 실시간 적응형 비디오 스트리밍

정보컴퓨터공학부 202155515 김남희

정보컴퓨터공학부 202155553 박은재

지도교수 김종덕

목 차

1. 서론

1.1. 연구 배경

현대 사회의 미디어 소비 패러다임은 OTT(Over-The-Top) 플랫폼을 중심으로 급격하게 재편되고 있다. YouTube, Netflix와 같은 스트리밍 서비스는 더 이상 단순한 선택지를 넘어, 현대인의 일상에 깊숙이 자리 잡은 필수적인 여가 활동으로 부상했다.

2024년 문화체육관광부의 「국민문화예술 여가활동 조사 결과 발표」에 따르면, TV 시청이 62.8%, 온라인/모바일 동영상 시청이 48.0%로 가장 많이 참여하는 세부 여가 활동으로 집계되었다. 이는 사람들이 산책이나 외식 같은 활동적인 여가보다 TV 및 동영상 시청, 인터넷 검색 등 온라인 여가를 더 많이 즐기고 있음을 보여준다. 이러한 변화는 인터넷 통신망의 발달과 온라인 콘텐츠의 기하급수적인 증가에 따른 것으로 분석된다.

<가장 많이 참여한 세부 여가활동(1~5순위 복수응답)>

(단위: %)

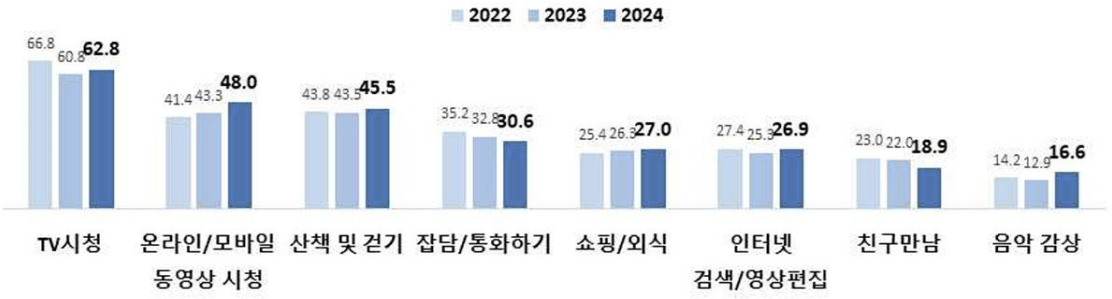


그림 1. 문화체육관광부, 「2024 국민문화예술 여가활동 조사 결과 발표」, 2024.

국내뿐만 아니라 글로벌 시장에서도 스트리밍 시장의 성장을 확인할 수 있다. Nielsen의 「2025년 5월 The Gauge™ 보고서」에 따르면 2025년 5월 스트리밍 서비스의 TV 시청률이 44.8%를 기록하며 사상 최고치를 달성했고, 지상파(20.1%)와 케이블(24.1%) 시청률의 합산을 처음으로 넘어섰다. 스트리밍 시청률은 2021년 이후 71% 증가한 반면 지상파 방송 시청과 케이블 시청은 각각 21%, 39% 감소했으며 이는 스트리밍이 미디어 시장의 주류가 되었음을 증명한다.

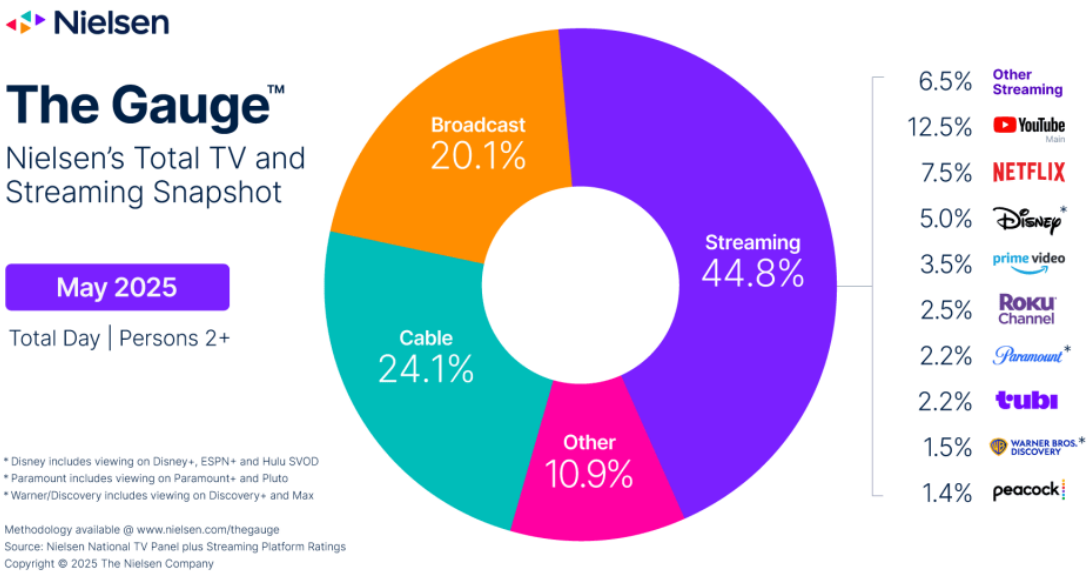


그림 2. Nielsen,「Streaming reaches historic TV milestone, eclipses combined broadcast and cable viewing for first time」, The Gauge™, 2025.5

이처럼 폭발적으로 증가하는 스트리밍 수요는 필연적으로 사용자들의 기대수준을 높인다. 사용자들은 이제 언제 어디서든, 심지어 이동 중인 불안정한 네트워크 환경에서도 끊김 없는 고화질 영상을 기대한다.

따라서 본 과제는 이러한 요구에 부응하여 기존 스트리밍 방식으로 인해 발생하는 기술적 과제를 해결하고, 변화하는 네트워크 환경과 영상 자체의 특성을 실시간으로 분석하여 이를 기반으로 사용자에게 최적의 시청 경험을 제공하는 적응형 스트리밍 서비스를 개발하고자 한다.

1.2. 기존 문제점

- 비효율적인 비트레이트 관리

기존 스트리밍은 클라이언트가 네트워크 상황에 맞춰 적절한 품질의 세그먼트를 요청하고, 서버에서 전달받아 재생한다. 예를 들어, 네트워크가 불안정해지면 비트레이트와 해상도를 낮춰 영상을 재생하게 되는데, 이때 모든 세그먼트를 영상 특성에 상관없이 일괄적으로 품질을 변경한다. 일반적으로 움직임이 많거나 복잡한 영상은 그렇지 않은 영상보다 더 많은 비트레이트가 필요한데, 일괄적으로 비트레이트를 변경하는 과정에서 복잡한 영상은 비트레이트가 부족해지고 이로 인해 품질 저하가 심화되는 현상이 발생한다.

반대로, 네트워크 상태가 안정화되면서 비트레이트와 해상도를 높여 재생하는 경우에는 단순한 영상이 필요 이상으로 높은 비트레이트를 할당 받게 된다.

따라서 비트레이트는 영상 특성과 네트워크 두가지 측면으로 분석하고 배분되어야 한다. 본 과제는 인간 시각 시스템(HVS)을 통해 비트레이트를 얼마나 배분할 것인지 결정하고, 이를 영상 특성 기반 인코딩 파라미터 최적화 모델을 통해 구현한다.

- 서버 스토리지 부족

세그먼트를 미리 인코딩하고 사용자가 요청할 때 클라이언트에게 전달해주는 기존 스트리밍 방식은 조합에 따라서 (해상도 * 비트레이트 * 프레임레이트) 개수의 세그먼트를 서버에 저장해둔다. 지나치게 많은 세그먼트를 서버에 저장해두면 스토리지가 부족해지는 상황이 발생할 우려가 있다.

이를 해결하기 위해 영상의 특성에 최적화된 인코딩 파라미터로 미리 인코딩하고, 네트워크 상황을 고려해 해상도로 세그먼트를 나누어 놓으면 서버 스토리지나 리소스 낭비를 최소화할 수 있을 것으로 본다.

- 잦은 품질 변경으로 인한 QoE 저하

기존의 적응형 스트리밍 방식은 네트워크 상황이 변하면 해상도나 배트레이트를 변경하여 세그먼트의 품질을 변경한다. 만약 사용자의 네트워크 환경이 지속적으로 변경되는 경우, 시청하고 있는 영상의 품질 또한 빈번하게 변경되는데 이러한 잦은 품질 변경은 사용자의 경험 품질(QoE) 저하로 이어진다.

이를 해결하기 위해 본 과제에서는 영상 특성에 최적화된 인코딩 파라미터로 고정된 세그먼트를 준비하여 비트레이트를 절약하고, 최소한의 품질 변경을 시도하여 사용자 경험 품질의 저하를 최소화한다.

1.3. 연구 목표

- 영상 특성 기반 인코딩 파라미터 최적화 모델

장면 복잡도를 포함한 여러 영상 특성을 추출한 다음, 이를 통해 인코딩 파라미터를 최적화하는 AI 모델을 구현한다. AI 모델은 장면의 복잡도에 따라 화면의 품질을 적절히 조절하여 사용자 체감 화질 저하는 최소화하되 비트레이트를 효율적으로 관리하는 것을 목표로 한다.

- 적응형 스트리밍 시스템

기존의 DASH 시스템을 활용하여 네트워크 상황에 따라 적절한 해상도로 세그먼트를 재생할 수 있도록 여러 해상도 조합으로 세그먼트를 미리 인코딩하여 준비한다. 기존 방식에 비해 적은 수의 조합으로 세그먼트를 서버에 저장하기 때문에 스토리지 관리에 용이할 것으로 본다.

- 웹페이지 개발 및 배포

시청자들이 적응형 스트리밍을 시청할 수 있는 웹사이트를 개발하고 배포한다. 회원가입 및 로그인, 회원 정보 변경, 시청 기록 조회 및 삭제, 즐겨찾기 서비스 등의 기능을 포함하여 사용자들의 편의성을 높인다.

1.4. 기대 효과

- 사용자 경험 (QoE) 향상

각 장면의 시각적 특성을 고려한 비트레이트 할당으로, 시청자는 네트워크 조건이 좋지 않더라도 체감 품질 저하를 덜 느끼고, 복잡한 장면에서 발생하는 아티팩트가 최소화된 고품질 영상을 경험할 수 있다.

- 대역폭 효율성 극대화

단순한 장면에서는 불필요한 비트레이트 할당을 줄여 대역폭을 절약하고, 절약된 대역폭을 다른 복잡한 장면에 재할당하여 전체적인 전송 효율성을 높일 수 있다.

- 파일 크기 최적화

각 장면의 특성에 맞는 효율적인 압출을 통해 전체 비디오 파일 크기를 줄여 저장 공간 절약 및 CDN(콘텐츠 전송 네트워크) 비용 절감 효과를 기대할 수 있다.

2. 연구 배경

2.1. 선형 콘텐츠와 스트리밍

지상파나 케이블 방송 등의 선형 콘텐츠는 일반적으로 브로드캐스트(broadcast) 방식으로 송출된다. 브로드캐스트는 송신자 측에서 특정 수신자를 지정하지 않고, 네트워크에 연결

된 모든 호스트에게 방송을 송출하는 특징을 가진다. 사용자는 이러한 특징 때문에 정해진 시간에 정해진 방송만 들을 수 있는 불편함을 겪었다.

이를 해결하기 위해 스트리밍은 주로 유니캐스트(unicast) 방식으로 데이터를 전송한다. 유니캐스트는 서버와 클라이언트 간의 1대1 통신으로 각 시청자에게 개별적인 네트워크 연결 및 데이터 전송을 요구한다. 이러한 스트리밍 방식은 사용자가 원하는 시간과 장소에서 자유롭게 영상을 시청할 수 있다는 장점을 지니지만, 초기 스트리밍의 경우 몇 가지 한계점을 가지고 있다.

스트리밍은 네트워크를 통해 시청자에게 영상을 전송하는데, 영상을 재생하기 위해 특정 최소 대역폭을 필요로 한다. 예를 들어, Full HD 영상은 최소 5Mbps, 4K UHD 영상은 최소 15Mbps의 대역폭을 권장한다. 그러나 동시 접속자 수가 급증하면 네트워크의 총 가용 대역폭이 모든 개별 사용자에게 필요한 비트레이트의 합계를 감당하지 못하게 된다. 이로 인해 각 사용자에게 할당되는 실제 가용 대역폭은 급격히 감소하게 되며, 시청자가 데이터를 버퍼에 충분히 할당하지 못해 버퍼링이 발생하게 된다.

무선 네트워크 환경에서는 지연, 유실, 프로토콜 오버헤드 등으로 인해 초기 지연 시간이 발생할 수 있으며, 이는 사용자가 원하는 속도로 영상을 전송받지 못하게 되어 버퍼링이나 화질 저하와 같은 서비스 품질(QoE) 저하로 이어진다.

2.2. 기존 스트리밍 방식과 DASH

스트리밍 방식은 크게 프로그레시브 다운로드와 RTMP/RTSP 스트리밍, 그리고 적응형 HTTP 스트리밍으로 나눌 수 있다.

프로그레시브 다운로드(Progressive Download)는 우리가 흔히 MP4 파일을 다운로드할 때 사용하는 방식으로, 비디오 파일을 웹 서버에 올려놓고 사용자에게 URL을 알려주면, 사용자는 파일을 다운로드하고 일정량 이상 다운로드가 완료되면 재생 할 수 있다. 전체 파일을 다운로드 받아야 하므로 대역폭 소모가 크고, 고정된 품질로만 다운받을 수 있다는 단점이 있다.

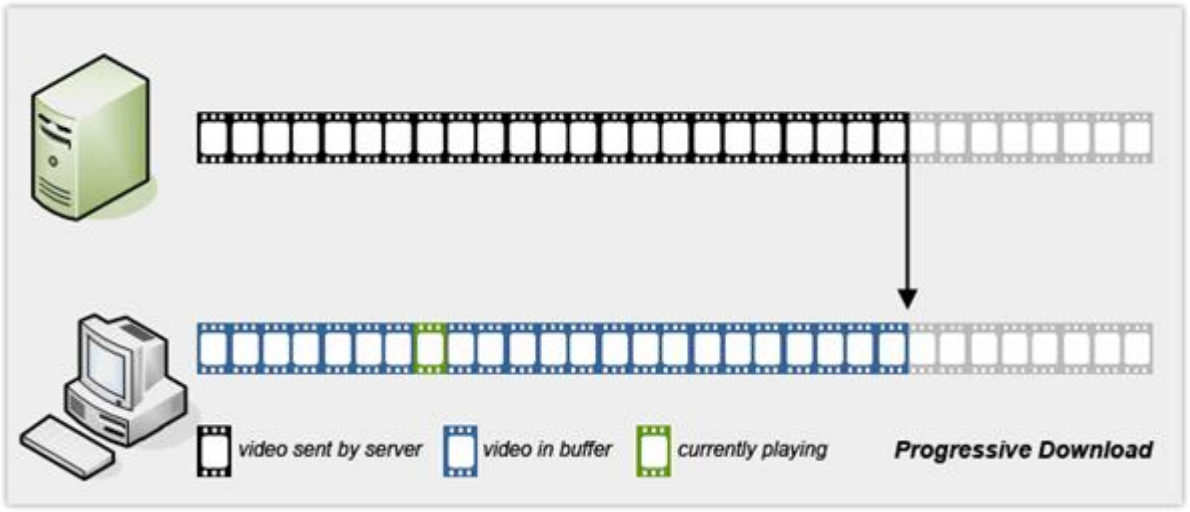


그림 3. Progressive Download

RTSP(Real Time Streaming Protocol)과 RTMP(Real Time Messaging Protocol)은 이러한 프로그레시브 다운로드의 약점을 보완하기 위해 만들어진 스트리밍 프로토콜이다. 이 방식은 전체 파일을 다운로드하는 것이 아니라 사용자가 시청하는 구간의 프레임만 전문 웹 서버에서 전송 받는다. RTMP는 재생 도중 비디오 품질을 변경할 수 있지만 전문 서버나 프로토콜이 필요하기 때문에 복잡성과 비용이 더 든다.

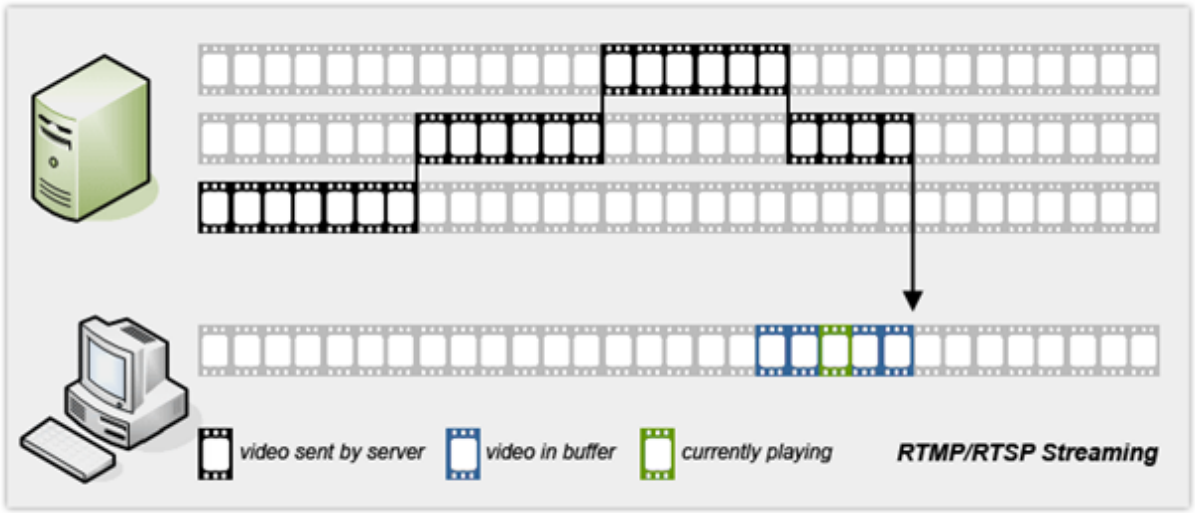


그림 4. RTSP/RTMP 스트리밍

적응형 HTTP 스트리밍(Adaptive HTTP Streaming)은 RTMP/RTSP 스트리밍의 대역폭 효율성 및 품질 전환 기능과 프로그레시브 다운로드는 전용 서버가 필요 없다는 편의성을 결합한 방식이다.

그림 5. Adaptive HTTP Streaming

MPEG-DASH(Dynamic Adaptive Streaming over HTTP)는 적응형 HTTP 스트리밍 기술의 대표적인 구현 표준으로, 서버가 여러 해상도와 비트레이트 조합으로 인코딩 된 영상 세그먼트를 미리 생성하여 저장해두고 클라이언트가 현재 네트워크 상황을 보고 예측하여, 어떤 세그먼트를 받을지 실시간으로 결정해서 재생하는 방식이다.

클라이언트는 영상 메타 정보가 담긴 MPD 파일을 통해 사용 가능한 해상도와 비트레이트 조합인 Representation을 확인한다. 각 Representation은 일정 품질을 유지하는 세그먼트들로 구성되어 있으며, 영상 재생 중에는 ABR 알고리즘이 네트워크 처리량과 버퍼 상태를 평가하여 가장 적절한 세그먼트를 선택해 재생한다.

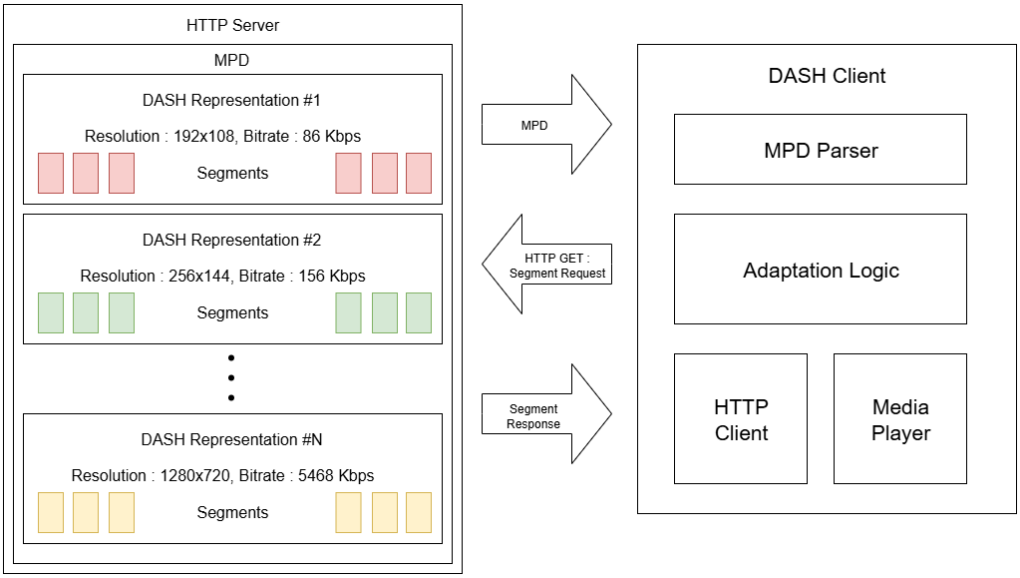


그림 6. DASH 구조

이러한 방식을 통해 DASH는 기존 스트리밍 방식이 가지고 있던 대역폭 비효율성, 고정된 품질, 특정 프로토콜 종속성 등을 극복하며 현재 스트리밍의 주요 기술로 자리잡았다.

2.3. 압축 방식과 인코딩 복잡도, 마스킹 효과

2.3.1. 공간 압축과 공간적 복잡도

공간 압축 (Spatial Redundancy)은 한 프레임 내에서 압축이 진행되는 인코딩 방식이다. 사람의 시각이 특정 영역의 데이터를 민감하게 반영하지 못하는 점을 이용하여 상관 관계가 높은 블록 영역을 압축한다.

공간적 복잡도가 높다는 것은 비디오 프레임에 질감, 가장자리, 미세 패턴과 같은 세부 정보가 많다는 것을 의미한다. 이러한 장면은 효율적으로 압축하기가 더 어렵기 때문에, 시각적 아티팩트 없이 압축하려면 더 많은 비트레이트가 요구된다.

인간의 시각 시스템(HVS)은 복잡한 질감 영역에서 압축 아티팩트(예: 블로킹, 링잉)에 덜 민감하게 반응하는 공간적 마스킹 효과를 가진다. 이는 복잡한 배경에 작은 아티팩트가 숨질 수 있기 때문이다. 따라서 일정 수준까지는 비트레이트를 줄여도 품질 저하가 크게 느껴지지 않을 수 있다.

2.3.2. 시간 압축과 시간적 복잡도

시간 압축 (Temporal Redundancy)은 연속된 프레임 간의 공통점을 찾아내어 유사한 블록에 대해서는 압축을 수행하지 않고 방향 좌표만을 설정하는 인코딩 방식이다.

시간적 복잡도가 높다는 것은 비디오 내에 빠르고 빈번한 움직임이나 활동이 많다는 것을 의미한다. 이러한 비디오는 인코딩 시 더 많은 리소스를 요구하며, 특히 움직임 추정 (Motion Estimation) 과정에서 계산 복잡도가 증가하게 된다.

인간의 시각 시스템은 움직임이 많은 장면에서 미묘한 변화나 아티팩트를 덜 감지하는 시간적 마스킹 효과를 보인다. 이 효과를 활용하면 비트레이트를 줄여도 주관적인 품질 저하 없이 압축 비디오 시퀀스의 비트레이트를 줄일 수 있다.

2.4. DASH의 한계 및 영상 특성 고려의 필요성

DASH(Dynamic Adaptive Streaming over HTTP)는 실시간 네트워크 상태를 기반으로 영상

의 해상도와 비트레이트를 조절한다. 그러나 영상 자체의 시간적(temporal) 및 공간적(spatial) 복잡도와 같은 고유한 시각적 특성을 고려하지 않아 비효율성을 초래하며, 네트워크 환경 변화 시 각 장면의 복잡도에 따라 두 가지 주요 문제점이 발생할 수 있다.

2.4.1. 복잡한 장면의 품질 저하 심화

움직임이 많고 변화가 빠른 고모션 영상 또는 세부 묘사나 질감이 복잡한 영상은 일반적으로 이러한 정보를 정확하게 표현하기 위해 더 높은 비트레이트가 요구된다. 비트레이트가 충분히 할당되지 않을 경우, 인코더는 제한된 비트 내에서 중요한 정보만을 유지하고 나머지는 손실시키면서 노이즈나 뭉개짐과 같은 압축 아티팩트를 유발할 수 있다. 기존 DASH 방식은 네트워크 환경이 불안정해질 경우, 영상 특성을 고려하지 않고 세그먼트의 품질, 즉 비트레이트를 일괄적으로 하향 조정하게 된다. 이로 인해 복잡한 장면의 비트레이트가 부족해져 품질 저하가 더욱 심화되는 문제가 발생하게 된다.

품질 저하를 완화하기 위해서는 복잡한 장면에도 충분한 비트레이트를 할당해야 한다. 그러나 네트워크 상황에 따라 비트레이트 조절이 필요하므로, 시청자의 눈에 품질 저하가 눈에 띄지 않을 정도의 비트레이트 수준을 유지하는 것이 중요하다. 이때 인간 시각 시스템(HVS)의 마스킹 효과를 활용할 수 있다. 예를 들어, HVS는 움직임이 많은 장면에서 미묘한 품질 저하를 덜 인지하는 시간적 마스킹 효과를 보이며, 복잡한 질감 영역에서는 압축 아티팩트에 덜 민감해지는 공간적 마스킹 효과가 나타난다. 이러한 시각적 인지 특성을 활용하여, 품질 저하가 눈에 띄는 임계점을 설정하고 그에 맞춰 비트레이트를 조절할 수 있다. 예를 들어, VMAF(Video Multimethod Assessment Fushion)와 같은 지각 기반 품질 지표를 일정 수준 이상으로 유지하면서 CRF(Constant Rate Factor) 값을 높여 비트레이트를 최적화할 수 있다.

2.4.2. 단순하고 정적인 장면에서의 비효율성

정적인 풍경 장면과 같이 변화가 적은 콘텐츠는 동적인 스포츠 중계 영상에 비해 낮은 비트레이트로도 높은 시각적 품질을 유지할 수 있다. 하지만 DASH에서 네트워크 환경이 개선될 경우, 정적인 장면도 동적인 장면과 동일하게 비트레이트를 상향 조정하여 전송한다. 이러한 방식은 정적인 장면에 필요 이상의 데이터가 할당되어 대역폭 낭비를 초래하게 된다.

이러한 대역폭 비효율성을 개선하기 위해서는 정적인 장면에 더욱 낮은 비트레이트를 할당하여 대역폭을 절약해야 한다. 하지만 이때 중요한 점은, 정적인 장면에서는 시각적으로 방해가 되는 요소가 적기 때문에, 시청자는 화면에 나타나는 모든 디테일에 집중한다

는 점이다. 이로 인해 비트레이트를 낮춰서 압축 아티팩트가 발생하면 시각적 마스킹 효과가 거의 없어서 시청자의 눈에 쉽게 인지될 수 있다. 예를 들어, 정적인 배경에 작은 블록 노이즈나 색상 밴딩이 생기면 움직임이 있는 장면에 비해 훨씬 더 눈에 띄는 것이 이 때문이다. 따라서 단순하고 정적인 장면에서는 비트레이트를 낮춰 대역폭을 절약해야 하지만, 동시에 시청자가 체감하는 품질 저하를 최소화하기 위해 정교한 비트레이트 최적화가 필요하다.

2.5. dash.js

dash.js는 MPEG-DASH 스트리밍을 재생하기 위한 참조 클라이언트 구현체로, JavaScript로 작성되어 있다. 이 라이브러리는 Media Source Extensions(MSE)을 지원하는 브라우저 환경에서 동작하며, MPEG-DASH 표준을 기반으로 적응형 스트리밍을 수행한다.

3. 연구 내용

3.1. 구성도

3.1.1. 전체 구성도

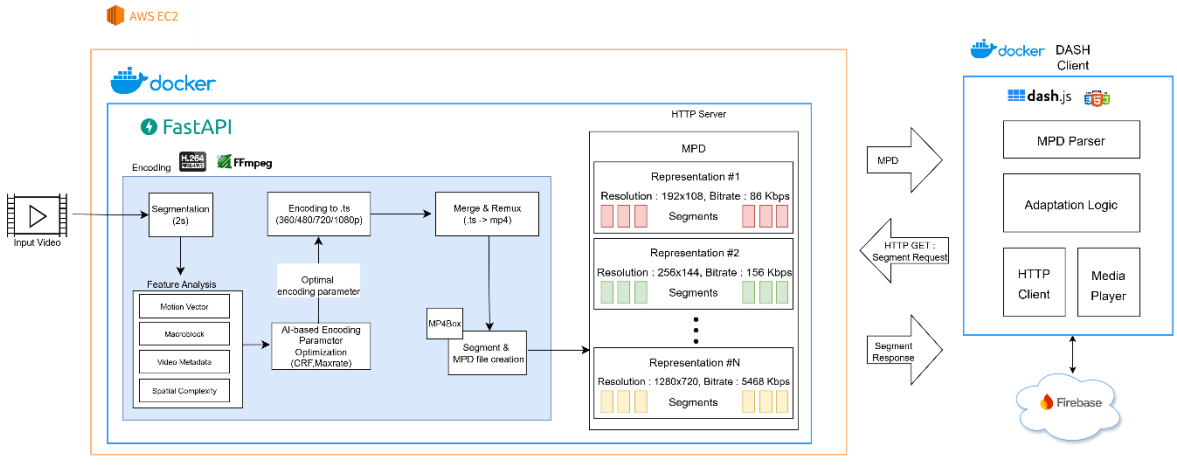


그림 7. 전체 구성도

1. 구성도는 서버 측과 클라이언트 측으로 구분
2. 기존 DASH에 영상 인코딩 최적화 모듈인 하늘색 부분을 추가해 영상 특성과 네트워크를 함께 고려하는 구조

3. 서버는 AWS EC2 인스턴스 상에 구축되고, 클라이언트는 사용자의 웹 브라우저에서 동작
4. 좌측의 영상은 사전에 서버에 업로드 된 콘텐츠
5. 사용자가 특정 영상을 시청하길 원할 때, 서버는 해당 영상을 처리하여 스트리밍

3.1.2. 서버측 구성도

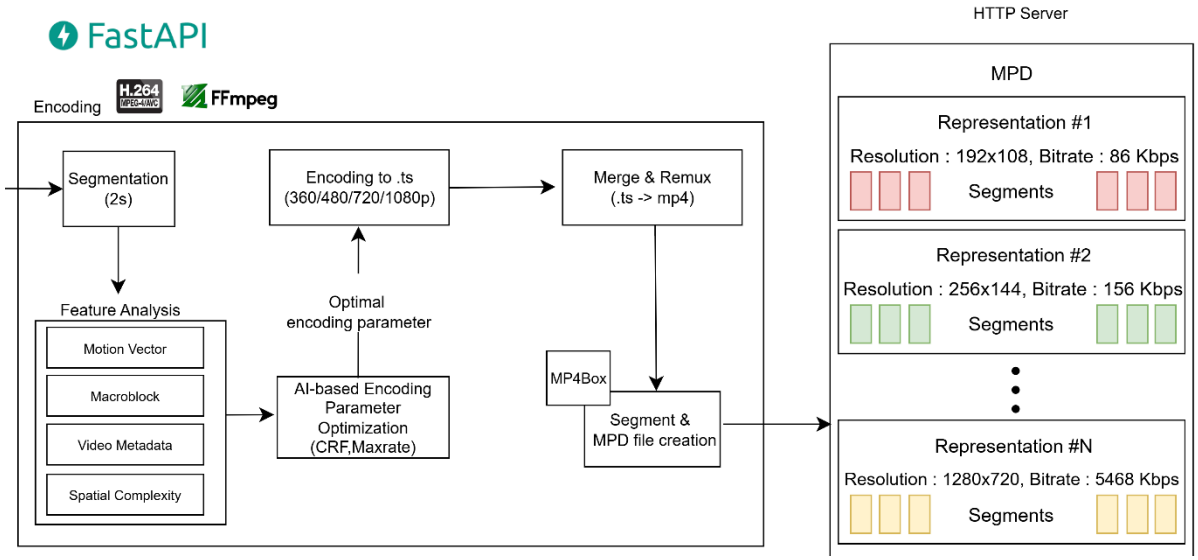


그림 8. 서버측 구성도

1. 세그먼트 분할
   A. 원본 영상을 2초 단위로 재인코딩하여 세그먼트 생성
   B. 각 세그먼트의 첫 프레임을 I-frame으로 강제해 경계 안정화
2. 세그먼트의 영상 특성 추출
   A. 모션 벡터, 매크로블록 분포, 영상 메타 데이터, 공간적 복잡도 추출
3. 영상 인코딩 최적화 모델
   A. 세그먼트의 영상 특성을 입력하여 최적화된 CRF, Max rate 예측
   B. CRF는 유동적으로 비트레이트를 조절하여 영상의 일정한 품질을 유지하도록 하며, Max rate는 갑작스러운 비트 전송량의 증가에도 버퍼링이 발생하지 않도록 최대 비트 전송량 제한
4. 최적화 인코딩 파라미터를 적용하여 세그먼트 인코딩
   A. 각 세그먼트를 최적화된 CRF, Max rate로 인코딩
   B. 해상도별로 각 세그먼트 인코딩 – 360p, 480p, 720p, 1080p
   C. 인코딩 된 파일은 .ts(MPEG-TS) 포맷으로 저장
5. 병합
   A. 해상도별 .ts 파일들을 이어 붙여 mp4 파일로 재포장

6. DASH 패키징
   A. MP4Box를 사용하여 manifest.mpd와 .m4s 생성
   B. -rap 옵션으로 세그먼트가 I-frame 경계에서 시작되게 보장
   C. Representation별 init segment를 MP4Box가 자동 생성
7. MPD 파일과 세그먼트가 서버에 저장
   A. MPD 파일은 세그먼트들의 정보를 담고 있으며, 클라이언트가 어떤 세그먼트를 요청할 지 판단하는 데 활용
   B. 세그먼트는 미리 최적화 파라미터로 인코딩 되어 클라이언트의 요청에 따라 FastAPI를 통해 HTTP 서버로 제공

3.1.3. 클라이언트측 구성도

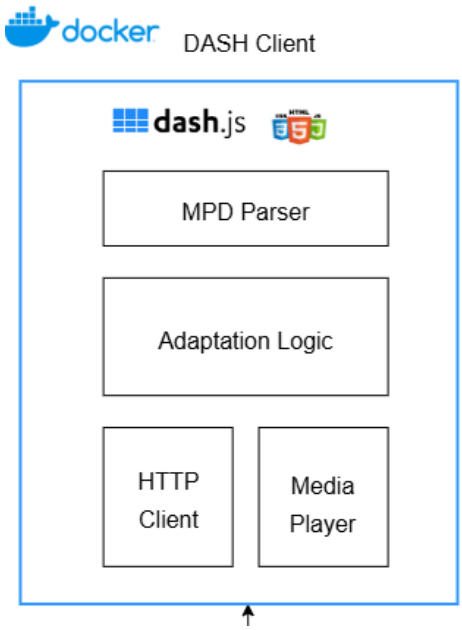


그림 9. 클라이언트측 구성도

1. dash.js 라이브러리 기반 영상 스트리밍 수행
   A. 서버에서 MPD파일 다운로드
   B. MPD파일을 파싱하여, 현재 사용 가능한 Representation 목록과 세그먼트 위치 등의 정보 추출
   C. dash.js 내부의 ABR 알고리즘이 네트워크 처리량과 버퍼 상태를 고려하여, 가장 적절한 품질의 세그먼트 선택
   D. 선택된 세그먼트는 URL을 통해 다운로드 되며, Media Source Extensions(MSE)을 통해 해당 세그먼트를 video buffer에 추가
   E. 세그먼트는 H.264로 압축된 인코딩 영상이므로, 브라우저는 이를 자동으로 디코딩

F. 디코딩 된 영상은 브라우저에서 재생됨

2. 세그먼트 구성 방식
   A. 본 시스템은 기존 DASH처럼 Representation 하나가 고정된 품질(해상도, 비트레이트)로 구성된 구조가 아니라, 각 세그먼트마다 영상의 특성을 분석해 예측한 최적의 CRF와 Maxrate 값으로 인코딩 된 구조
   B. 한마디로 서버 측에서 영상 품질에 따라 비트레이트를 결정하고, 클라이언트 측은 네트워크에 따라 해상도를 결정하는 양방향 시스템
   C. 서로 다른 CRF/Maxrate을 적용한 세그먼트들을 해상도별 .ts 파일로 인코딩한 뒤, 이를 병합하여 mp4 파일로 재포장(remux)하고, MP4Box를 사용해 .m4s 조각과 manifest.mpd로 패키징
   D. dash.js는 MPD를 통해 생성된 .m4s 세그먼트를 일반 DASH 세그먼트처럼 읽어, 실시간 적응형 스트리밍을 정상적으로 수행

3.2. 데이터 준비

3.2.1. 영상 데이터셋 수집

- Kinetics-400 데이터셋 활용: 영상 특성 분석 및 모델 학습을 위해 Kinetics-400 데이터셋을 주요 데이터셋으로 채택했다. 이 데이터셋은 영상 내 동작 분석 연구에 광범위하게 사용되는 대규모 액션 인식 데이터셋이다. YouTube에서 추출된 약 10초 내외의 액션 인식용 클립으로 구성되며, 400여 개의 다양한 동작 클래스를 포함하고 있다. 본 과제는 영상 특성 인지를 위해 다양한 장르의 영상 데이터가 필수적이므로, Kinetics-400의 광범위한 장르 커버리지가 모델의 일반화 성능 향상에 적합하다고 판단했다. 추가적으로, 특정 장르에 대한 심층 분석을 위해 유튜브에서 직접 선별한 MP4 파일 (약 3분 길이)도 보조 데이터로 활용했다.

- 데이터셋 다운로드 및 저장: Kinetics-400 데이터셋은 일반적으로 GibHub 다운로드 스크립트를 통해 YouTube 원본 영상을 MP4 파일로 저장하여 사용할 수 있다. 그러나 Kinetics 데이터셋의 전체 크기는 약 500GB에 달하므로, 로컬 컴퓨터에 직접 다운로드하는 것이 현실적으로 어렵다고 판단했다. 이를 해결하기 위해 다운로드 스크립트 내의 저장 경로를 구글 드라이브로 수정하여 클라우드 환경에 데이터를 저장함으로써 효율적인 데이터 접근성을 확보했다.

- 데이터셋 구성: Kinetics-400은 Train, Val(Validation), Test 세 가지 폴더로 구분되어

제공된다. 특히 Train 데이터셋에는 약 141,410개의 동영상이 포함되어 있으며, 각 영상은 다양한 장르로 구성된 약 10초 내외의 클립으로 이루어져 있다. 구글 코랩 (Colab)의 컴퓨팅 자원이 부족한 관계로 그 중에서 약 1,000개를 랜덤으로 선정하여 모델 학습에 사용했다.

3.2.2. 세그먼트 단위 분할 및 전처리

본 과제는 DASH 환경을 모사하고 세그먼트 단위로 영상 특성을 분석하여 인코딩 파라미터를 최적화하는 모델을 구현하는 것을 목표로 한다. 이를 위해 수집된 영상 데이터를 세그먼트 단위로 분할하는 전처리 과정을 수행했다.

- 세그먼트 분할 기준: Kinetics-400 데이터셋의 영상은 대부분 약 10초 내외의 클립이며, 추가로 준비된 MP4 파일은 약 3분가량의 동영상이다. 이 모든 영상을 2초 단위의 세그먼트로 균일하게 분할했다. FFmpeg을 사용하여 각 비디오 파일의 영상 길이와 세그먼트 개수를 분석하고 원본 영상을 복사하여 개별 세그먼트 파일로 저장하는 방식을 사용했다.
- 말단 세그먼트 처리: 영상 특성 분석 및 모델 학습 단계에서 2초 미만의 마지막 세그먼트를 분석 대상에서 제외했다. 이는 짧은 세그먼트에서 불안정한 특성 분석 결과가 나올 가능성을 배제하고, 모델의 학습 효율성을 높이기 위함이다. 그러나 머신러닝 모델을 실제 스트리밍 시스템에 적용하고 인코딩 파라미터를 최적화할 때는 마지막 세그먼트도 포함하여 영상을 분할해야 전체 영상을 스트리밍할 수 있다. 이 경우 마지막 세그먼트에 대해서 별도의 영상 특성 분석을 수행하는 대신, 직전 세그먼트에서 결정된 인코딩 파라미터를 이어받아 적용하여 일관된 스트리밍 품질을 유지하도록 설계했다.

```
[husky.mp4] 영상 길이: 130.23초
[husky.mp4] 생성할 세그먼트 수: 66개 (마지막 세그먼트: 0.23초)
[husky.mp4] ✓ segment_63.mp4 생성완료 (126.0s ~ 128.0s, 2.0초)
[husky.mp4] ✓ segment_64.mp4 생성완료 (128.0s ~ 130.0s, 2.0초)
[husky.mp4] ✓ segment_65.mp4 생성완료 (130.0s ~ 130.2s, 0.2초)
```

3.3. 데이터 분석 (영상 특성 추출)

영상 데이터의 효율적인 분석을 위해 시간적 복잡도, 공간적 복잡도, 그리고 영상의 메타

데이터를 포함하는 다양한 영상 특성(Features)을 추출한다. 이러한 특성들은 영상의 내용과 동적인 변화를 정량적으로 파악하는 데 중요한 역할을 한다.

3.3.1. 모션 벡터 (Motion Vector) 분석

3.3.1.1. 모션 벡터 정의 및 추출 방법

모션 벡터는 비디오 압축 과정에서 인접한 프레임 간의 움직임을 나타내는 2차원 벡터다. 이는 한 프레임의 특정 영역(블록)이 다음 프레임에서 어느 위치로 이동했는지를 표현한 값이다. 모션 벡터의 크기가 크고 그 분포가 다양할수록 영상은 동적인 특성을 가지며, 반대로 모션 벡터의 크기가 작고 균일하게 분포할수록 정적인 특성을 가진다.

그림 10. 모션 벡터 비교

예를 들어, 뉴스 영상과 같이 객체의 움직임이 적고 카메라가 고정된 영상에서는 모션 벡터가 작고 균일하게 분포한다. 반면, 액션 영상과 같이 객체의 움직임이 크고 카메라 이동이 빠른 영상에서는 모션 벡터의 크기가 크고 다양하게 분포하는 경향을 보인다.

본 과제에서는 모션 벡터를 분석하기 위해 motion-vector-extractor 라이브러리를 사용했다. 이 라이브러리는 H.264 및 MPEG-4 Part 2로 인코딩 된 비디오에서 프레임, 모션 벡터, 프레임 유형 및 타임스탬프를 추출하는데 특화되어 있다. motion-vector-extractor를 사용하여 각 프레임을 cap.read() 메서드로 읽어오면 다음 정보를 반환한다.

Index	Name	Type	Description
0	**Success**	Bool	프레임 및 모션 벡터를 성공적으로 검색하면 True 반환, 그렇지 않거나 스트림 끝에 도달하는 경우 False 반환
1	**Frame**	Numpy array	영상 프레임을 포함하는 dtype uint8 타입의 (h,w,3) 배열. W와 h는 각각 프레임의 너비와 높이. 프레임을 디코딩 할 수 없는 경우 빈 numpy

			ndarray를 반환
2	**Motion vectors**	Numpy array	프레임의 N개 모션 벡터를 포함하는 dtype int32 타입의 (N, 10) 배열. 프레임에 모션 벡터가 없는 경우 빈 numpy ndarray를 반환 각 벡터의 column은 다음과 같은 의미 0: 현제 프레임에서 참조 프레임의 오프셋 1: 벡터 매크로 블록의 너비 2: 벡터 매크로 블록의 높이 3: 모션 벡터가 참조 프레임에서 가리키는 x위치 4: 모션 벡터가 참조 프레임에서 가리키는 y위치 5: 현재 프레임에서 벡터 원점의 x위치 5: 현재 프레임에서 벡터 원점의 y위치 7: x방향의 매크로블록 변위를 곱하여 정수 변환 8: y방향의 매크로블록 변위를 곱하여 정수 변환 9: 스케일링 factor를 제공하여 실제 픽셀 변위를 계산하는 데 사용
3	**Frame_type**	String	프레임 유형을 나타내는 Unicode 문자열로 "I(Intra-coded)", "P(Predicted)", "B(Bi-directionally predicted)"프레임을 표현. 알 수 없는 프레임 유형은 "?"을 반환
4	**Timestamp**	Double	본 과제에서는 이 값을 직접적으로 사용하지 않음

3.3.1.2. 모션 벡터 전처리 (노이즈 제거)

모션 벡터를 활용한 영상 특성 분석 초기 단계에서는 영상이 동적일수록 모션 벡터의 개수가 많을 것이라는 가설을 세웠다. 그러나 실제 데이터를 분석해본 결과 예상과는 반대로 고모션 영상보다 저모션 영상에서 모션 벡터의 개수가 훨씬 더 많이 추출되는 현상이 관찰되었다.

해상도가 동일한 고모션 영상(Sports)과 저모션 영상(News)에서 약 10초 단위로 추출된 모션 벡터 개수를 비교한 결과, 다음 표와 같이 저모션 영상에서 평균 350만 개의 모션 벡터가 추출된 반면, 고모션 영상에서는 평균 120만 개의 모션 벡터가 추출되어 약 3.1배 가량의 차이를 보였다.

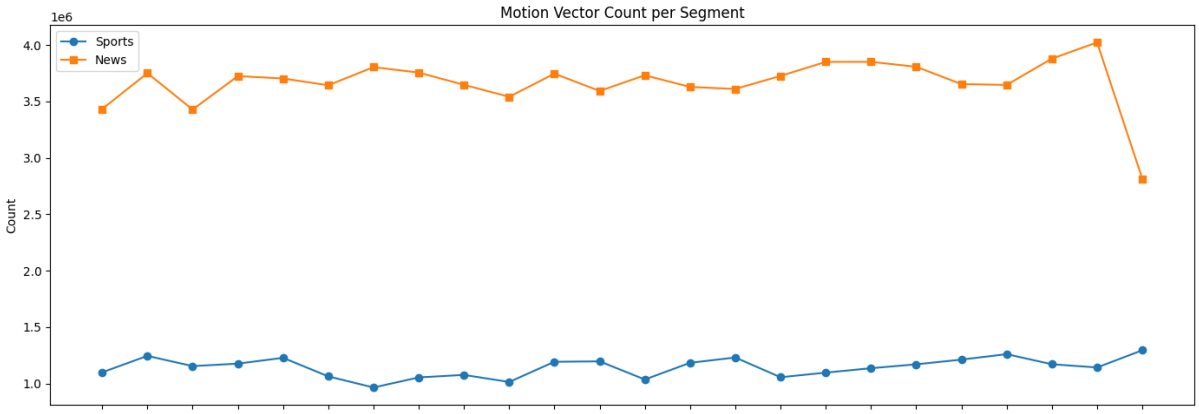


그림 11. 고모션/저모션 영상의 모션 벡터 개수

이러한 현상의 주요 원인은 News와 같이 정적인 배경 내에서 발생하는 객체의 자잘한 움직임이나 노이즈까지 모두 모션 벡터로 분류되었기 때문이다. 사람의 눈으로는 움직임이 없다고 판단하는 배경 영역에서도 매우 작은 크기의 모션 벡터들이 과도하게 많이 측정되어 전체 모션 벡터 개수를 불필요하게 증가시키는 문제가 발생했다.

초기 모델에는 모션 벡터의 개수, 모션 벡터 크기의 통계량(평균, 최대, 표준편차)을 측정하여 영상 특성으로 활용했다. 그러나 지나치게 작은 모션 벡터들이 이상치로 작용하여 모션 벡터 크기의 평균이 0에 수렴하는 등 통계적 왜곡이 발생했다. 결국 실질적인 움직임을 나타내는 중요한 모션 벡터들이 대다수의 의미 없는 작은 벡터들에 의해 제대로 표현되지 못하여 영상의 동적 특성을 정확히 반영하지 못하게 되었다.

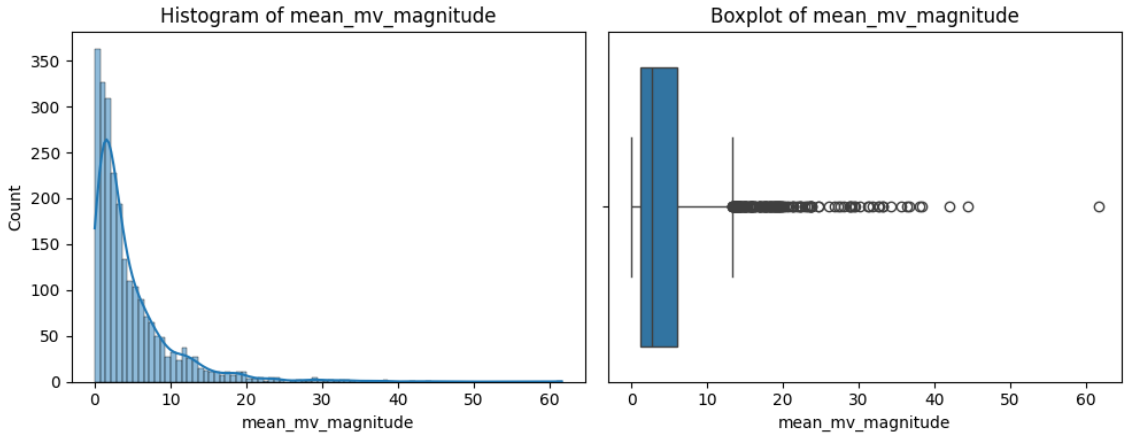


그림 12. 모션 벡터 크기의 평균이 이상치로 인해 0에 수렴하는 모습

이러한 문제를 해결하기 위해, 모션 벡터의 크기가 미리 정의된 임계값 이하인 벡터들을 노이즈로 간주하고 모션 벡터 통계 분석에서 제외하는 전처리 방법을 도입했다. 이는 미세한 흔들림이나 배경 노이즈로 인한 비실질적인 움직임 정보를 필터링하여, 실제 객체의 유의미한 움직임만을 분석에 활용하기 위함이다.

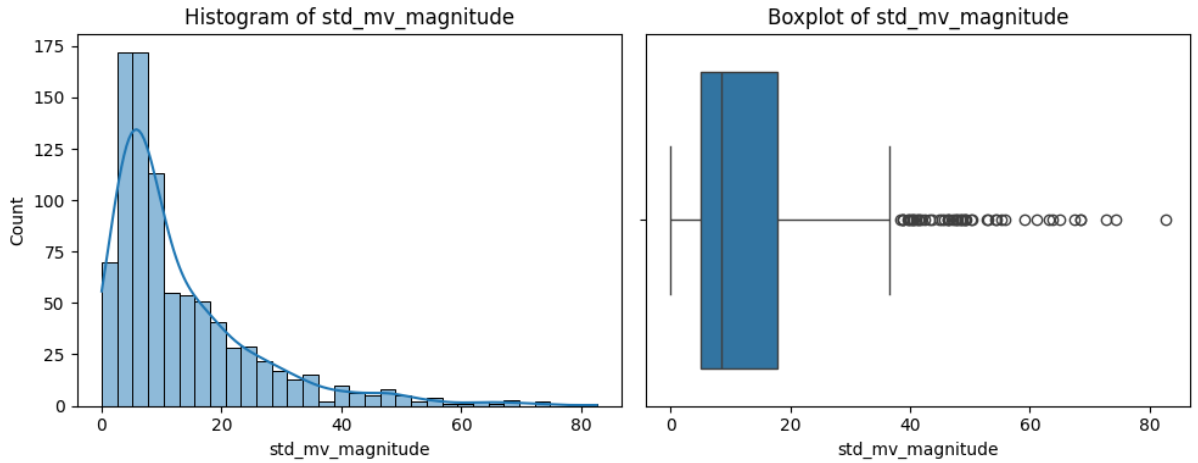


그림 13. 노이즈 제거 후 모션 벡터 크기의 평균

노이즈를 제거한 결과 다음과 같이 모션 벡터 크기의 평균이 0에 수렴하던 문제를 효과적으로 해결하고, 비교적 넓은 범위에 걸쳐 분포하는 것을 확인할 수 있다. 그러나 여전히 데이터 내에 존재하는 다른 유형의 이상치들이 분석 결과에 영향을 미칠 수 있으므로, 추가적인 이상치 탐지 및 제거 기법의 적용이 필요한 것으로 보인다.

3.3.1.3. 세그먼트 내 프레임 단위 통계 지표 분석

초기 모델에서는 세그먼트 내 프레임의 순서에 관계없이, 모든 프레임에서 추출한 모션 벡터의 크기를 한 번에 수집하여 평균, 표준편차, 최댓값을 계산하였다.

그러나 이 방식은 세그먼트 내에서 프레임 간 모션 벡터 크기의 변화 양상을 반영하지 못한다는 한계가 있었다. 이를 보완하기 위해, 본 과제에서는 다음과 같은 프레임 단위 통계 지표를 추가적으로 정의하여 분석하였다.

- 프레임 간 모션 벡터 크기의 분산 (Motion Variance Across Frames): 각 프레임별 모션 벡터 크기의 평균을 구한 후, 이를 세그먼트 단위로 모아 분산을 계산한다. 이 지표는 세그먼트 내 움직임의 시간적 변화 정도를 측정한다.
- 프레임 간 모션 벡터 크기의 일관성 (Temporal Smoothness): 연속하는 프레임 간 평균 모션 벡터 크기의 차이를 계산하여, 변화 폭이 낮을수록 일관성이 높다고 본다. 일관성이 높을수록 움직임이 안정적이어서 영상 압축이 용이하고, 반대로 변화가 클수록 급격한 움직임으로 인해 압축이 어렵다.
- 프레임별 의미 있는 모션 벡터 비율의 평균: 사전에 설정한 임계 값을 초과하는

모션 벡터를 '의미 있는' 움직임으로 간주하고 각 프레임에서 (의미 있는 모션 벡터 개수 / 전체 모션 벡터 개수)를 계산한 후 세그먼트 단위로 평균을 구한다. 이는 전체 모션 벡터 중 실제 움직임에 해당하는 비율을 나타내며, 미세한 움직임(노이즈)을 제외한 실질적인 모션 강도를 평가하는 지표로 활용된다.

이 지표들을 통해, 세그먼트 내부의 모션 패턴을 보다 세밀하게 파악하고 영상 특성 기반의 적응형 스트리밍을 위한 분석의 정밀도를 높이고자 했다.

3.3.1.4. 동적/정적 영상 간 지표 비교 분석

영상 콘텐츠의 동적인 특성을 정량적으로 분석하기 위해, 고모션 영상인 'Husky Agility'와 저모션 영상인 'News'를 선정하여 주요 모션 벡터 지표들을 비교 분석했다. 두 영상은 동일한 해상도(1080x720)와 프레임 레이트(30fps)로 구성되었으며, 10초 단위로 분할된 세그먼트에서 각 지표를 추출하여 그 특성을 비교했다. 분석 결과는 아래 표와 같으며, 이는 각 영상의 동적 특성을 명확히 구분하는 데 중요한 근거를 제공한다.

Metric	**설명**	**Husky Agility (동적)**	**News (정적)**
Mean MV Magnitude	모션 벡터 크기의 평균	18.14 ± 11.47	3.49 ± 1.05
Std MV Magnitude	모션 벡터 크기의 표준편차	16.55 ± 10.19	5.12 ± 2.87
Max MV Magnitude	모션 벡터 크기의 최댓값	288.16 ± 143.14	104.21 ± 29.25
Median MV Magnitude	모션 벡터 크기의 중앙값	13.49 ± 9.08	2.18 ± 0.38
95th Percentile MV	모션 벡터 크기의 95% 백분위 수	48.83 ± 30.76	9.86 ± 6.87
Direction Entropy	방향 엔트로피	3.22 ± 0.47	3.60 ± 0.31
Directional Consistency	방향 일관성	0.43 ± 0.07	0.42 ± 0.04
Motion Energy	모션 에너지(모션 벡터 크기의 제곱)	838.30 ± 1037.46	47.76 ± 62.14

Partition Diversity	파티션 다양성	0.71 ± 0.25	0.31 ± 0.19
Temporal Smoothness	프레임 간 모션 크기의 시간적 일관성	13.09 ± 7.27	3.06 ± 3.56
Motion Variance	프레임 간 모션 크기의 분산	208.72 ± 210.28	12.88 ± 33.34
Significant Motion Ratio	프레임별 의미 있는 모션 벡터의 비율	0.66 ± 0.23	0.10 ± 0.09
MB16x16 Ratio	16*16 파티션 비율	0.85 ± 0.06	0.95 ± 0.04
MB16x8 Ratio	16*8 파티션 비율	0.11 ± 0.04	0.03 ± 0.03
MB8x8 Ratio	8*8 파티션 비율	0.04 ± 0.02	0.02 ± 0.02

그림 14. 동적/정적 영상의 모션 벡터 비교

주요 지표들을 Heatmap 분석을 통해 상관관계를 분석한 결과는 다음과 같다.

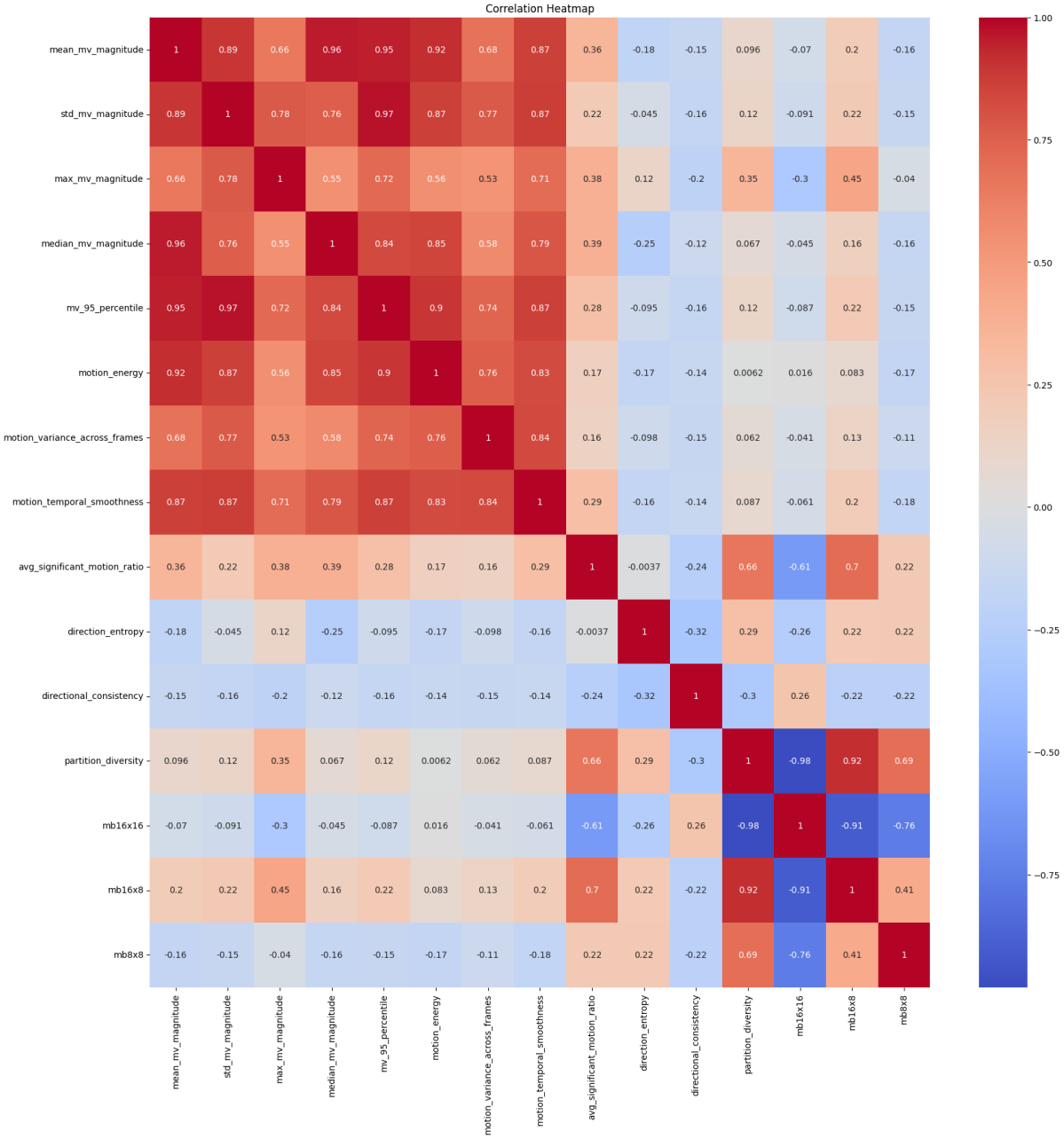


그림 15. 모션 벡터 비교 지표 Heatmap

- 모션 벡터 크기 관련 지표 (Mean, Std, Max, Median MV Magnitude, 95소 percentile MV, Motion Energy): 'Husky Agility' 영상은 평균, 표준 편차, 최댓값, 중앙값, 95% 백분위수 등 모든 모션 벡터 크기 관련 지표에서 'News' 영상보다 월등히 높은 값을 보인다. 이는 동적인 영상이 빠르고 격렬한 움직임, 그리고 빈번한 카메라 전환을 포함하고 있어, 각 프레임 간 픽셀의 이동 거리가 크고 변화량이 크다는 것을 의미한다.

- Direction Entropy 및 Directional Consistency: 방향 엔트로피는 모션 벡터 방향의

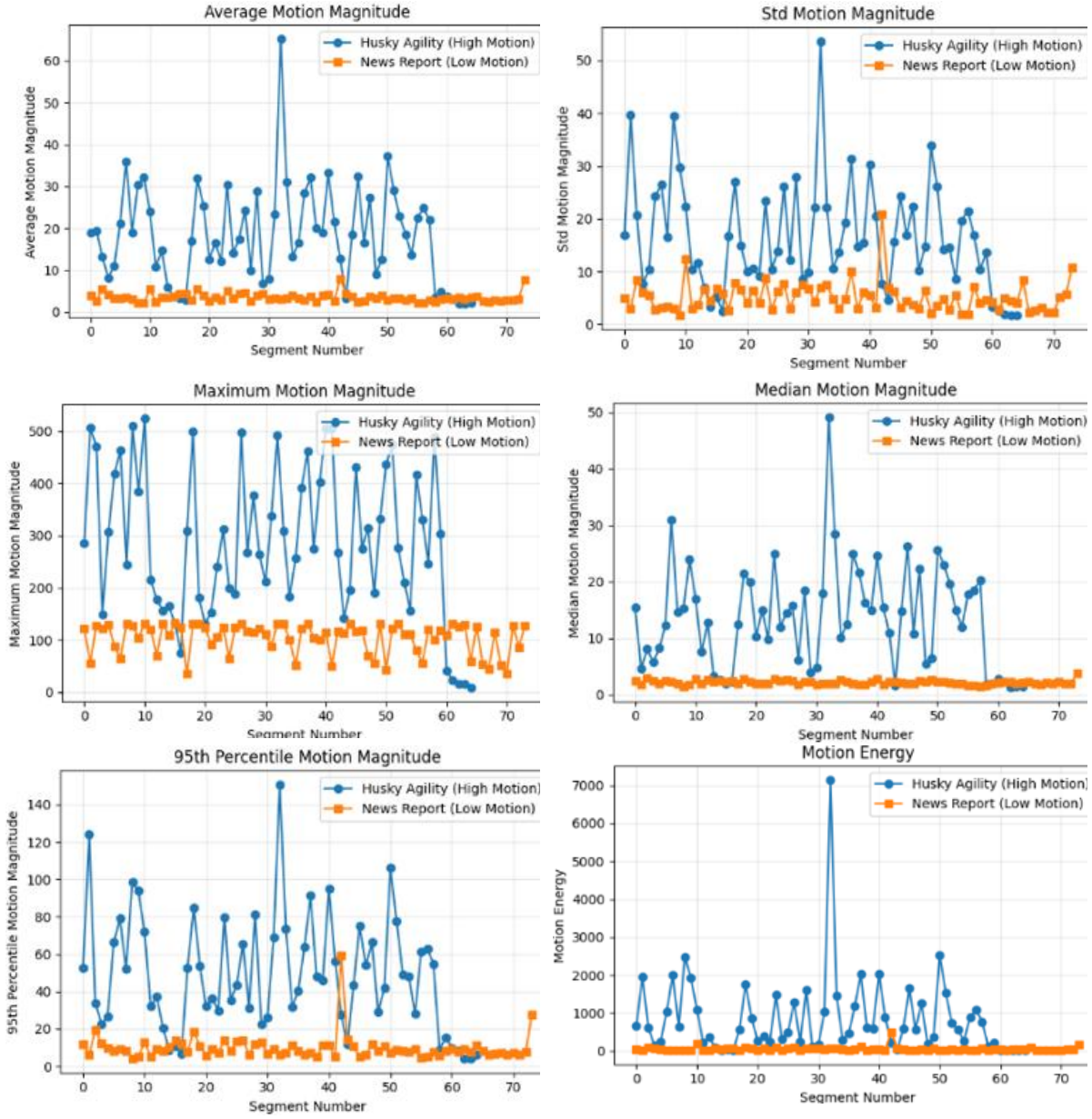
다양성을 측정하는 지표이다. 이 때 News 영상의 엔트로피가 Husky Agility 영상보다 약간 높게 나타난 것을 볼 수 있는데, 이는 정적 영상임에도 불구하고 앵커의 미세한 표정 변화, 배경의 작은 로고 움직임, 혹은 카메라의 미세한 흔들림 등 전역적으로 일관되지 않은 작은 움직임들이 다양한 방향으로 분산되어 있기 때문이다. 반면 Husky Agility는 특정 개체나 카메라의 움직임이 주요 움직임이기 때문에 모션 벡터들이 일관된 방향성을 띠는 경향이 강하여 방향 일관성이 상대적으로 높게 나타났다.

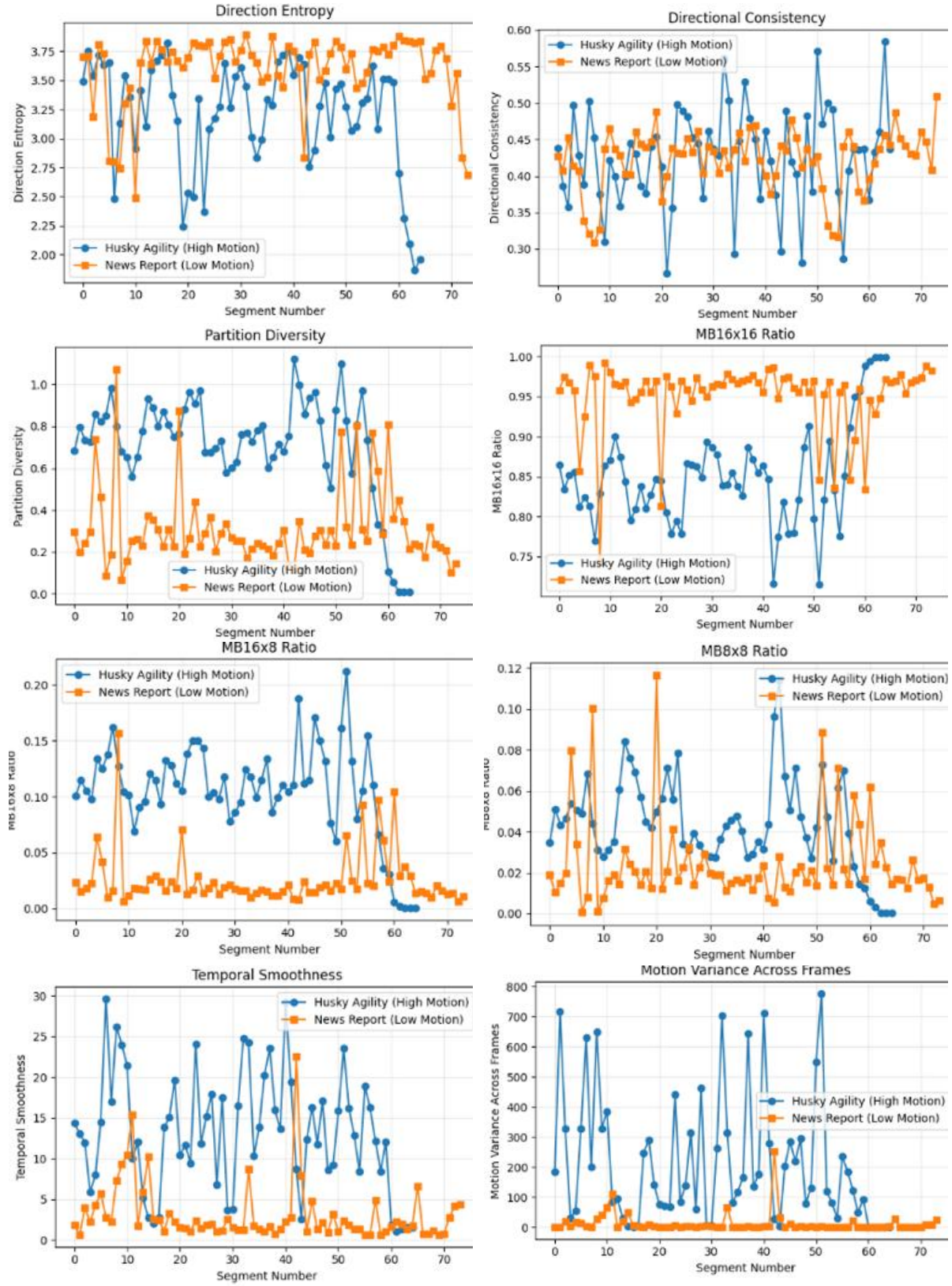
- Partition Diversity 및 MB Block Ratio (MB16x16, MB16x8, MB8x8): 파티션 다양성은 비디오 인코딩 시 매크로 블록의 분할 방식이 얼마나 다양한지를 나타내는 지표다. 값이 클수록 영상 내 복잡한 텍스처나 세밀한 움직임이 많아 인코더가 픽셀을 효율적으로 예측하기 위해 다양한 크기의 블록으로 분할했다는 것을 의미한다. Husky Agility는 News 영상에 비해 파티션 다양성이 높게 나타났는데, 이는 동시에 MB16x8, MB8x8과 같은 비교적 작은 블록의 비율이 높다는 결과도 나타낸다. 작은 블록 분할은 복잡한 움직임 영역을 더 정밀하게 인코딩하기 위해 필요하며, 결과적으로 압축 난이도를 증가시킨다. 반면 News 영상은 움직임이 적고 배경이 정적이므로 인코더가 넓은 영역을 단일 블록으로 처리하기 위해 MB16x16과 같은 큰 블록의 비율이 높게 나타난다. Heatmap에서도 파티션 다양성은 큰 블록의 비율과 음의 상관관계를, 작은 블록 블록과는 양의 상관관계를 가진다는 것을 확인할 수 있다.

- Temporal Smoothness, Motion Variance, Significant Motion Ratio: Temporal Smoothness는 프레임 간 모션 벡터 크기의 시간적 일관성을 나타내는 지표로, 값이 클수록 프레임 간 모션 변화가 급격하다는 것을 의미한다. Motion Variance는 프레임 간 모션 크기의 분산을 측정한다. Husky Agility는 격렬한 움직임으로 인해 프레임 간 모션 변화가 매우 크고 불규칙하여 두 지표 모두에서 높은 값을 보인다. Significant Motion Ratio는 프레임 내에서 미리 정의된 임계값을 초과하는 유의미한 모션 벡터 블록의 비율을 나타낸다. News 영상의 경우, 앵커의 미세한 입 모양 변화나 배경의 아주 작은 움직임들이 임계값 이하로 분류되어 노이즈로 처리되어서 이 지표 값이 약 0.1 정도로 매우 낮게 측정되었다.

Husky Agility와 News 영상을 세그먼트 단위로 분할하여 각 지표의 시간적 변화를 그래프로 시각화한 결과, Husky Agility 영상의 후반부에서 움직임이 점차 줄어드는 구간에 Average Motion Magnitude나 Direction Entropy, Significant Motion Ratio가 낮아지고

MB16x16 Ratio가 높이지는 등 대부분의 지표들이 영상의 동적 특성 변화를 반영하는 일관적인 패턴을 보인다. 이러한 결과는 모션 벡터 기반 지표들이 영상 콘텐츠의 동적인 복잡성을 실시간으로 감지하고 분석하는데 효과적임을 나타낸다.




Direction Entropy
Direction Entropy
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)
Directional Consistency
Directional Consistency
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)
Partition Diversity
Partition Diversity
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)
MB16x16 Ratio
MB16x16 Ratio
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)
MB16x8 Ratio
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)
MB8x8 Ratio
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)
Temporal Smoothness
Temporal Smoothness
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)
Motion Variance Across Frames
Motion Variance Across Frames
Segment Number
Husky Agility (High Motion)
News Report (Low Motion)

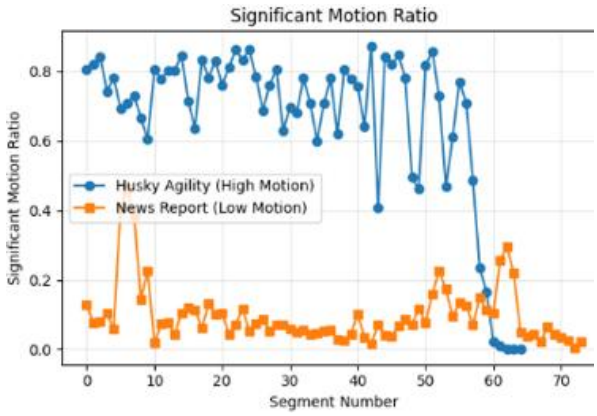


그림 16. 모션 벡터 지표 분석 그래프

3.3.2. 매크로블록(Macroblock) 분석

3.3.2.1. 매크로 블록 정의 및 추출 방법

매크로블록(Macroblock, MB)은 영상 데이터를 처리하는 핵심 단위로 사용된다. H.264/AVC 표준에서는 매크로블록을 기반으로 하는 다양한 예측 기법을 사용해 높은 압축 효율을 달성한다. 매크로블록은 일반적으로 16*16 픽셀 크기이며, 영상의 복잡도에 따라 8*8, 4*4 크기와 같이 더 작은 서브 매크로블록 파티션으로 분할될 수 있다. 비디오 데이터는 원본 영상에서 세그먼트, GOP(Group of Pictures), 프레임, 매크로블록, 서브 매크로블록 파티션 순서로 계층적으로 처리된다.

매크로블록은 FFmpeg의 -debug mb_type 옵션을 통해 로그 파일로 출력할 수 있다. 로그 파일에는 각 프레임마다 매크로 블록의 유형이 기호로 표시되어 있으며, 각 기호는 FFmpeg의 공식 문서를 통해 해석 가능하다. 본 과제는 매크로블록이 영상의 움직임에 미치는 영향을 분석하기 위해 크게 세 가지 타입(Intra 블록, Inter 블록, Skip 블록)으로 분류했다. 영상의 움직임이 많을수록 Intra/Inter 블록의 비율이 증가하고 Skip 블록의 비율은 감소하는 경향을 보이는 반면, 움직임이 적은 영상일수록 Skip 블록의 비율이 증가하고 Intra/Inter 블록의 비율은 감소하는 경향을 나타낸다.

그림 17. 매크로블록 분포 비교

3.3.2.2. 매크로블록 타입별 분포 분석

- Intra 블록 (화면 내 예측 블록): Intra 블록은 현재 프레임 내의 인접한 픽셀 정보만을 사용하여 부호화 되는 블록이다. 이는 외부 프레임 참조 없이 독립적으로 인코딩 되며, 주로 장면 전환이나 복잡한 텍스처의 등장, 예측이 어려운 불규칙한 움직임이 있는 구간에 많이 분포한다.

  뉴스 영상과 같이 비교적 정적인 영상은 Intra 블록 비율이 낮고, 뉴스 장면이 전환될 때는 순간적으로 Intra 블록의 비율이 급격히 증가하는 경향을 보인다. Husky Agility와 같은 동적인 영상은 장면 전환이 잦고 움직임이 많아서 대체적으로 Intra 블록의 비율이 높게 나타난다.

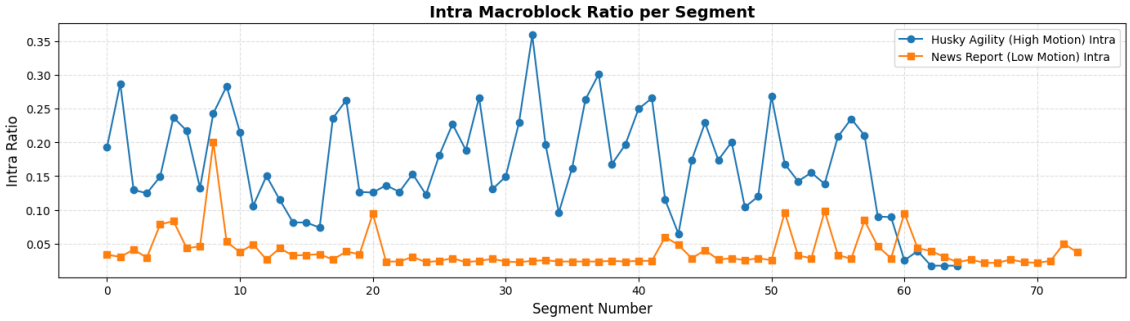


그림 18. Intra 블록 비교

- Inter 블록 (화면 간 예측 블록): Inter 블록은 현재 프레임을 인코딩할 때 이전 또는 이후 프레임의 정보를 참조하여 예측하고 부호화 하는 블록이다. 이는 주로 움직임이 많거나 예측이 활발하게 일어나는 구간에서 높은 비율로 분포한다.

  Husky Agility 영상의 경우, 활발한 움직임이 나타나는 구간에서 Inter 블록의 비율이 높게 나타나지만, 영상의 마지막 구간으로 갈수록 움직임이 감소함에 따라 예측 효율이 떨어지거나 움직임 자체가 줄어들어 Inter 블록의 비율 또한 낮아지는 경향을 보인다.

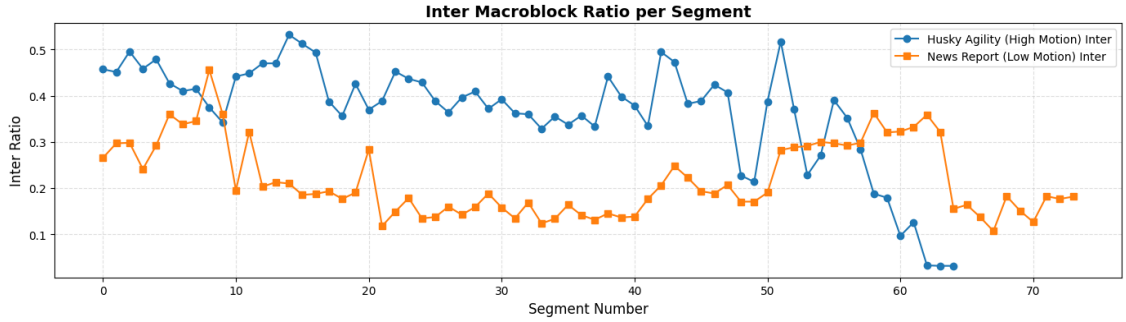


그림 19. Inter 블록 비교

- Skip 블록: Skip 블록은 이전 프레임의 해당 매크로블록과 현재 프레임의 매크로블록 간에 변화가 거의 없을 때, 움직임 벡터나 잔여 정보를 전송하지 않고 단순히 이전 프레임의 정보를 복사하여 부호화 하는 블록이다. Skip 블록은 전송 데이터의 양을 최소화하여 인코딩 효율을 극대화한다. Skip 블록은 주로 움직임이 거의 없는 정적인 구간이나 배경 영역에 많이 분포한다.

  News 영상과 같이 정적인 배경이 많고 객체의 움직임이 제한적인 경우, Skip 블록의 비율이 높게 나타난다. Husky Agility 영상에서도 움직임이 완전히 멈추는 구간에서는 Skip 블록의 비율이 증가하는 경향을 보인다.

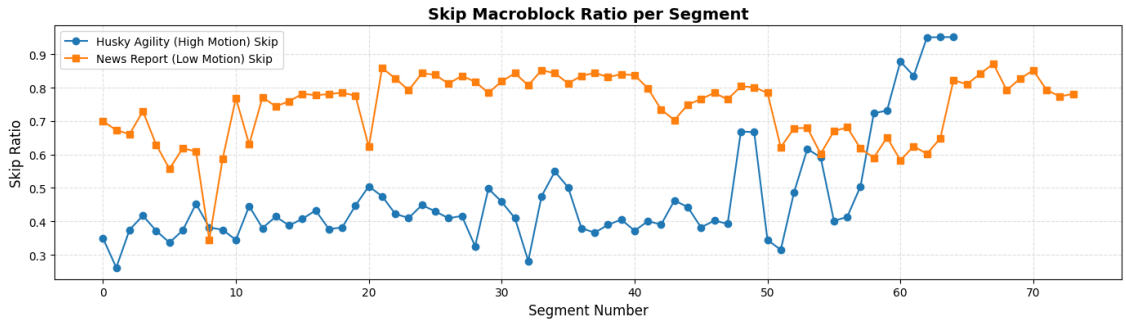


그림 20. Skip 블록 비교

3.3.3. 영상의 공간적 복잡도

3.3.3.1. Edge Detection - Canny Edge Detection

영상의 공간적 복잡도를 추출하는 방법으로 초기 모델은 Edge Density와 Texture Complexity를 사용했다. Edge Density는 Canny Edge Detection 알고리즘으로 픽셀 간 명암 변화가 큰 부분을 검출한 다음 전체 픽셀 수로 나누어 밀도를 구한다. Texture Complexity는 라플라시안 2차 미분 필터를 사용해 필터의 급격한 강도 변화를 측정하여

세부적인 패턴과 질감을 측정하도록 한다. 훈련 데이터를 이용해 Edge Density와 Texture Complexity의 통계를 내보면 다음과 같다.

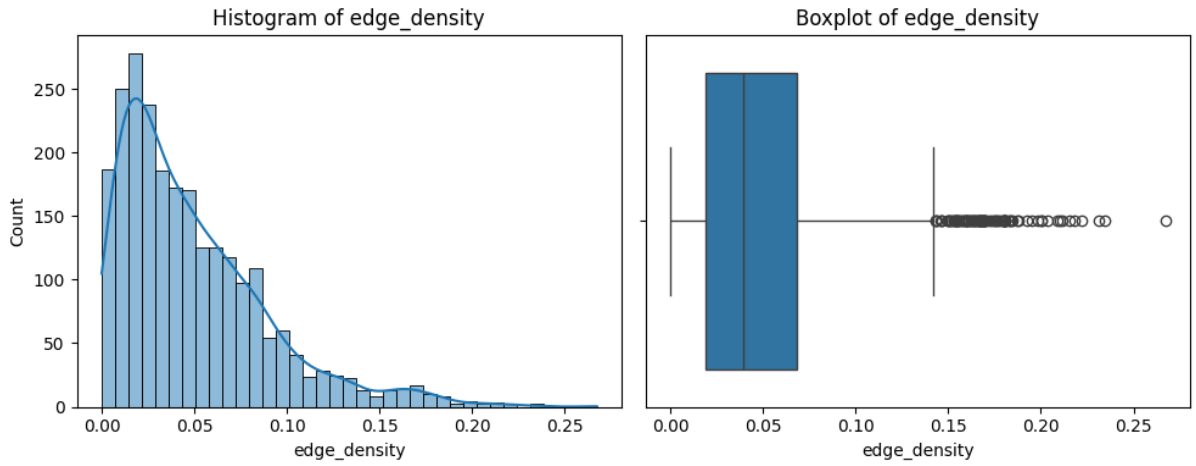


그림 21. Edge Density

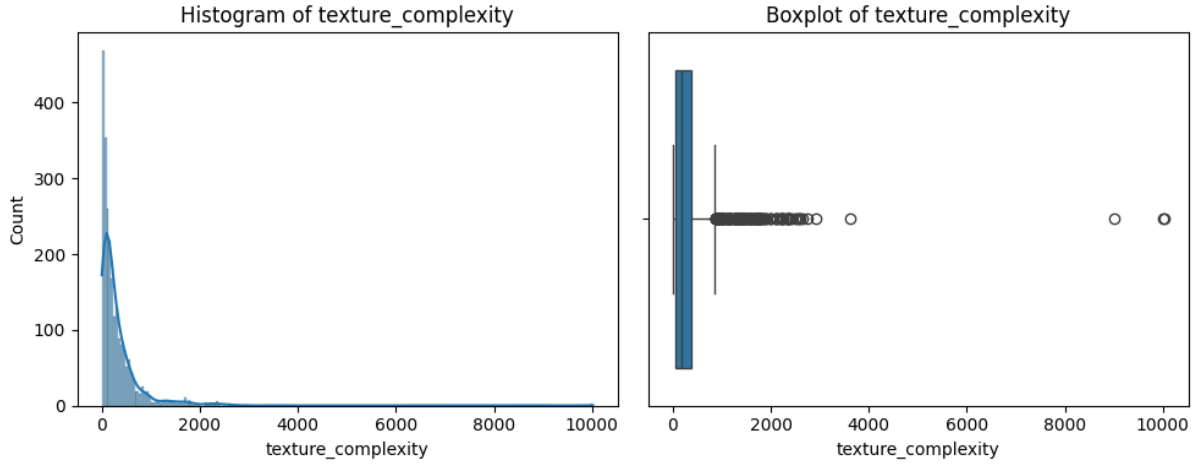


그림 22. Texture Complexity

히스토그램 분포를 확인해보면 두 지표 모두 거의 0에 수렴하며, 이상치가 많아서 제대로 된 데이터를 얻기 힘들다. 분포가 0에 가까운 이유를 알아보기 위해 Canny Edge Detection을 사용해 이미지의 경계선을 처리한 이미지를 시각화 했다.

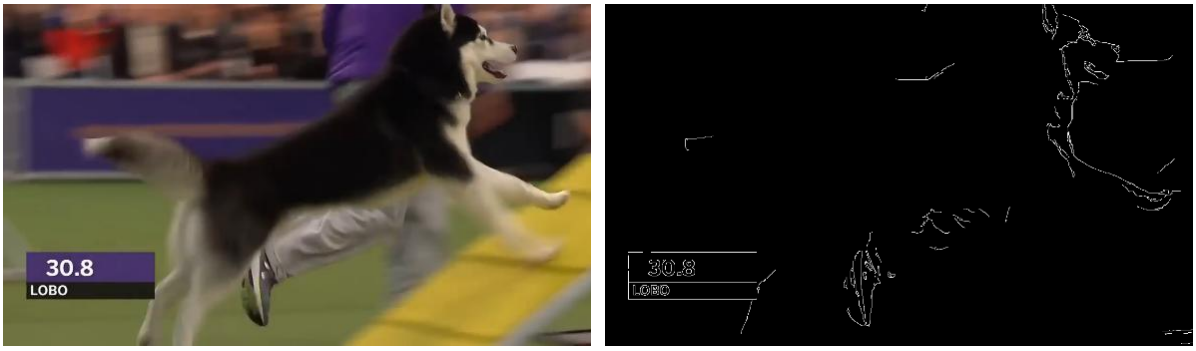


그림 23. Canny Edge Detection

원하던 결과는 실제 경계선(강아지, 판)을 모두 표시하는 거지만 실제로 확인해보니 개의 얼굴과 자막을 제외하면 경계선이 거의 검출되지 않은 것을 볼 수 있다. 이는 카메라 이동이나 객체 이동으로 인한 모션 블러 때문으로 추측했다. 경계선 부분이 움직임으로 인해 블러 처리되면서 실제 경계선으로 인식되지 않은 것이다.

3.3.3.2. 경계선 검출 분석

이를 해결하기 위해 Sobel, Laplacian, Canny, Local Binary Pattern(LBP)를 사용해 경계선 검출을 시행해본다.

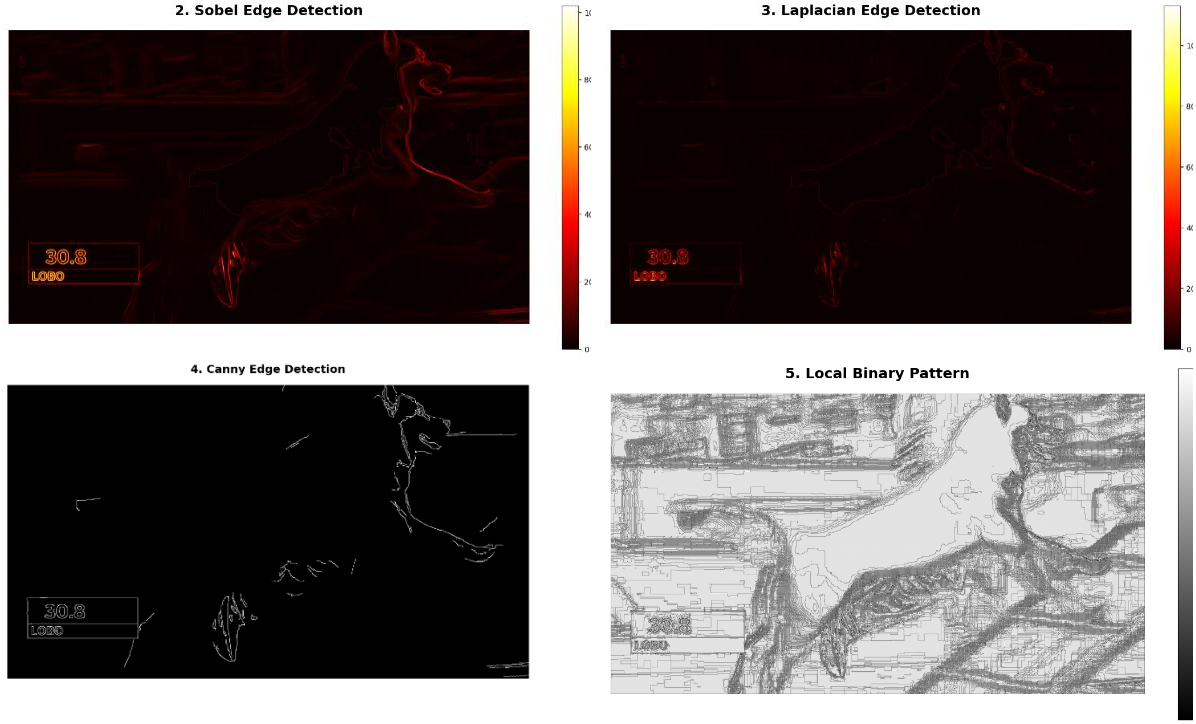


그림 24. Edge Detection Metric

결과를 보면 LBP가 다른 방법에 비해 모션 블러에 영향을 덜 받으면서 패턴을 잘 표현하고 있다. LBP는 주변 픽셀과의 밝기 비교를 통해 로컬 패턴을 이진 코드로 표현하는

텍스처 특징 추출 기법으로, Gradient 크기에 의존하지 않고 이웃 간 픽셀을 비교하기 때문에 패턴의 상대적 관계가 잘 유지된다.

반면 Canny Edge Detection은 픽셀 간의 gradient를 측정하고 임계점을 기준으로 경계를 파악한다. 만약 모션으로 인해 경계선이 블러 처리되면 pixel 값이 비교적 부드럽게 변하고, 기울기 값이 임계점을 넘기 힘들어져 경계선을 검출할 수 없다.

3.3.3.3. Edge Detection – Local Binary Pattern

LBP 이미지는 0~9 사이의 값으로 이루어져 있는데 9에 가까울수록 주변 픽셀과 거의 차이가 나지 않는 평탄한 이미지(배경)이고, 0에 가까울수록 경계선이라고 판단했다. 이를 이용하여 LBP 값이 3 이하인 픽셀 수를 전체 픽셀 수로 나누어 프레임의 Edge Density를 계산한다.

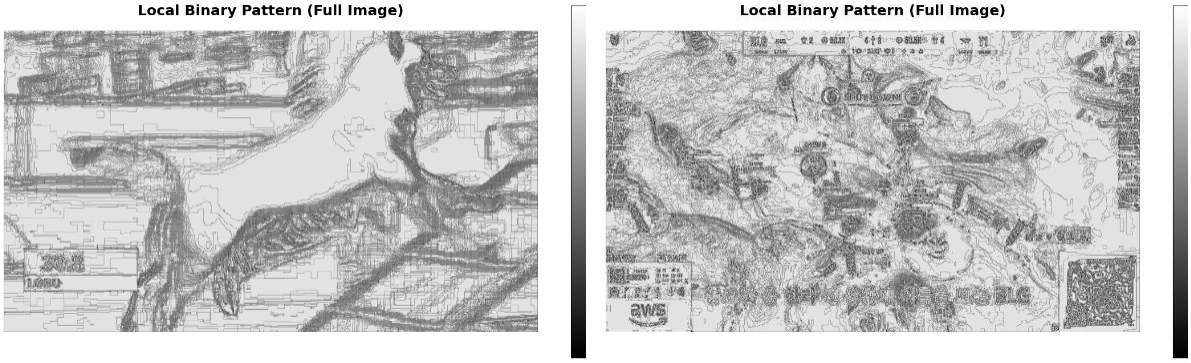


그림 25. LBP 비교

3.3.3.4. Pixel Entropy

추가적으로 공간적 복잡도를 분석하기 위해 텍스쳐 복잡도를 측정한다. 기존에 활용한 라플라시안 필터와 달리 픽셀을 255개 구간으로 펼쳐 Pixel Value Histogram을 만들어 엔트로피를 구해 Pixel Entropy라는 지표를 정의한다. 영상의 공간적 복잡도가 높을수록 픽셀 값이 다양하게 분포하므로 Pixel Entropy가 증가하는 경향을 보인다.

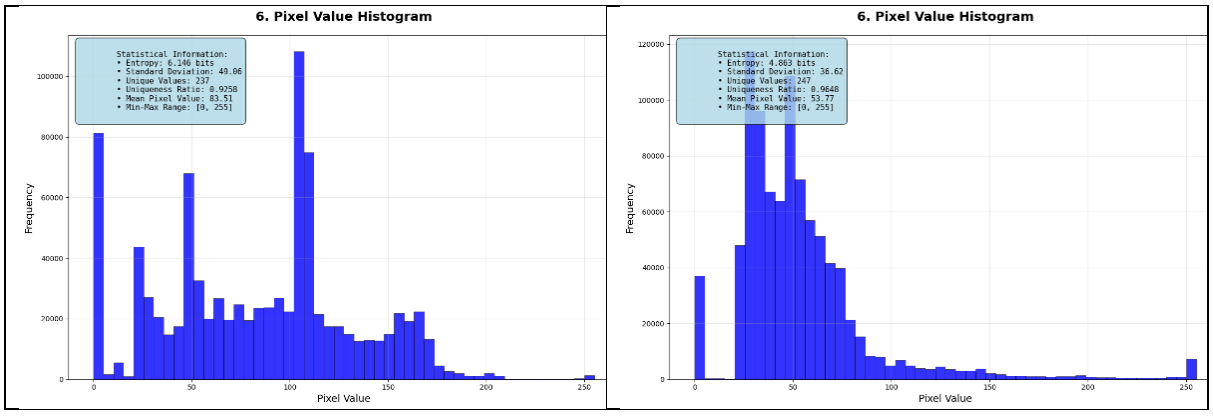


그림 26. Histogram 비교

3.3.4. 영상 메타 데이터

영상 메타 데이터로는 영상의 해상도(width, height)와 프레임 레이트, 영상 길이를 포함한다.

- **2초 미만의 영상 처리**

세그먼트를 2초 단위로 분할 시 마지막 세그먼트는 2초 미만의 아주 작은 세그먼트로 분할된다. 이러한 마지막 세그먼트들은 세그먼트 길이를 box plot으로 시각화하면 이상치로 분류되어 정상적인 통계를 얻기 어렵다.

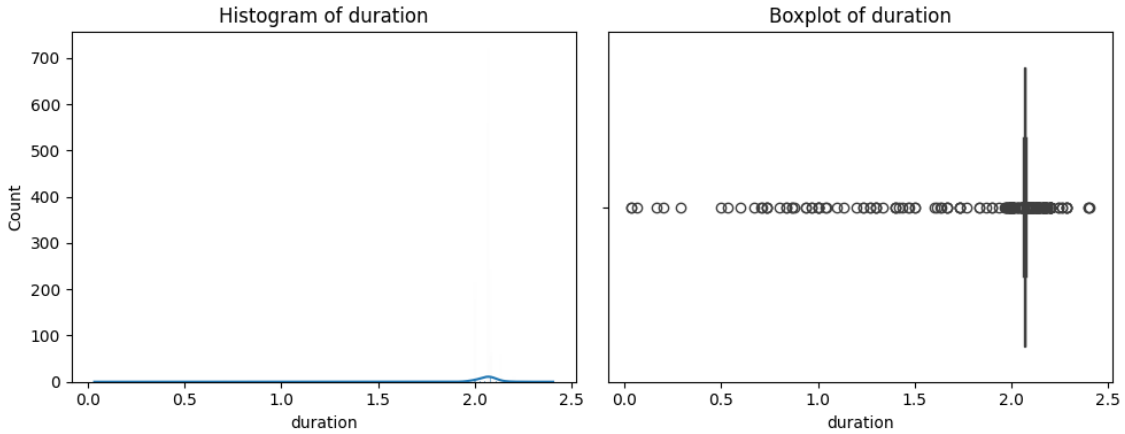


그림 27. 영상 길이 변경 전

따라서 훈련 데이터에서는 2초 단위로 분할하되, 2초 미만의 마지막 세그먼트는 버리고 모델 학습 데이터를 만든다. 실제 서비스에서는 마지막 세그먼트의 인코딩 파라미터를 예측하지 않고, 바로 직전의 세그먼트에서 가져와 그대로 적용하는 방식을 사용한다.

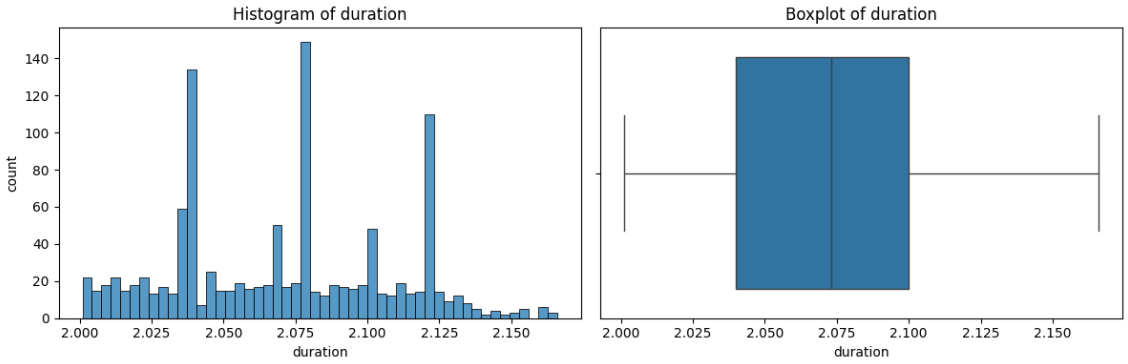


그림 28. 영상 길이 변경 후

마지막 세그먼트를 제외하고 시각화하면 세그먼트의 길이가 약 2.000 ~ 2.150 정도로 정

확하게 2초 단위로 분할되지 않았다. 이는 세그먼트 분할 시 GOP 구조로 인하여 키프레임의 위치마다 분할하기 때문이다. 약 0.15초 정도의 오차는 AI 모델에 유의미한 영향을 미치 않는다고 판단했다.

3.4. 인코딩 파라미터 및 비트레이트 배분 방식

영상 스트리밍 서비스에서 비트레이트 배분 방식은 영상의 압축 효율과 화질, 그리고 파일 크기에 직접적인 영향을 미치므로 신중하게 선택해야 한다. H.264와 같은 비디오 인코더는 다양한 비트레이트 배분 방식을 제공하여 특정 목표(화질, 파일 크기, 대역폭)에 맞춰 인코딩을 수행할 수 있도록 한다. 주요 비트레이트 배분 방식은 다음과 같다.

- CQP (Constant QP): 각 프레임에 고정된 양자화 수준(Quantization Parameter)을 적용하여 인코딩하는 방식
- CBR (Constant Bitrate, 고정 비트레이트): 전체 영상을 지정된 비트레이트로 꾸준히 유지하며 인코딩하는 방식으로 파일 크기 예측과 대역폭 할당에 유리하지만, 복잡한 장면에서 화질 저하나 단순한 장면에서 비트를 낭비
- VBR (Variable Bitrate): 영상의 복잡도에 따라 비트레이트를 동적으로 조절하는 방식
- ABR (Average Bitrate): 인코딩 과정에서 평균 비트레이트를 목표로 설정하고, 이 평균값을 맞추면서 복잡도에 따라 비트레이트를 조절하는 방식
- CRF (Constant Rate Factor): 일정한 시각적 화질을 유지하는 것을 목표로, 장면에 따라 필요한 비트레이트를 동적으로 조절하는 방식. 사용자가 설정한 CRF 값에 따라 화질이 결정되며, 값이 낮을수록 고화질을 의미

장면에 따라 비트레이트를 조절하는 방식을 비교해 본다.

	제어 대상	파일 크기 예측	화질 일관성	적합한 용도
VBR	비트레이트 범위 (최대, 최소)	예측 어려움	범위 내에서 변동	높은 화질, 파일 크기 제한이 중요하지 않은 경우
ABR	평균 비트레이트	대략적으로 예측 가능	평균 비트레이트 주변에서 변동	대략적인 파일 크기 예측이 필요하면서 비교적 좋은 화질이 필요
CRF	화질 수준	예측 어려움	일관된 화질 유지	일정한 품질의 스트리밍 유지

그림 29. 비트레이트 배분 방식

본 과제의 목표는 영상 스트리밍 시 화질 저하를 완화하고 비트레이트를 효율적으로 배분하여 사용자 경험을 향상시키는 것이다. 이를 위해 CRF (Constant Rate Factor) 인코딩 파라미터를 선택했다. CRF는 영상의 시각적 복잡도에 따라 비트레이트를 유연하게 조절하여, 전체 영상에 걸쳐 일관된 시각적 품질을 유지할 수 있다는 장점이 있다.

그러나 CRF 방식은 네트워크 상황을 직접적으로 고려하지 않기 때문에, 순간적으로 영상의 비트레이트가 급격히 높아질 수 있다. 이러한 비트레이트 스파이크는 네트워크 과부하, 버퍼링으로 이어져 사용자 경험을 저하시킬 수 있다.

따라서 이러한 문제를 방지하지 위해 CRF와 Max rate를 결합한 Capped CRF 방식을 비트레이트 배분 방식으로 결정했다. Max rate는 인코딩되는 비트스트림의 상한선을 설정하여, 아무리 복잡한 장면이라도 지정된 비트레이트 상한선을 초과하지 않도록 제어한다.

3.5. 모델 학습용 데이터셋 구축 및 품질 평가

3.5.1. 훈련 데이터셋 생성 프로세스

모델 학습에 필요한 (입력: 영상 특성, 출력: 최적 파라미터) 쌍을 만들기 위해 다음과 같은 프로세스를 진행하여 데이터셋을 생성했다.

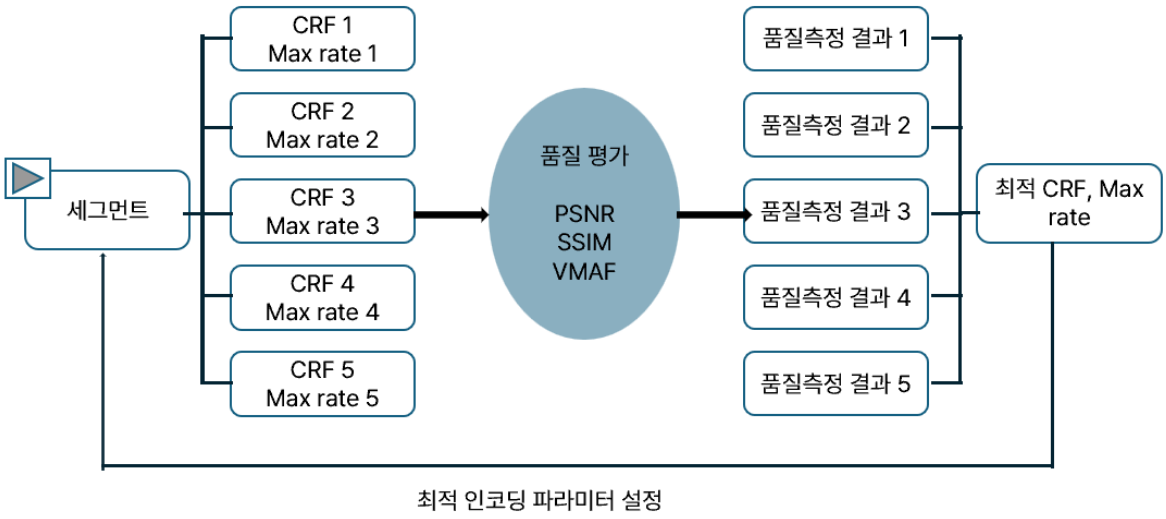


그림 30. 훈련 데이터셋 생성 프로세스

1. 영상 세그먼트 분할 (Segmentation): 원본 비디오를 2초 단위의 세그먼트로 분할
2. 파라미터 조합 기반 다중 인코딩(Multi-Pass Encoding): 각 세그먼트에 대해 미리

정의된 범위의 CRF와 Max rate 파라미터 조합으로 인코딩을 수행한다.

3. VMAF 점수 및 비트레이트 측정: 생성된 모든 버전의 인코딩 결과물에 대해 VMAF 점수와 평균 비트레이트를 측정하여 기록한다.
4. 최적 파라미터 선정: 측정된 결과 중, 최소한의 비트레이트로 목표 VMAF 점수(90점)을 달성하는 (CRF, Max rate) 조합을 해당 세그먼트의 최적 파라미터로 선정한다. 이 과정을 통해 (해당 세그먼트의 영상 특성 값, 최적 CRF, 최적 Max rate)로 구성된 하나의 훈련 데이터 샘플이 완성된다.

3.5.2. 품질 평가 (Video Quality Metric)

미리 정의된 인코딩 파라미터로 처리된 세그먼트와 원본과의 시각적 차이를 정량화 하여 최적화해야 한다. 이러한 품질 평가 방법 중, PSNR(Peak Signal-to-Noise Ratio), SSIM(Structural Similarity Index Measure), VMAF (Video Multimethod Assessment Fusion)가 가장 널리 사용된다.

3.5.2.1. PSNR (Peak Signal-to-Noist-Ratio)

PSNR은 원본 비디오 신호의 최대 가능 전력과 인코딩 과정에서 발생하는 노이즈 전력 간의 비율을 측정하는 품질 평가 지표이다. 이 지표는 두 영상의 픽셀 값 차이인 평균 제곱 오차(MSE)를 기반으로 계산되며, 계산이 비교적 간단해서 실시간 애플리케이션에 적용하기 좋다. 그러나 PSNR은 인간 시각 시스템(HVS)의 복잡한 특성을 반영하지 못하여, 실제 인간이 인지하는 품질과 불일치할 수 있다.

3.5.2.2. SSIM (Structural Similarity Index Measure)

SSIM은 원본 영상과 인코딩 영상 간의 구조적 유사성을 측정하여 품질을 평가한다. 이 지표는 휘도(Luminance), 명암(Contrast), 구조(Structural)의 세 가지 측면에서 이미지의 유사성을 계산한다. SSIM은 PSNR보다 인간 시각 시스템(HVS)을 적절하게 반영하지만 특정 왜곡에 민감하다는 단점이 있다.

3.5.2.3. VMAF (Video Multimethod Assessment Fusion):

VMAF는 Netflix에서 개발한 품질 평가 지표로, 여러 기본 품질 지표들을 머신러닝으로 융합하여 인간의 시각적 품질 평가를 가장 잘 반영한다. VMAF는 다음과 같이 동작한다.

1. Pixel Neighborhood 레벨: 특징 추출

- Spatial feature extraction (공간적 특징 추출)
  - ✓ VIF (Visual Information Fidelity, 시각 정보 충실도): 원본 영상이 가진 정보량이 인코딩 후 얼마나 잘 보존되었는지를 평가
  - ✓ DLM (Detail Loss Metric, DLM): 영상의 디테일이 사라지거나 흐릿해지는 현상을 측정, 고주파수 영역의 디테일 손실에 집중
- Temporal feature extraction (시간적 특징 추출)
  - ✓ TI (Temporal Information): 현재 프레임의 특정 블록과 이전 프레임의 동일 위치 블록을 비교하여 얼마나 변화(움직임)이 있었는지 측정, 시각적 마스킹 효과를 반영하기 위한 지표

2. Frame 레벨: Spatial Pooling & Fusion
   - 프레임 내 공간적 풀링: 1단계에서 얻은 한 프레임 내의 모든 블록 점수들을 가중합으로 계산하여 프레임 전체의 VIF, DLM, TI 대표값을 산출
   - SVM prediction (Fusion): 프레임 단위로 요약된 특징값을 훈련된 SVM 머신러닝 모델에 입력하여 해당 프레임에 대한 최종 품질 점수를 예측
3. Video 레벨: Temporal Pooling
   - 시간적 풀링: 2단계에서 계산된 모든 프레임의 점수(per-frame scores)를 다시 모아 하나의 최종 대표값으로 계산

3.5.2.4. DMOS 상관관계 비교

본 과제에서는 품질 평가 지표로 VMAF를 선택하였는데, 이는 사람이 인지하는 화질 수준과 가장 유사하며 인간의 시각적 판단을 잘 반영하기 때문이다. 이를 증명하기 위해 DMOS (Different Mean Opinion Score)과 비교해볼 수 있다. DMOS는 주관적 품질 평가 방법을 통해 얻어지는 점수로, 원본 영상과 비교해서 인코딩 된 비디오가 얼마나 손상되었는지에 대한 사람들의 평가를 나타낸다. DMOS는 인간의 시각적 인식을 직접적으로 반영하므로, 다른 품질 평가 지표들이 인간 시각을 얼마나 잘 반영하는지 평가하는 기준이 된다. 실험 결과를 보면, VMAF가 다른 품질 평가에 비해 DMOS와 정비례 관계에 가까우므로 높은 상관 관계를 가지고 있음을 알 수 있다.

본 과제에서 VMAF로 인코딩 영상이 얼마나 품질이 저하되었는지 인간의 시각 시스템 기준으로 측정하고, 품질 저하를 최소화하는 동시에 비트레이트를 효율적으로 관리한다.

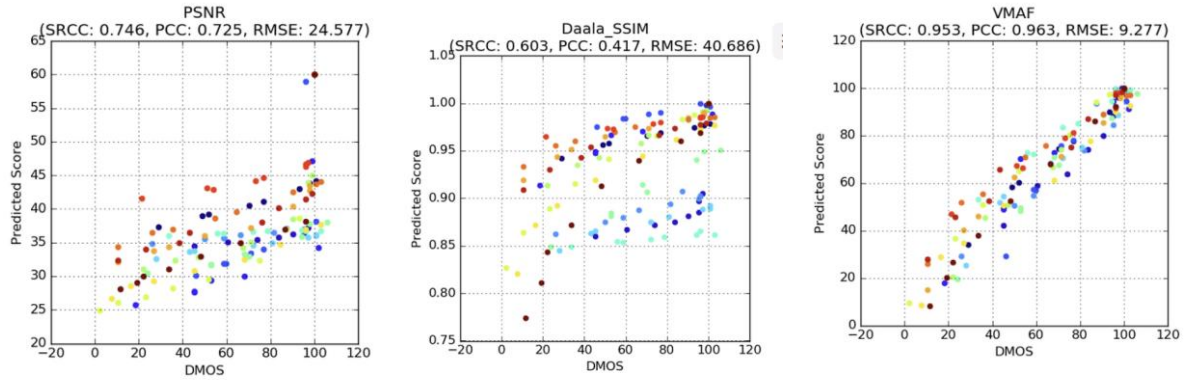


그림 31. PSNR, SSIM, VMAF 비교

3.5.3. 최적 인코딩 파라미터 탐색 전략

- 초기 탐색 전략

초기 모델은 최적 인코딩 파라미터를 탐색하기 위해 CRF와 Max rate를 각 4개씩 기준을 정하고 총 16개의 파라미터로 인코딩하여 비교했다. 그 결과 다음과 같이 최적 인코딩 파라미터 선정 결과를 볼 수 있었다.

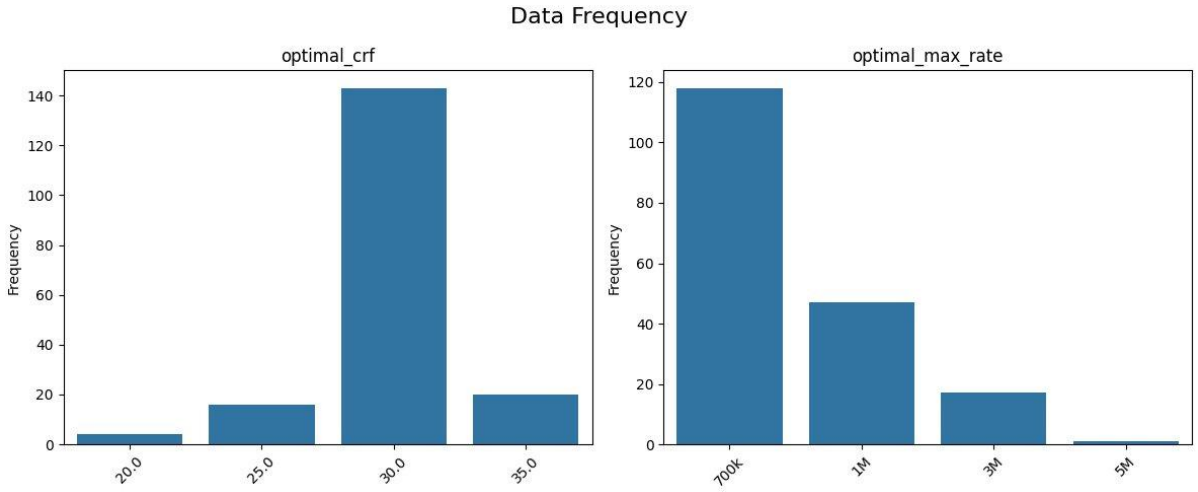


그림 32. 초기 최적화 결과

대부분의 세그먼트가 CRF 30, Max rate 700k 구간에서 최적 인코딩 파라미터를 선택한다. 이렇게 인코딩 파라미터의 다양성을 챙길 수 없고, 대부분 한 범위로 몰리는 경우가 생기는 문제가 발생하여 다음과 같이 2단계로 나누어 다시 최적화를 시행했다.

- 최종 탐색 전략

1단계: 최적 CRF 탐색

1. 입력: 원본 영상 세그먼트와 탐색할 CRF 값 범위(18~43) 준비
2. 반복 인코딩: 준비된 영상 세그먼트를 CRF 18, 20, 22… 등 모든 CRF 값으로 각각 인코딩, 이 때 Max Rate 옵션은 None으로 지정하여 CRF의 영향만 측정
3. 성능 측정: 인코딩 된 모든 결과물에 대해 VMAF 점수와 비트레이트 측정
4. 최적 CRF 결정: 측정 결과 중에서 '목표 VMAF 점수를 만족하면서 비트레이트가 가장 낮은' 최적 CRF 값을 탐색

2단계: 최적 Max Rate 탐색

1. 입력: 1단계에서 사용했던 원본 영상 세그먼트, 1단계에서 찾아낸 최적 CRF, 원본 영상 세그먼트의 평균 비트레이트, 평균 비트레이트에 적용할 승수(Multiplier) 범위 (1.0~3.0)
2. 반복 인코딩: 영상 세그먼트를 최적 CRF로 고정한 채, 승수를 바꿔가며 계산된 Max rate 값으로 각각 인코딩
3. 성능 측정: 인코딩 된 모든 결과물에 대해 VMAF 점수와 비트레이트를 다시 측정
4. 최적 Max rate 결정: 측정 결과 중에서 '목표 VMAF 점수를 만족하면서 비트레이트가 가장 낮은' 최적 Max rate 값을 탐색

3.5.4. 인코딩 방식별 효율성 비교 실험

CRF 기반의 파라미터 최적화가 왜 효율적인지를 입증하기 위해, 전통적인 CBR 방식과 비트레이트 효율성을 비교하는 실험을 진행했다. 실험 목표는 'VMAF 90점'을 달성하기 위해 각 방식이 얼마나 많은 비트레이트를 필요로 하는지 확인하는 것이다.

- CBR 인코딩 결과

  CBR 방식은 목표 비트레이트를 높일수록 VMAF 점수가 증가하고 평균 비트레이트와 파일 크기가 증가한다. 전체적으로 일정한 비트레이트로 전송되며, VMAF 90점을 달성하기 위한 최소 CBR 비트레이트는 '900k'이다.

	파일 크기	평균 비트레이트	VMAF 점수
원본	12.40M	1.65M	NaN

700k	5.99M	0.80M	88.57
800k	6.71M	0.89M	88.94
900k	7.42M	0.99M	91.48

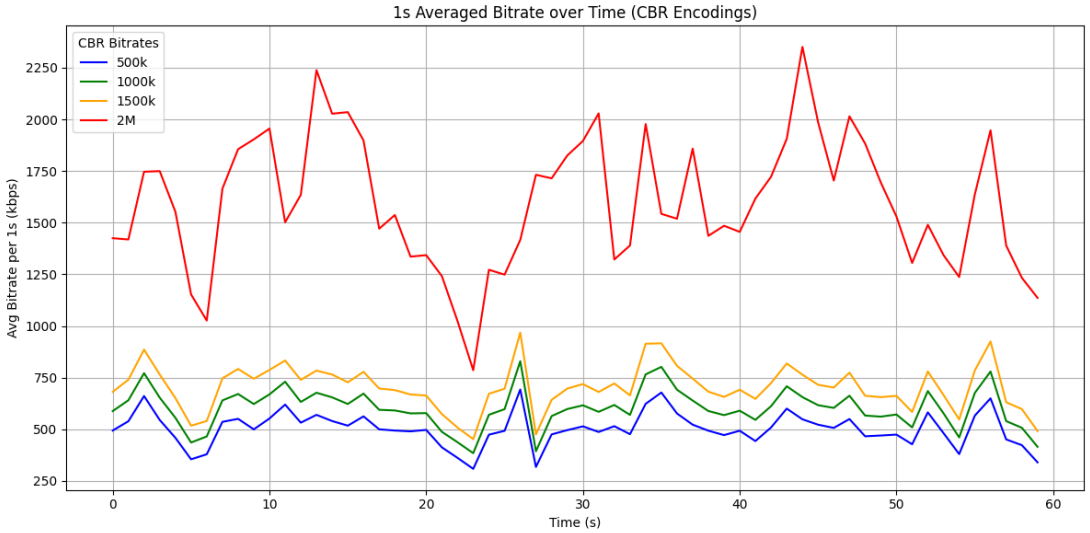
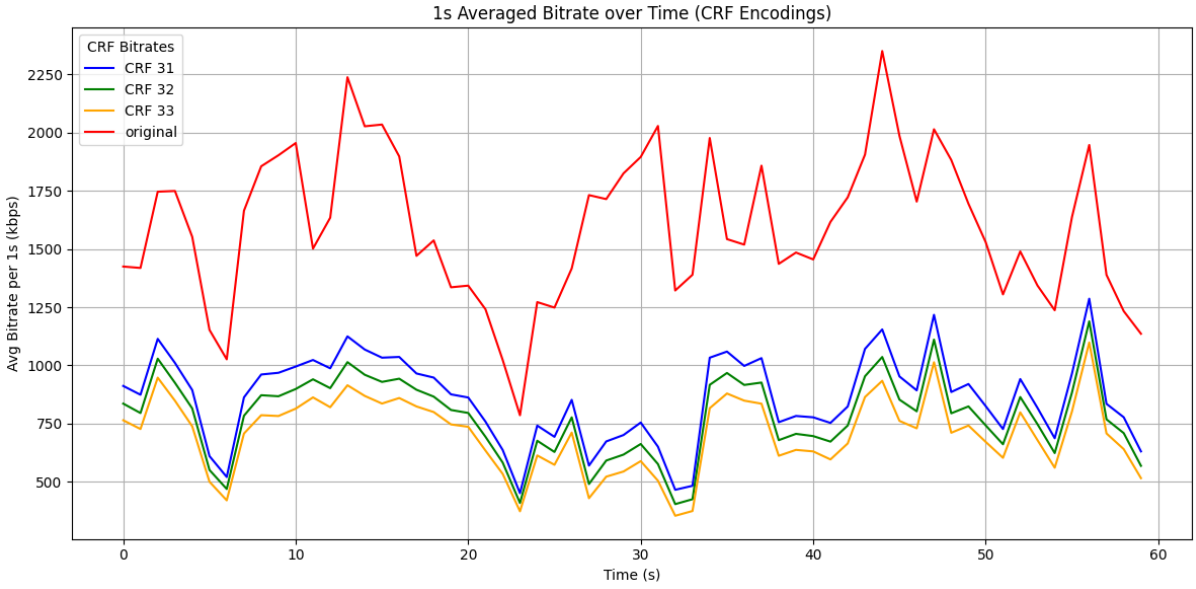


그림 33. CBR 비교

- CRF 인코딩 방식

CRF 인코딩은 CRF 값이 낮을수록 VMAF 점수가 증가하고 평균 비트레이트와 파일 크기가 증가한다. 구간의 복잡도에 따라 비트레이트가 달라지며 VMAF 90점을 달성하기 위한 CRF는 31이다.

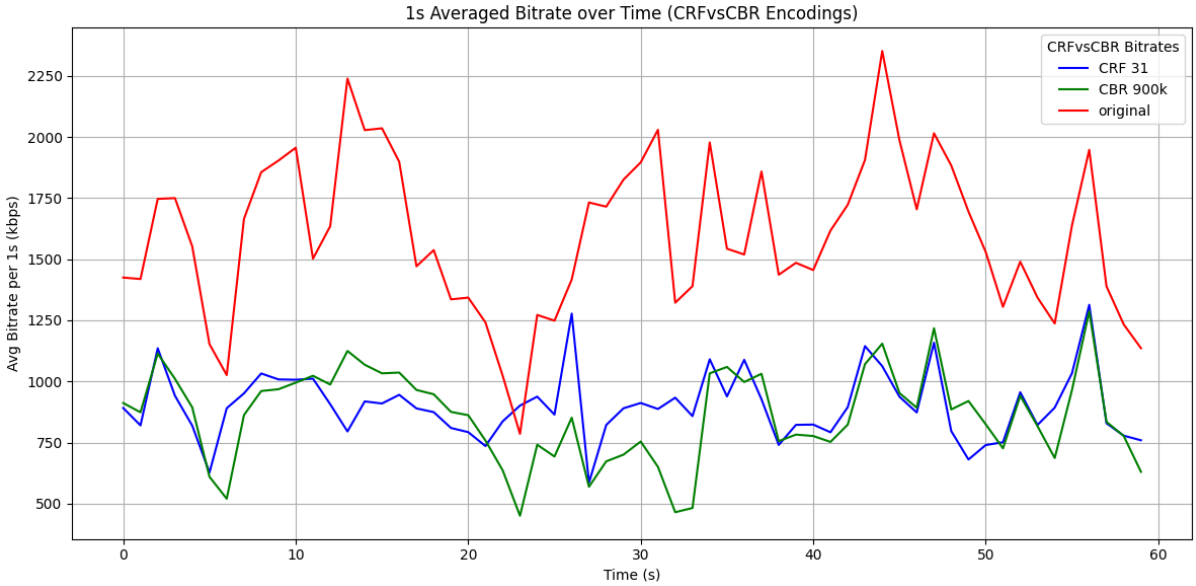
	파일 크기	평균 비트레이트	VMAF 점수
원본	12.40M	1.65M	NaN
CRF 31	7.13M	0.95M	91.71
CRF 32	6.55M	0.87M	98.68
CRF 33	6.03M	0.80M	87.42



- CBR과 CRF 비교

VMAF 90점을 달성하기 위한 최소 조건은 CBR 900k, CRF 31이다. 두 지표를 분석하면 비슷한 점수이면 CRF에서 비트레이트와 파일 크기가 작다. 또한 CRF 인코딩은 복잡도에 따라 비트레이트를 효율적으로 조절할 수 있다.

	파일 크기	평균 비트레이트	VMAF 점수
원본	12.40M	1.65M	NaN
CBR 900k	7.42M	0.99M	91.48
CRF 31	7.13M	0.95M	91.71



3.5.5. 모델 구현 및 훈련

3.5.5.1. RandomForest 모델

랜덤 포레스트(RF)는 Decision Tree를 여러 개 만들어 학습하는 앙상블 모델이다. 랜덤 포레스트는 특성 중요도를 제공하므로 해석이 비교적 쉽고 작은 데이터셋에서도 안정적인 성능을 보인다.

학습 결과 모델의 성능은 다음과 같다.

Metric	CRF Score	Max Rate Score
MSE	2.79	2003725.30
RMSE	1.67	451.35
MAE	1.29	299.44
R2	0.26	0.60

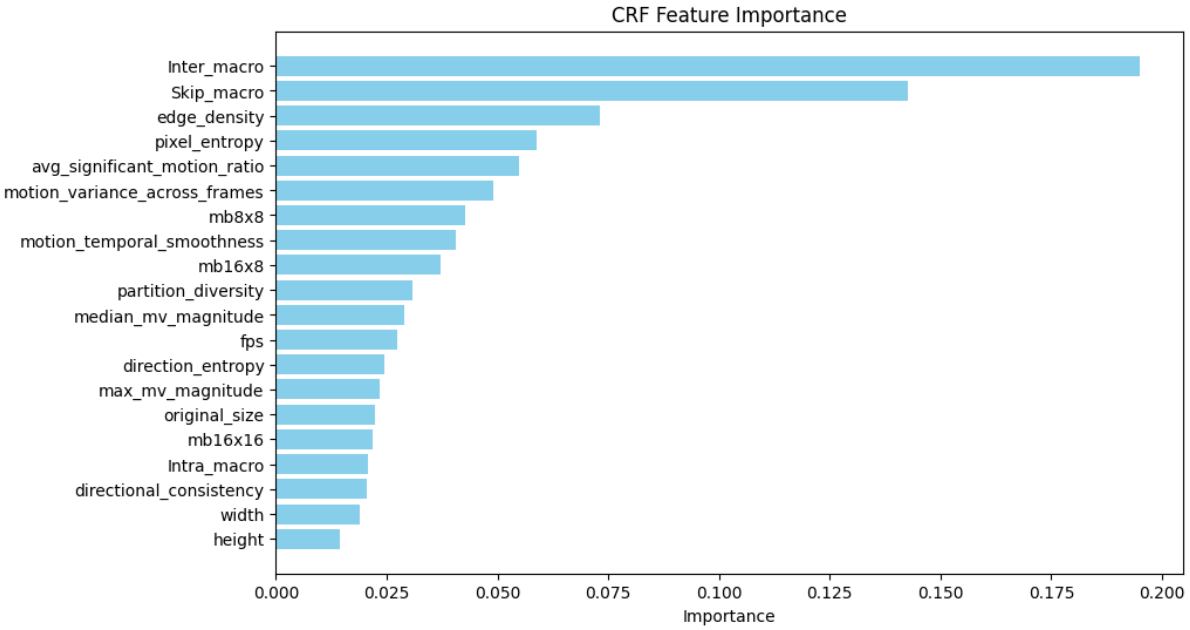
CRF 모델 성능은 괜찮지만, Max rate는 MAE 297로 성능이 좋지 않다고 생각했다.

낮은 성능의 원인을 분석해보았다.

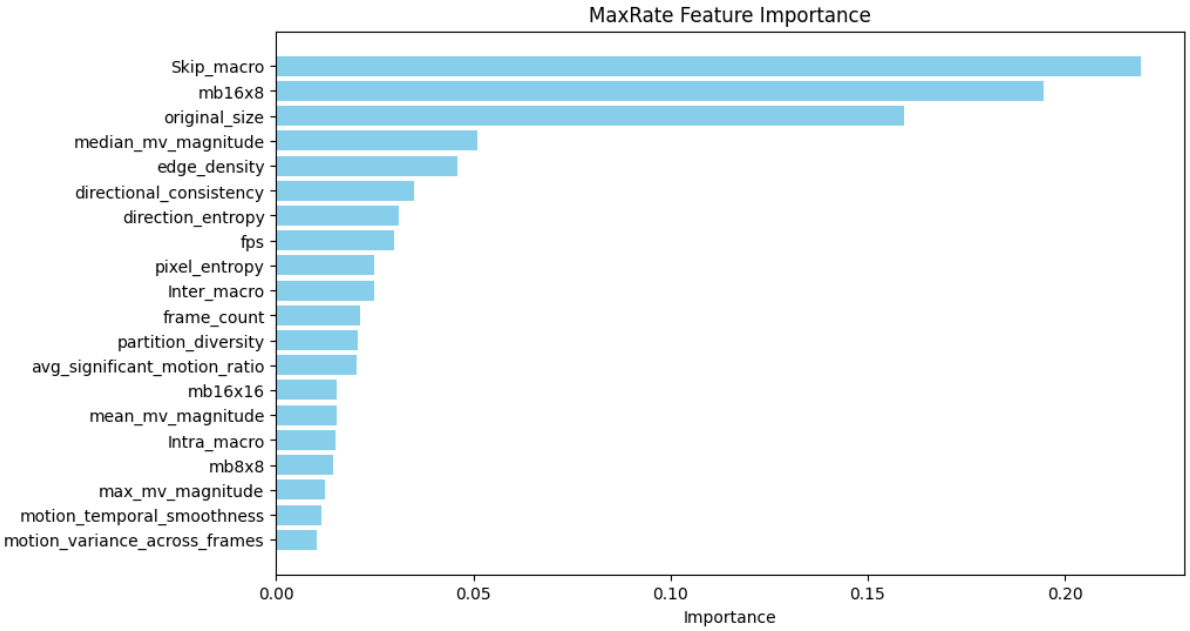
1. 복잡한 Max rate 결정 메커니즘: Max rate는 단순히 비디오의 시각적 복잡도만으로는 결정되지 않는다. 대역폭 제한, 타겟 디바이스 등의 디코딩 능력, 원하는 비트스트림 안정성 그리고 인코딩 목표 등 다양한 외부 요인들이 복잡적으로 상호작용하여 결정된다. 그러나 현재 input feature는 원본 비디오의 특성만 입력했으므로, max rate 결정에 영향을 미치는 외부 제약 조건 정보가 포함되어 있지 않아 정확히 예측하기 어렵다.
2. 데이터의 다양성 및 불균형: 수 kbps ~ 수천 kbps까지의 넓은 max rate 범위는 데이터셋에 저비트레이트부터 고비트레이트까지 다양한 인코딩 설정이 포함되어 있음을 의미한다. 만약 특정 Max rate 구간의 데이터가 부족하거나, Max rate 분포가 불균형하다면 모델 학습이 어려워질 수 있다.
3. 랜덤 포레스트 모델의 한계: 비디오 인코딩 파라미터 최적화는 비선형적이고 복잡한 관계를 가지는 경우가 많다. 랜덤 포레스트가 이러한 복잡성을 완전히 포착하지 못할 수도 있다.

- 특성 중요도 분석

RandomForest 모델은 특성 중요도를 제공한다. 초반에 랜덤 포레스트 모델을 사용한 것은 지나치게 많은 특성 중에서 파라미터 최적화에 중요하지 않은 특성은 제외하기 위해 사용했다.



다음과 같이 CRF를 최적화하는 데에는 주로 Inter 매크로블록과 Skip 매크로블록의 비율, Edge density, pixel 엔트로피와 같은 공간적 복잡도, 의미 있는 모션 벡터의 비율 등이 특성 중요도가 높았다.



Max rate의 최적화에는 주로 Skip Macro 블록과 파티션 사이즈, 원본 파일 크기, edge density, direction entropy 등의 특성 중요도가 높았다.

- 하이퍼 파라미터 튜닝

  모델의 예측 성능을 극대화하고 일반화 오류를 최소화하기 위해 하이퍼파라미터 튜닝을 진행했다. 탐색 과정의 효율성을 고려하여, 모든 조합을 탐색하는 Grid Search 방식 대신 RandomizedSearchCV 방법을 채택했다. RandomizedSearchCV는 지정된 범위 내에서 하이퍼파라미터 조합을 무작위로 샘플링 하여 교차 검증을 수행하므로, 넓은 탐색 공간을 효율적으로 탐색하며 최적의 해를 근사적으로 찾아낸다.

  수행 결과, 아래 <표 1>에 제시된 하이퍼파라미터 조합에서 가장 뛰어난 검증 성능을 기록했다. 따라서 해당 조합을 RandomForest 모델의 구성으로 확정하고 이를 기반으로 전체 훈련 데이터셋에 대한 학습을 진행했다.

하이퍼파라미터 (Hyperparameter)	CRF model	Max rate model	설명
n_estimators	100	500	앙상블을 구성하는 의사결정 트리의 개수
min_samples_split	2	5	노드를 분할하기 위한 최소 샘플 수
min_samples_leaf	1	2	리프 노드가 되기 위한 최소 샘플 수
max_features	Log2	Sqrt	최적의 분할을 위해 고려할 최대 피쳐의 수
max_depth	40	None	트리의 최대 깊이

표 1 RandomForest 하이퍼파라미터 튜닝 결과

3.5.5.2. XGBoost

XGBoost는 여러 개의 의사결정 트리를 묶어 사용하는 강력한 앙상블 알고리즘이다. 이전 트리가 틀린 문제에 다음 트리가 집중하며 순차적으로 오차를 보완해 나간다. 뛰어난 최적화 기술로 속도가 매우 빠르고, 과적합 방지 기능으로 예측 성능이 높다. 이러한 장점 덕분에 XGBoost는 정형 데이터를 다루는 문제에서 가장 선호되는 알고리즘 중 하나다.

Metric	CRF	Max Rate Score
MSE	2.57	176690.39
RMSE	1.60	420.34

MAE	1.22	288.81
R2	0.56	0.60

표 2. XGBoost 모델 평가

- 하이퍼 파라미터 튜닝

하이퍼파라미터 (Hyperparameter)	CRF model	Max rate model	설명
subsample	0.8	0.8	각 트리를 만들 때 무작위 샘플링 비율
n_estimators	400	200	앙상블을 구성할 의사결정 트리의 개수
min_child_weight	10	10	하나의 리프 노드에 필요한 최소한의 데이터 가중치 합
max_depth	3	3	의사 결정 트리가 가질 수 있는 최대 깊이
learning_rate	0.05	0.1	각 트리가 이전 트리의 오차를 보정하는 정도
colsample_bytree	1.0	1.0	각 트리를 만들 때 전체 특성을 모두 사용

- 손실 함수 곡선

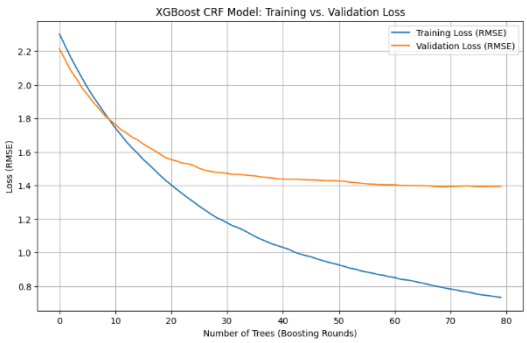


표 3. XGBoost CRF 손실 함수 곡선

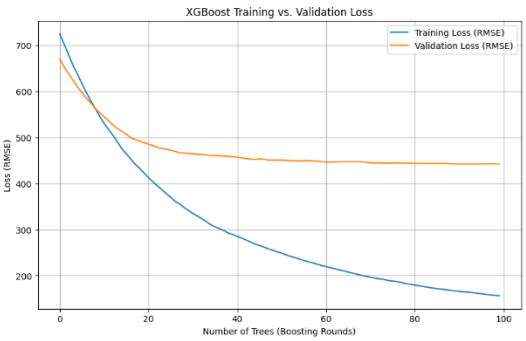


표 4. XGBoost Max rate 손실 함수

3.5.5.3. MLPRegressor (Multi-Layer Perceptron)

MLP 모델은 다층 퍼셉트론 모델로, 입력층(Input Layer)과 출력층(Output Layer) 사이에 하나 이상의 은닉층(Hidden Layer)을 포함하는 인공 신경망(ANN)의 대표적인 순방향 구조이다. 활성화 함수를 통해 비선형성을 도입하므로, 복잡한 비선형 관계를 효과적으로 학습할 수 있으나 모델의 복잡도와 파라미터 수가 많아 많은 양의 훈련 데이터를 필요로 한다.

Metric	CRF Score	Max Rate Score
MSE	3.42	179137.37
RMSE	1.85	423.24
MAE	1.42	300.13
R2	0.42	0.60

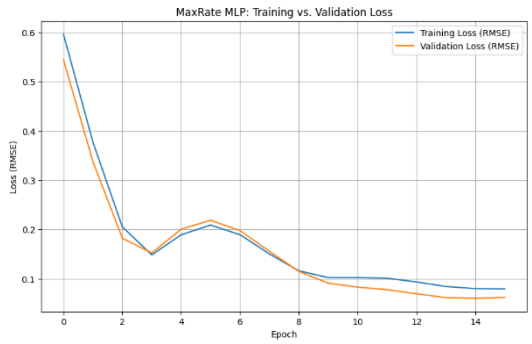
표 5. MLPRegressor 모델 평가

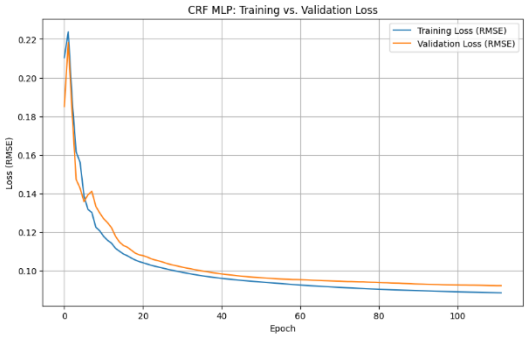
- 하이퍼파라미터 튜닝 결과

하이퍼파라미터 (Hyperparameter)	CRF 모델 최적값	Max Rate 모델 최적값	설명
Solver	Adam	Adam	가중치 최적화를 위한 옵티마이저
Learning_rate	Constant	Constant	학습률을 훈련 내내 상수로 유지하는 방식
Hidden_layer_sizes	(80,)	(100,)	모델의 은닉층 구조
Alpha	0.1	0.001	L2 정규화 강도
Activation	relu	relu	은닉층에서 사용할 활성화 함수

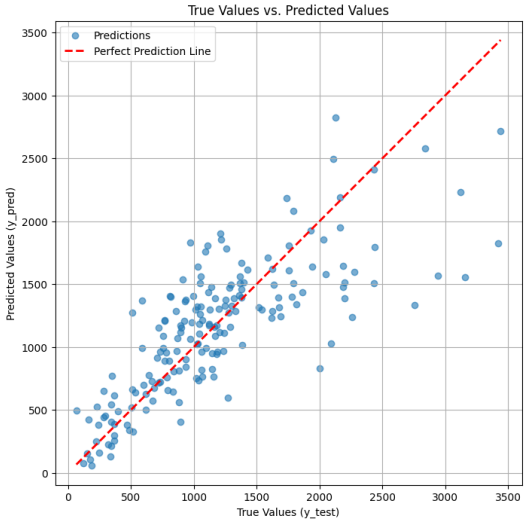
표 6. MLP 하이퍼파라미터 튜닝 결과

- 훈련 손실 곡선





- Max Rate 산점도 그래프



3.6. 실시간 영상 분석 및 최적화

3.6.1. 인코딩 파라미터 예측 결과

```
[husky_encode13]=== 영상 특성 추출 및 인코딩 파라미터 최적화 ===
[segment_0 원본: 1280x720] CRF=35, maxrate=8291k
[segment_0 640x360] CRF=34, maxrate=2073k
[segment_0 854x480] CRF=34, maxrate=3688k
[segment_0 1280x720] CRF=35, maxrate=8291k
[segment_0 1920x1080] CRF=35, maxrate=18655k
[segment_1 원본: 1280x720] CRF=35, maxrate=8287k
[segment_1 640x360] CRF=34, maxrate=2072k
[segment_1 854x480] CRF=34, maxrate=3686k
[segment_1 1280x720] CRF=35, maxrate=8287k
[segment_1 1920x1080] CRF=35, maxrate=18646k
[segment_2 원본: 1280x720] CRF=35, maxrate=8207k
```

그림 34. 인코딩 파라미터

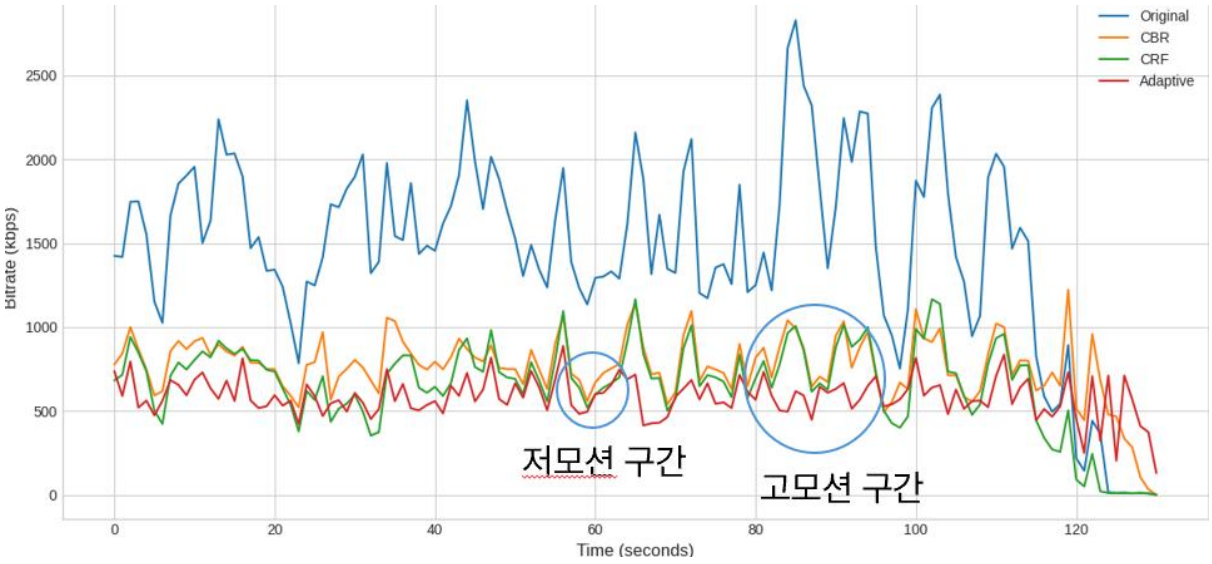
모델을 스트리밍 시스템에 적용하면 다음과 같이 세그먼트마다 최적화된 CRF, Max rate를 예측하는 것을 확인할 수 있다.

3.6.2. CBR, CRF, 최적화 모델 비교

최적화 모델 적용 후, 원본 대비 약 59.85%의 비트레이트 절감 효과를 보였으며, 기존 CRF 방식 대비 약 10%의 비트레이트를 절감할 수 있다.

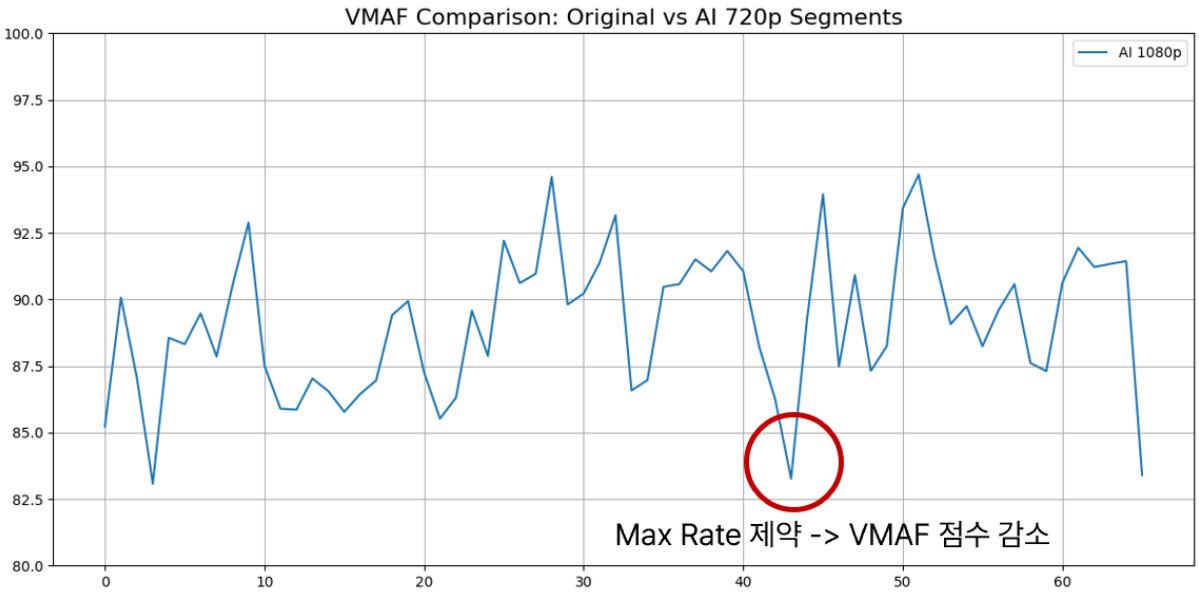
인코딩 방식	평균 비트레이트 (kbps)	Savings
원본	1254.96	N/A
CBR	761.59	47.66%
CRF	656.98	54.85%
최적화 모델	584.16	59.85%

3.6.3. 구간별 비트레이트 분석



저모션 구간은 시각적 마스킹이 거의 적용되지 않아 비트레이트를 과도하게 줄이면 품질이 저하되므로 최대한 원본과 비슷한 수준으로 비트레이트를 유지한다. 고모션 구간은 시각적 마스킹 효과로 인해 비트레이트를 줄여 화질이 저하되더라도 인간 시각 시스템이 이를 거의 감지하지 못한다. 따라서 CRF를 높이는, 즉 비트레이트를 최대한 줄일 수 있다. 그래프에서도 고모션 영상에서는 비트레이트 절감이 특히 크다는 것을 확인할 수 있다.

3.6.4. VMAF 품질 평가



각 세그먼트에 대해서 VMAF 품질 평가를 시행한 결과, 평균적으로 89.5점이 나왔다. 이는 기준 VMAF 점수가 90이었으므로 매우 양호한 품질이라고 볼 수 있다. 그러나 표시한 부분처럼 VMAF 점수가 82점까지 떨어지는 세그먼트가 있는데, 이는 Max rate 제약으로 인해 비트레이트를 충분히 전송하지 못해 원본과 비교하여 품질이 떨어진 것으로 본다.

3.7. 웹 서비스

3.7.1. 사전 준비

3.7.1.1. 로그인 방법 구성

Firebase 프로젝트를 생성한 후 로그인 방법으로 이메일/비밀번호와 Google 계정 로그인을 활성화시켰다.

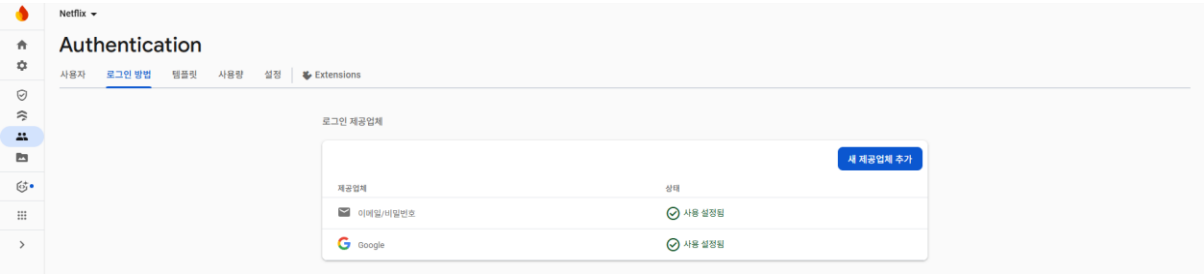


그림 35. 로그인 방법

3.7.1.2. 사용자 인증 설정

Firebase Authentication을 활용하여 사용자 계정을 관리(비밀번호 재설정, 계정 사용 중지, 계정 삭제) 할 수 있다. 각 사용자는 Firebase에서 발급되는 고유한 UID를 가지며, 이를 기반으로 Firebase 데이터베이스와 연동된다. 이를 통해 사용자별 데이터를 저장하고 관리할 수 있다.

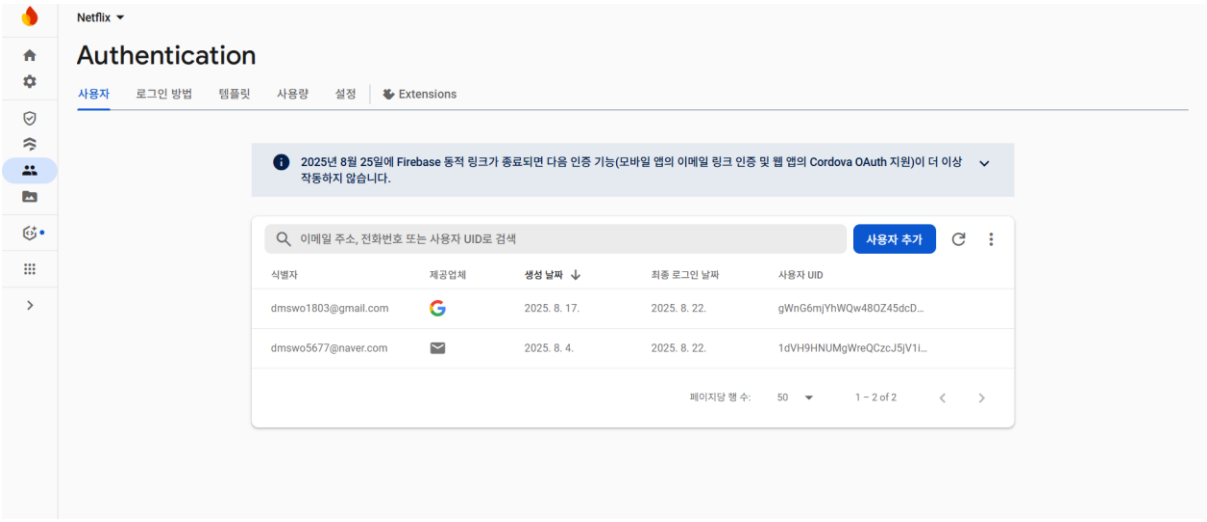


그림 36. 사용자 인증 설정

3.7.1.3. Firestore 데이터 구조

Cloud Firestore는 사용자 계정마다 다양한 데이터를 저장하고 관리한다. 본 프로젝트에서는 사용자별로 Users, LikedMovies, MyList, WatchedMovies 컬렉션을 생성하여, 각 UID를 기반으로 문서를 생성하여 관리한다. [그림 39]는 특정 사용자의 WatchedMovies에 저장된 데이터로, 사용자가 시청한 영상의 정보가 포함되어 있다. 이를 통해 사용자는 자신의 시청 기록, 좋아요를 누른 영상, 나중에 볼 영상 목록을 조회하고 관리할 수 있다.

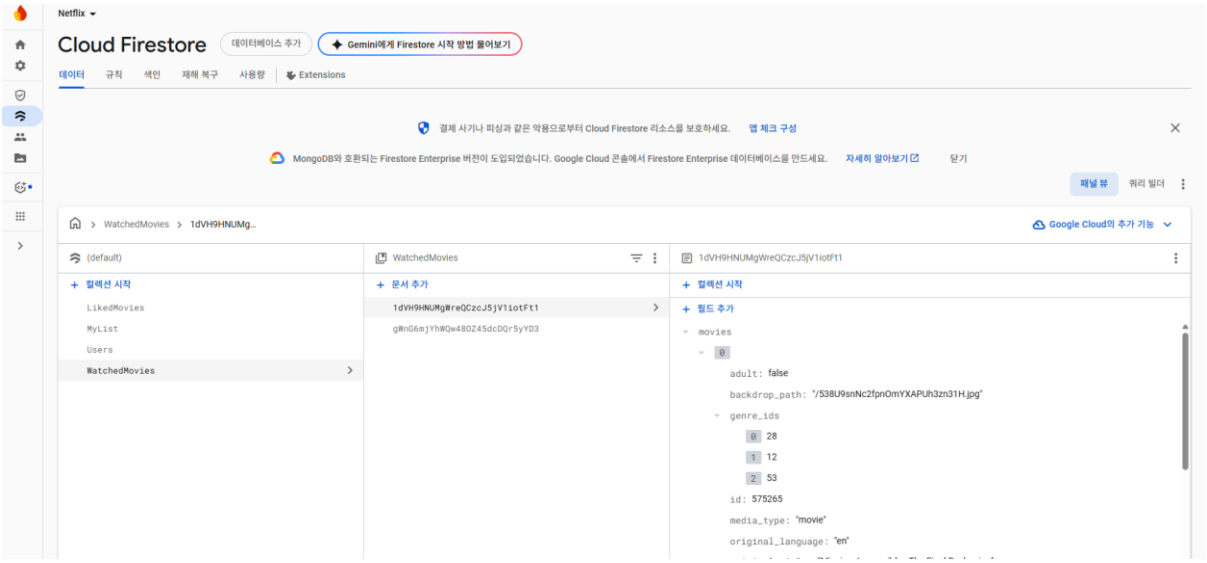


그림 37. Cloud Firestore

3.7.1.4. Cloud Firestore 보안 규칙 설정

보안 규칙을 설정하여 각 경로(Users/{uid}, MyList/{uid}, WatchedMovies/{uid}, LikedMovies/{uid})를 로그인 된 사용자 uid와 동일할 때만 읽기, 쓰기 권한을 허용하였다. 이를 통해 다른 사용자의 데이터에 접근하지 못하도록 권한을 제한하였다.

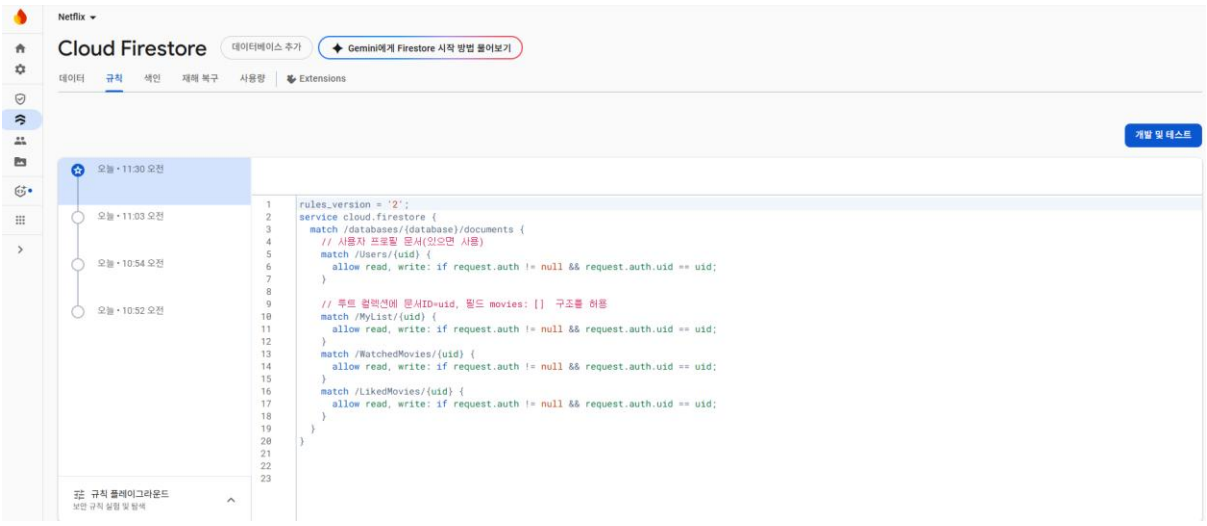


그림 38. 보안 규칙 설정

3.7.2. 유스케이스 다이어그램

스트리밍 시스템의 유스케이스 다이어그램이다.

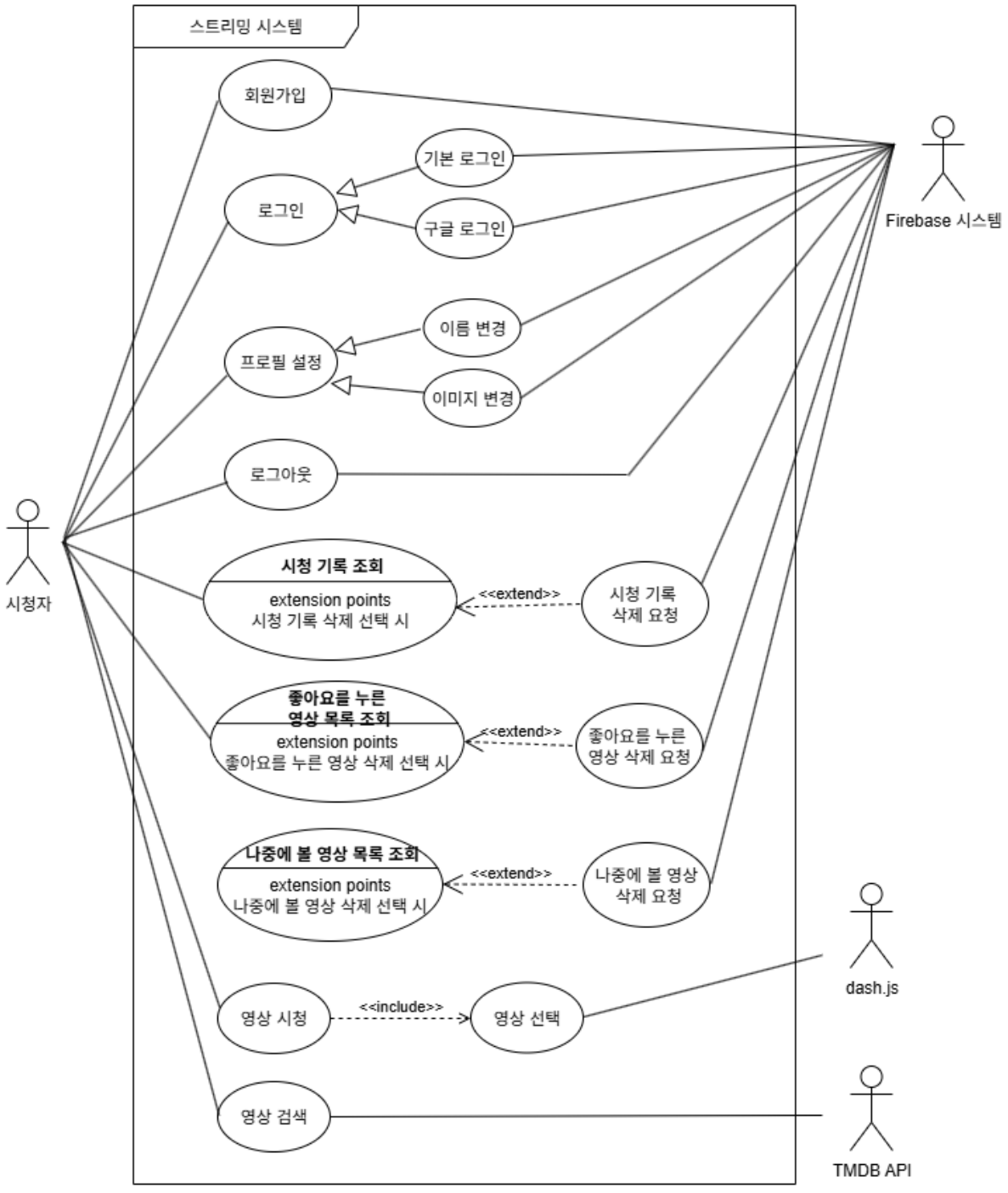


그림 39. 유스케이스 다이어그램

3.7.3. 시퀀스 다이어그램

3.7.3.1. 회원가입 시퀀스 다이어그램

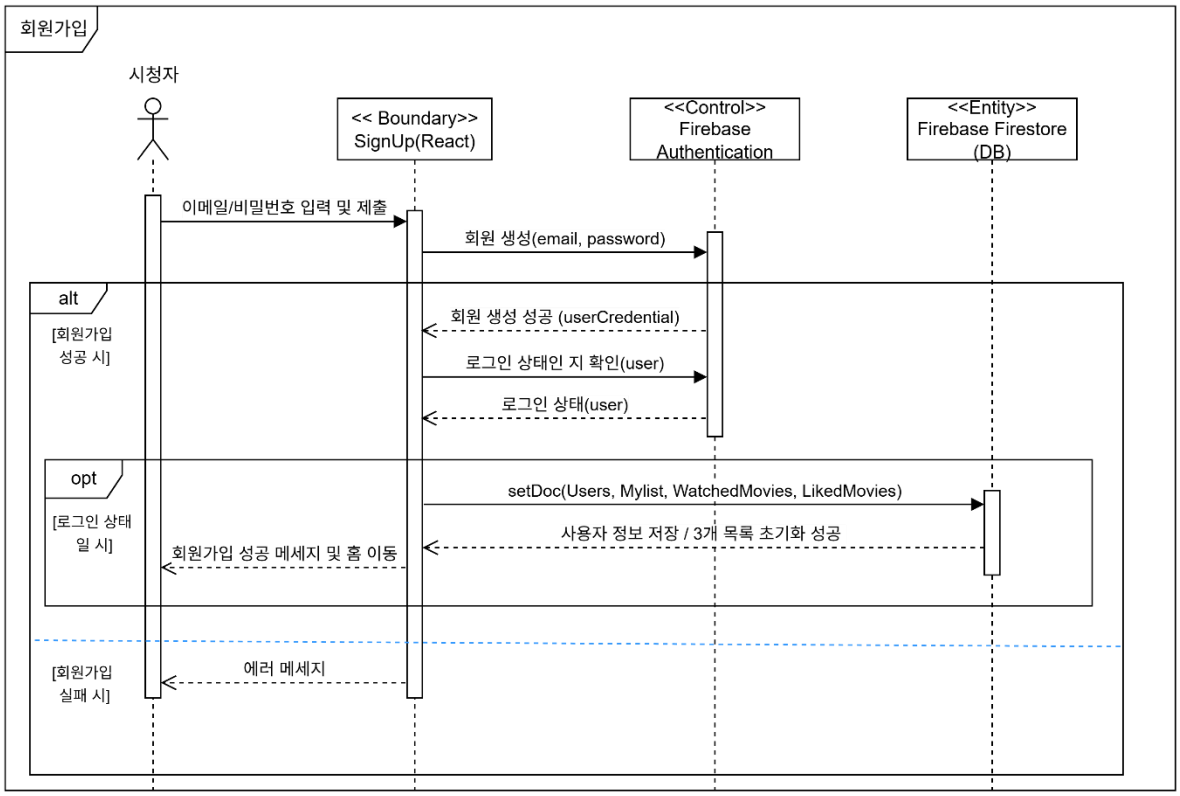


그림 40. 회원가입 시퀀스 다이어그램

1. 시청자가 회원가입 요청

- 이메일, 비밀번호를 입력한 뒤, 회원가입 버튼 클릭

2. 회원 생성

- 이메일, 비밀번호 정보를 바탕으로 회원 생성

3-1. 회원가입 성공 시

- 새로 생성된 사용자 정보와 인증 관련 데이터를 담고 있는 userCredential 객체 반환
- user 정보를 바탕으로 회원가입 이후, 사용자가 실제 로그인 된 상태인지 확인

3-1-1. 로그인 상태일 시

- 사용자 정보(email, uid) 저장
- 사용자 전용 MyList, WatchedMoives, LikedMovies 목록을 빈 배열로 초기화 (나중에 사용자가 각 목록에 맞는 영화를 추가하거나 삭

제할 수 있게 해주는 작업)

- 회원가입 성공 메시지 및 홈 이동

3-2. 회원가입 실패 시

- 에러 메시지

3.7.3.2. 로그인 시퀀스 다이어그램

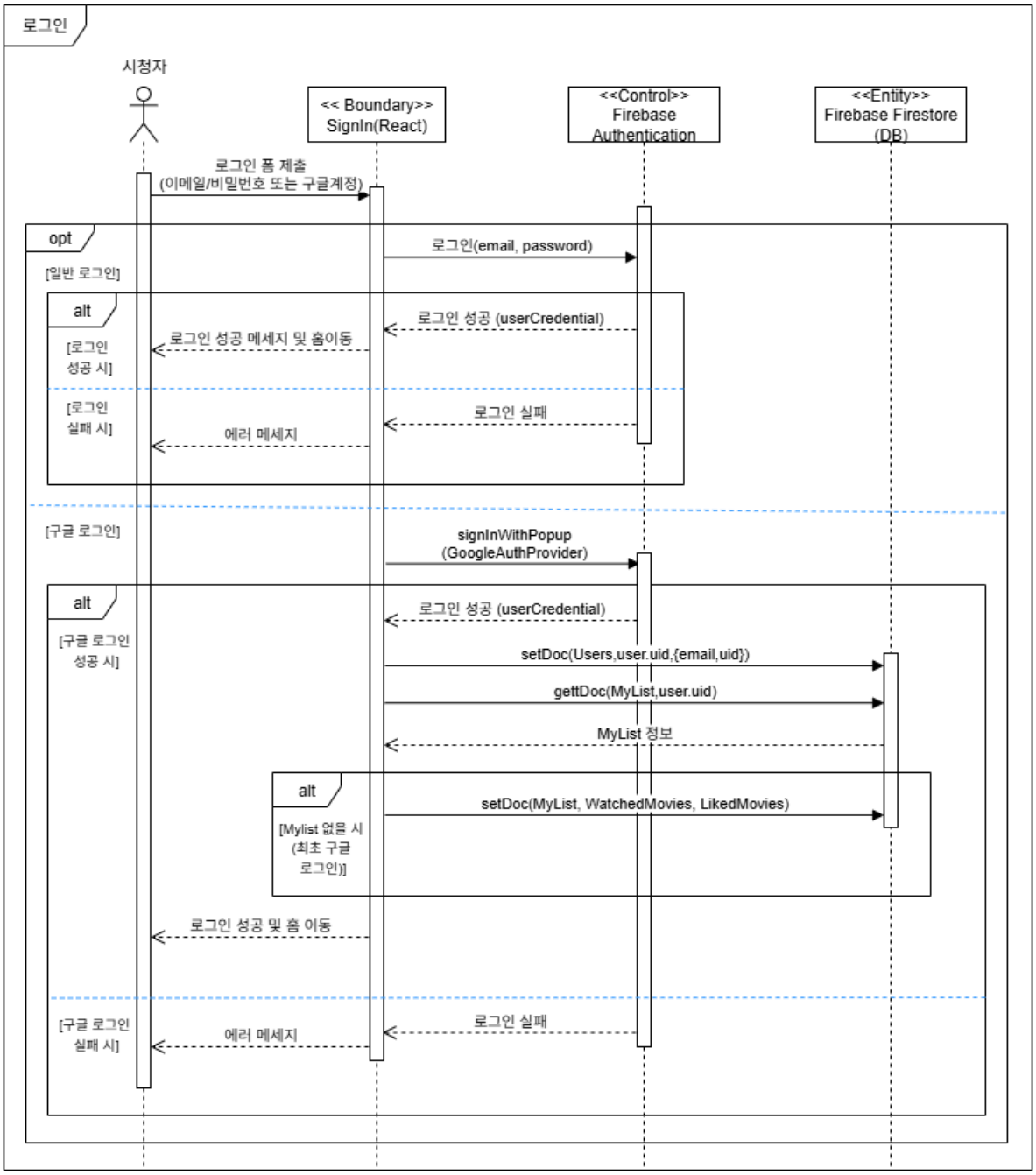


그림 41. 로그인 시퀀스 다이어그램

1. 시청자가 로그인 요청

- 이메일/비밀번호 (일반 로그인) 또는 구글 계정 (구글 로그인) 폼 제출

2-1. 일반 로그인

- email, password 정보를 바탕으로 로그인

2-1-1. 일반 로그인 성공 시

- 로그인 성공 메시지 및 홈 이동

2-1-2. 일반 로그인 실패 시

- 에러 메시지

2-2. 구글 로그인

- 구글 로그인 팝업을 사용하여 로그인 처리

2-2-1. 구글 로그인 성공 시

- result (userCredential과 동일한 구조) 반환
- 사용자 정보 저장(email, uid)
- 해당 사용자의 MyList 정보 가져와서 존재하지 않으면, (최초 구글 로그인 시) Mylist, WatchedMovies, LikedMovies 목록 초기화
  (구글 로그인은 회원가입시 수행하는 사용자 별 3가지 목록을 초기화하지 않았기 때문에 최초 로그인 시 해당 작업 수행)
- 로그인 성공 및 홈 이동

2-2-2. 구글 로그인 실패 시

- 에러 메시지

3.7.3.3. 로그아웃 시퀀스 다이어그램

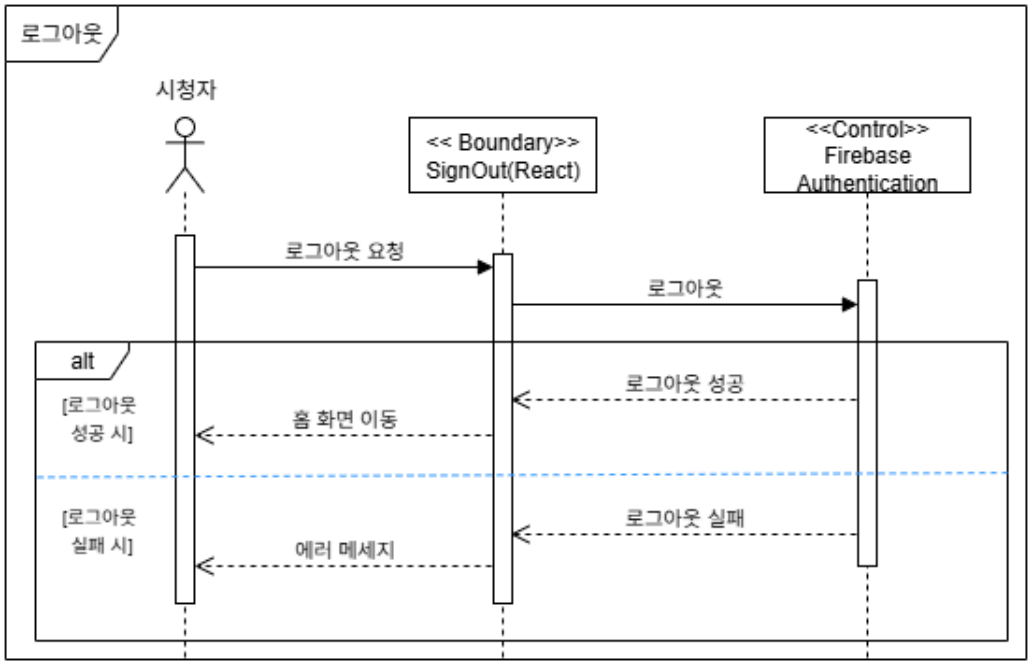


그림 42. 로그아웃 시퀀스 다이어그램

3.7.3.4. 프로필 이름 변경 시퀀스 다이어그램

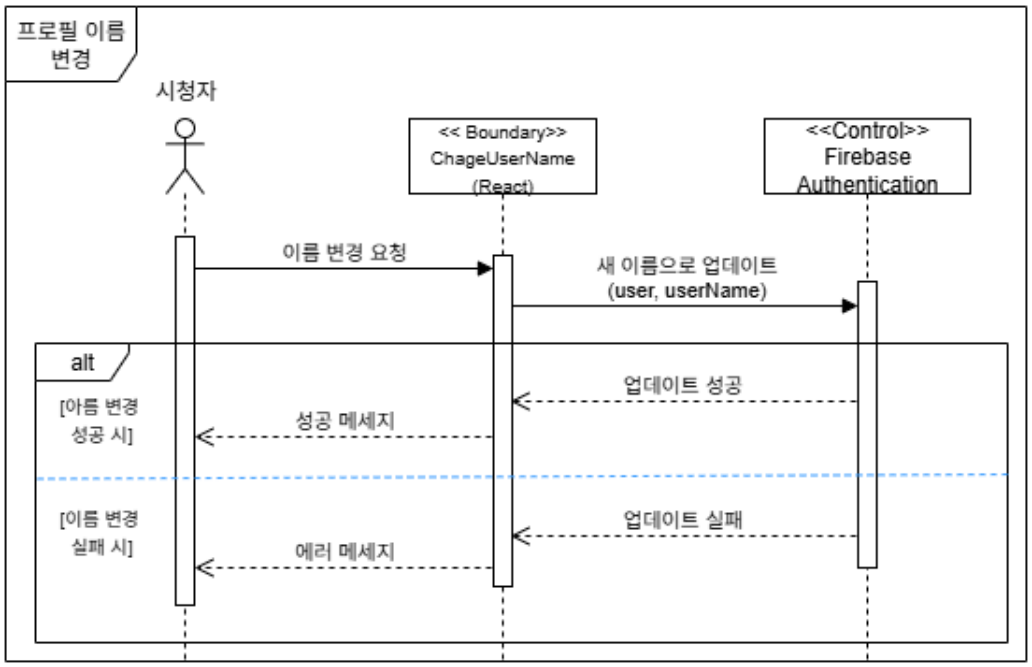


그림 43. 프로필 이름 변경 시퀀스 다이어그램

3.7.3.5. 프로필 이미지 변경 시퀀스 다이어그램

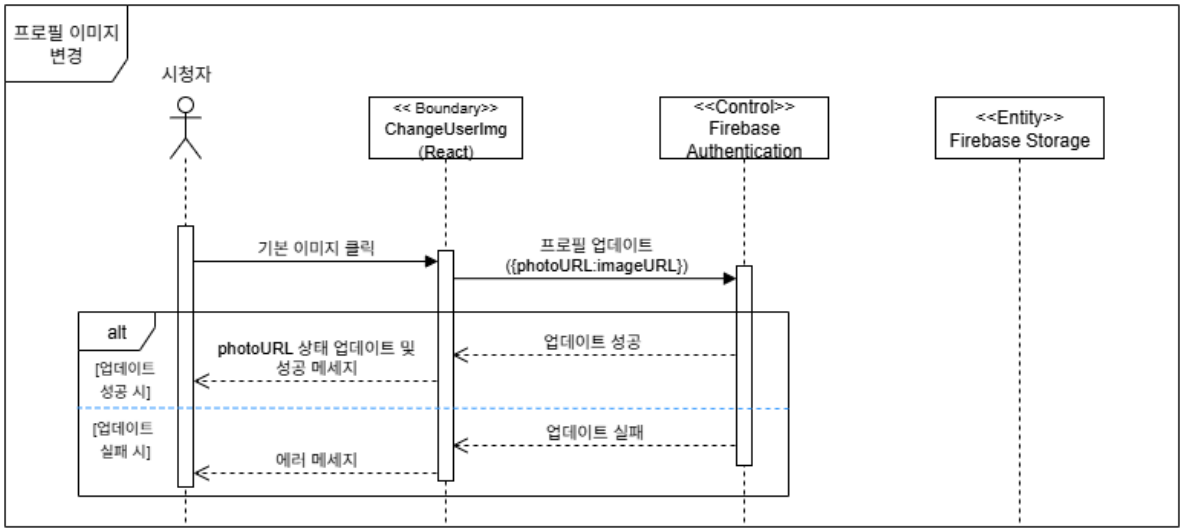


그림 44. 프로필 이미지 변경 시퀀스 다이어그램

1. 기본 이미지로 변경

- 제시되어 있는 4개의 기본 이미지 중 프로필 사진으로 바꿀 이미지 1개 선택
- 프로필 업데이트 (imageURL을 받아와서 firebase auth에 업데이트
  -> 프로필에 반영)

2-1. 업데이트 성공 시

- photoURL 상태 업데이트 및 성공 메시지

2-2. 업데이트 실패 시

- 에러 메시지

3.7.3.6. 시청 기록 조회 시퀀스 다이어그램

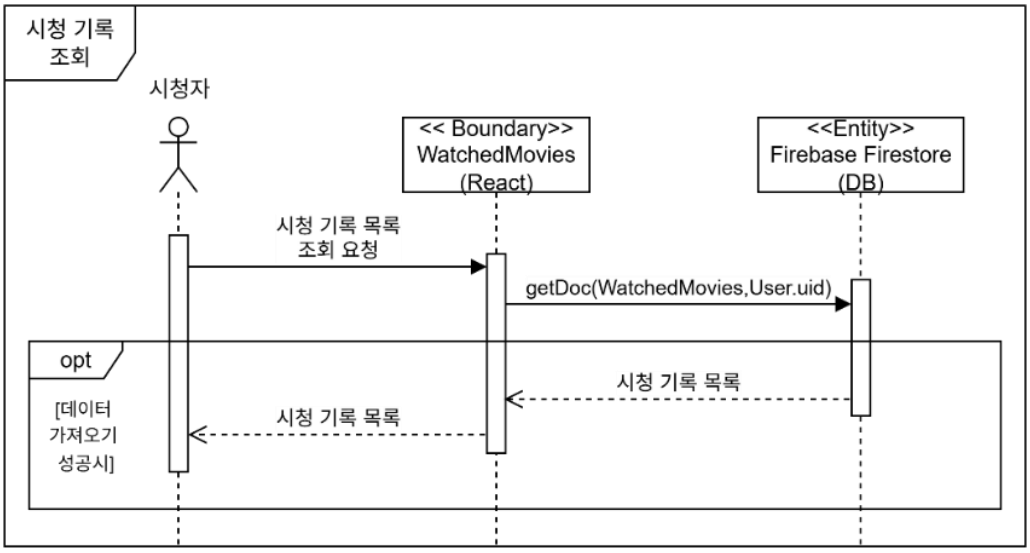


그림 45. 시청 기록 조회 시퀀스 다이어그램

1. 사용자가 지금까지 시청한 영상 목록 조회를 요청

- 사용자 별 WatchedMovies 목록 가져옴
- 데이터를 성공적으로 가져오면 시청 기록 목록을 보여줌

3.7.3.7. 시청 기록 삭제 시퀀스 다이어그램

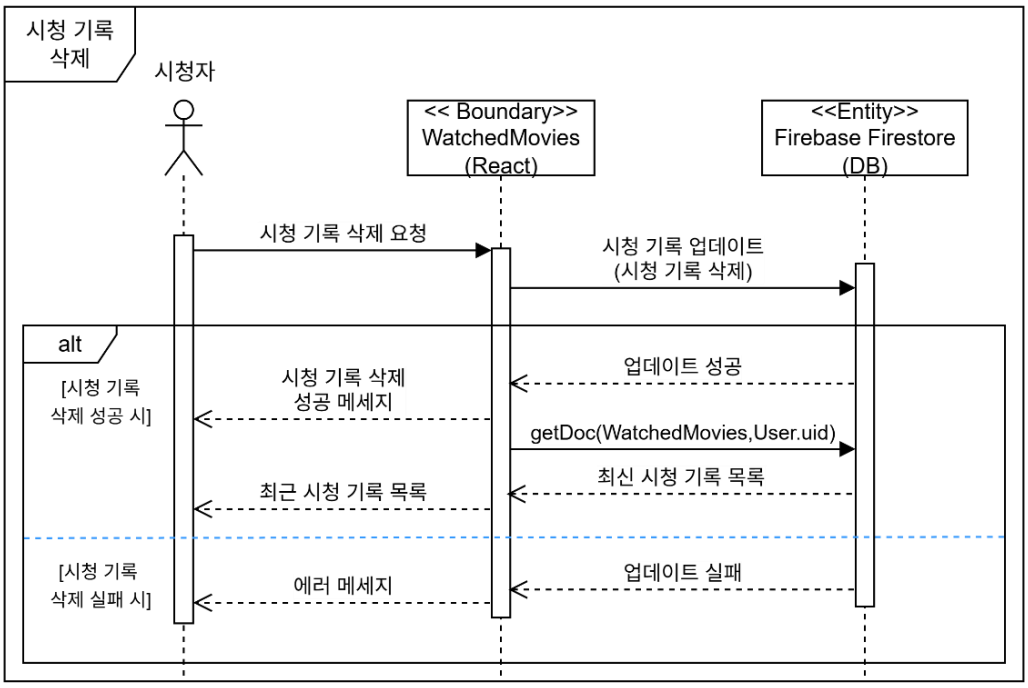


그림 46. 시청 기록 삭제 시퀀스 다이어그램

1. 사용자가 지금까지 시청한 영상 시청 기록 중 특정 영상의 시청 기록 삭제 요청

- 특정 영상의 시청 기록을 삭제하고, 시청 기록 업데이트 (UI에 반영)

2-1. 시청 기록 삭제 성공 시

- 시청 기록 삭제 성공 메시지
- 사용자 별 WatchedMovies 목록을 가져와서 업데이트 된 최신 시청 기록을 보여줌

2-2. 시청 기록 삭제 실패 시

- 에러 메시지

3.7.3.8. 좋아요를 누른 영상 목록 조회 시퀀스 다이어그램

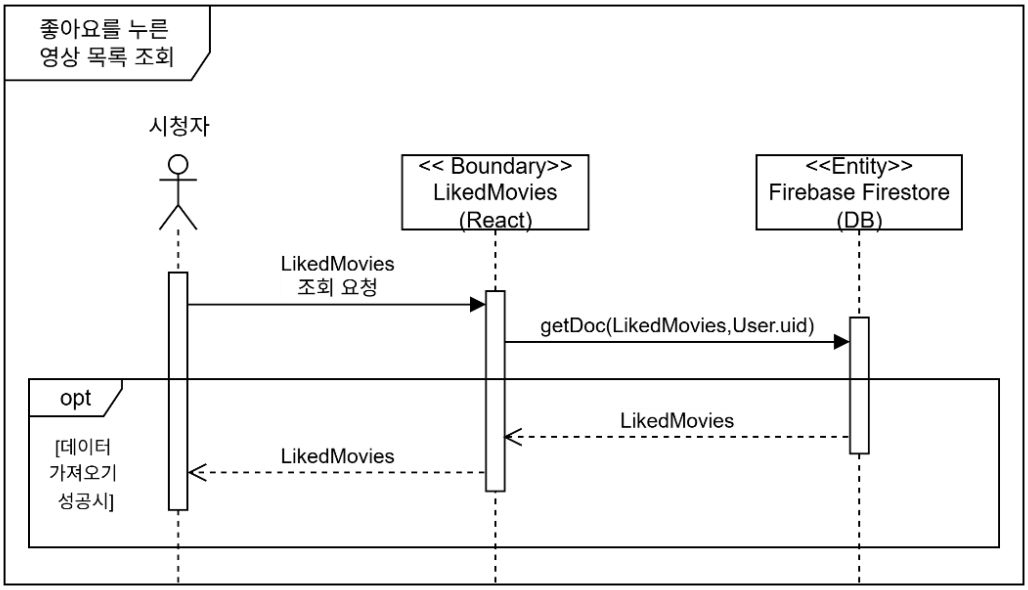


그림 47. 좋아요를 누른 영상 목록 조회 시퀀스 다이어그램

(시청 기록 조회 기능의 시퀀스 다이어그램 구조와 유사)

3.7.3.9. 좋아요를 누른 영상 목록 삭제 시퀀스 다이어그램

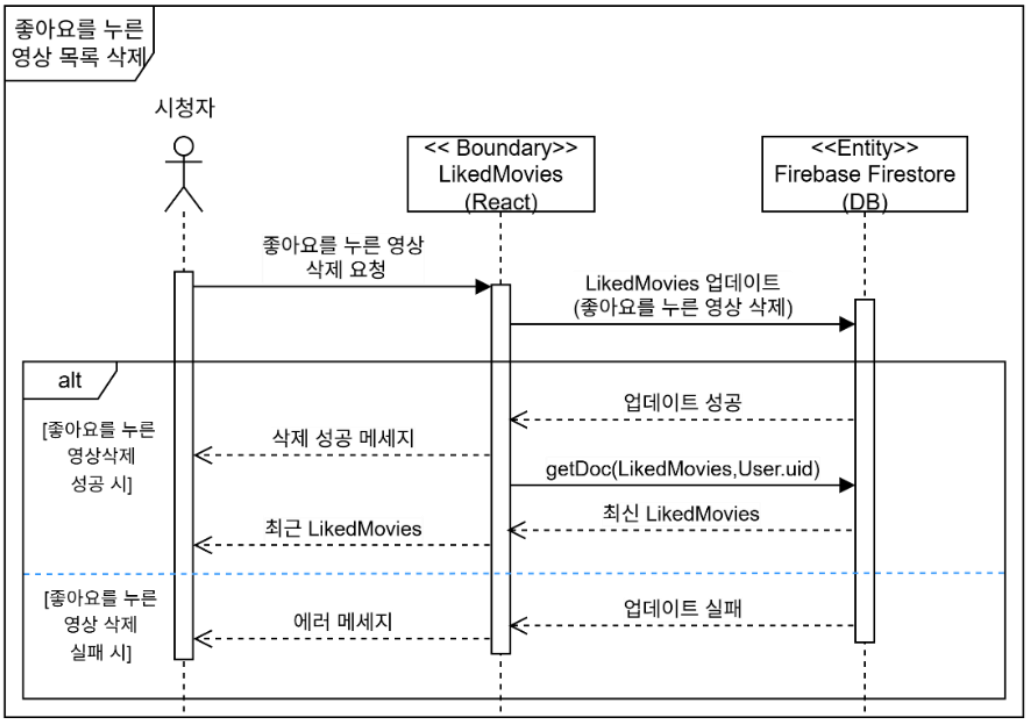


그림 48. 좋아요를 누른 영상 목록 삭제 시퀀스 다이어그램

(시청 기록 삭제 기능의 시퀀스 다이어그램 구조와 유사)

3.7.3.10. 나중에 볼 영상 목록 조회 시퀀스 다이어그램

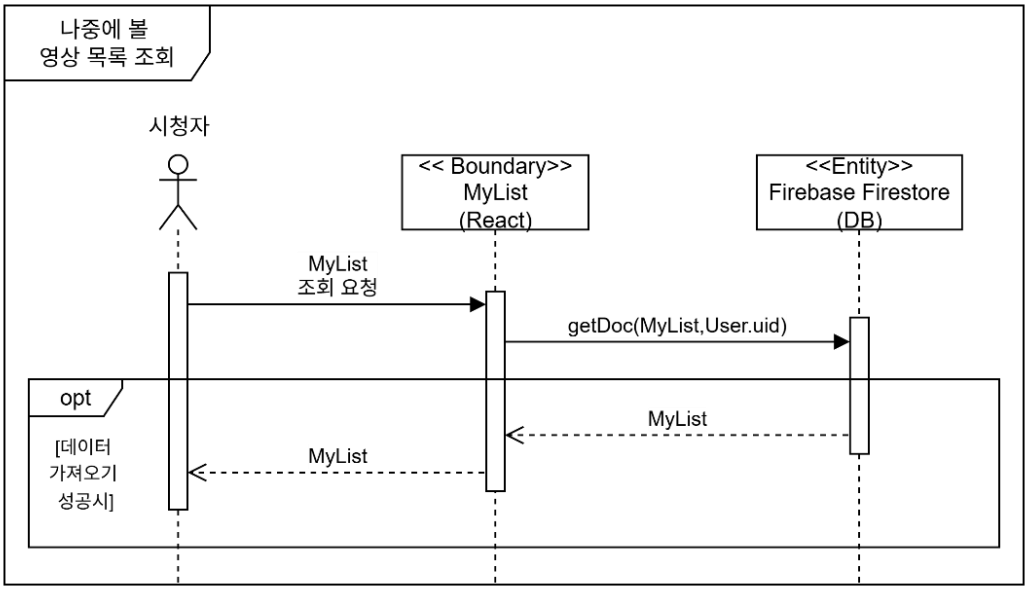


그림 49. 나중에 볼 영상 목록 조회 시퀀스 다이어그램

(시청 기록 조회 기능의 시퀀스 다이어그램 구조와 유사)

3.7.3.11. 나중에 볼 영상 목록 삭제 시퀀스 다이어그램

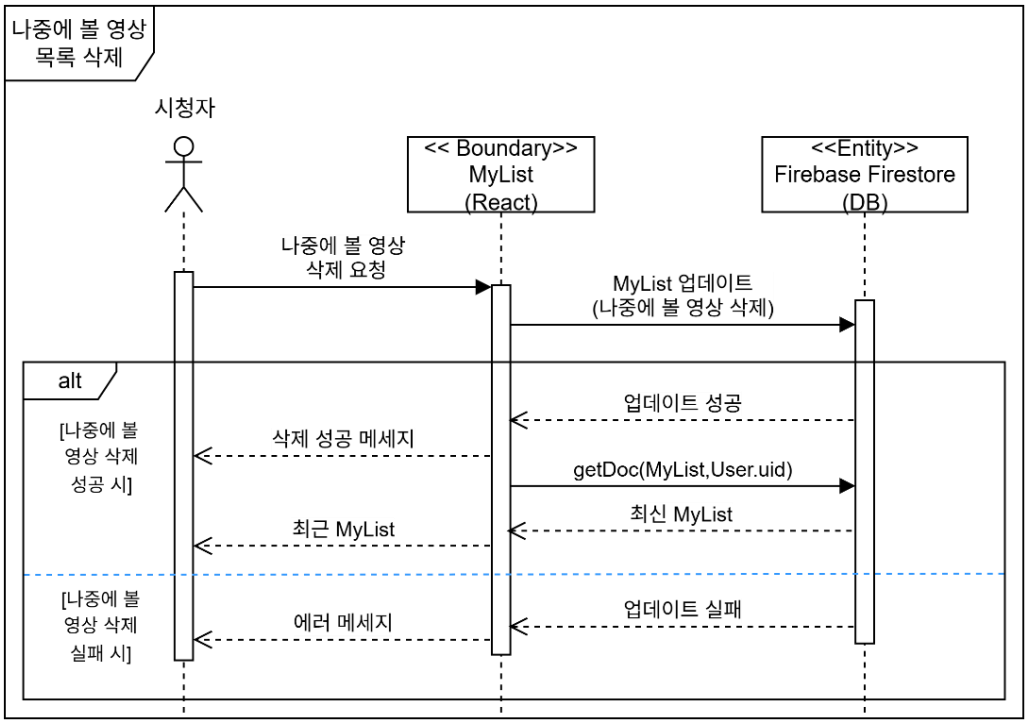


그림 50. 나중에 볼 영상 목록 삭제 시퀀스 다이어그램

(시청 기록 삭제 기능의 시퀀스 다이어그램 구조와 유사)

3.7.3.12. 적응형 스트리밍

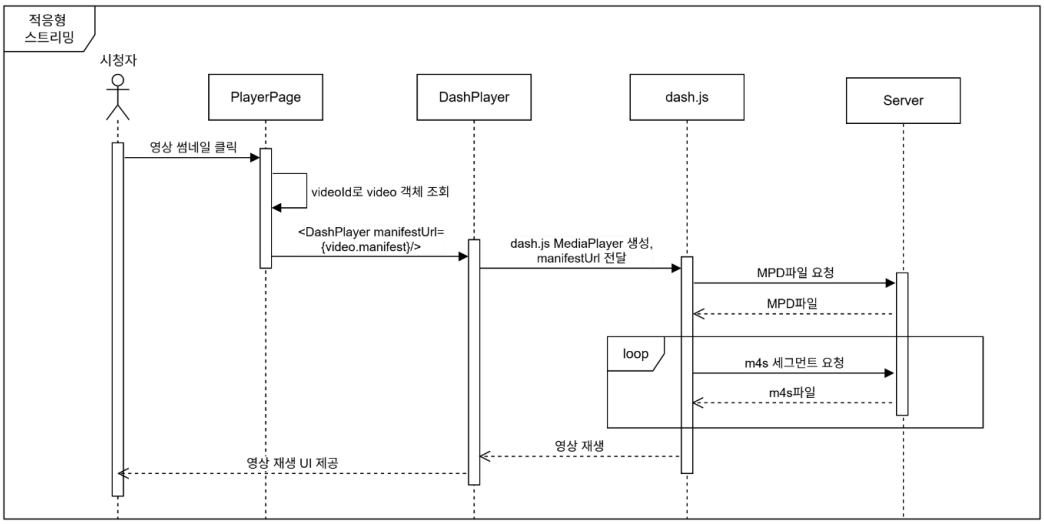


그림 51. 적응형 스트리밍 시퀀스 다이어그램

1. 스트리밍을 원하는 영상의 썸네일 클릭

2. PlayerPage

- URL 파라미터에서 videoId 추출
- 해당 videoId에 맞는 video객체 찾기
- 해당 video 객체의 manifest 찾아서 DashPlayer에 전달

3. DashPlayer

- dash.js MediaPlayer 생성
- manifestUrl 전달
- ABR 자동 품질 조절 활성화

4. dash.js

- MPD 파일 요청
- m4s 세그먼트 반복 요청

5. 영상 재생

3.7.4. dash.js 기반 적응형 스트리밍 구현

3.7.4.1. PlayerPage에서 DashPlayer 호출

사용자가 스트리밍 할 영상을 선택하면 해당 영상의 manifestUrl을 DashPlayer 컴포넌트에 아래와 같이 전달한다. 이 때 manifestUrl은 서버가 제공하는 MPD 파일 경로다.

```
<DashPlayer manifestUrl = {video.manifest} />
```

3.7.4.2. 영상 목록 관리 (videos.js)

videos.js에서 클라이언트가 스트리밍 할 수 있는 영상 목록을 아래와 같이 정의하고 각 항목은 id, title, manifest, thumb 속성을 가진다.

```
//vidoes.js
export default [
  {
    id: 1,
    title: '영상 1',
    // 서버에서 제공하는 MPD manifest 경로
    manifest: '/stream/husky/manifest.mpd',
    thumb: '/assets/thumbs/video1.png', // 썸네일 이미지 경로
  },
  {
    id: 2,
    title: '영상 2',
    manifest: '/stream/Sports/manifest.mpd',
    thumb: '/assets/thumbs/video2.png',
  },
  {
    id: 3,
    title: '영상 3',
    manifest: '/stream/news_v2/manifest.mpd',
    thumb: '/assets/thumbs/video3.png',
  }
]
```

3.7.4.3. DashPlayer 구현

DashPlayer에서 dash.js MediaPlayer 객체를 생성하고 초기화한다.

- updateSetting : ABR(Adaptive Bitrate) 기능을 활성화해 네트워크 상태에 따라 자동으로 화질 조절
- initialize : 전달받은 manifestUrl을 기반으로 스트리밍을 시작하고 <video> 태그에 영상 출력

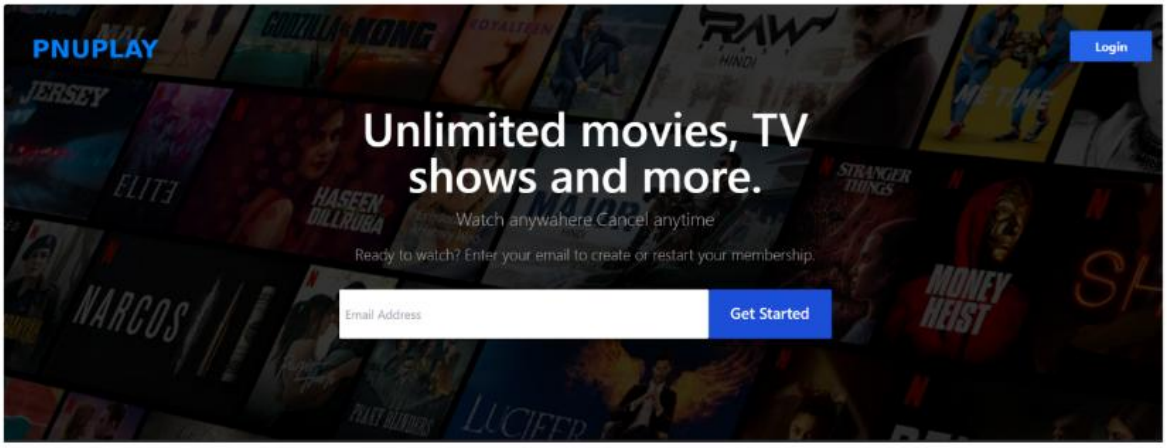
```
const player = dashjs.MediaPlayer().create();
playerRef.current = player;
player.updateSettings({
    streaming: {
        abr: {
            autoSwitchBitrate: {
                video: true,
                // audio: true
            }
        }
    }
});
player.initialize(videoRef.current, manifestUrl, true);
```

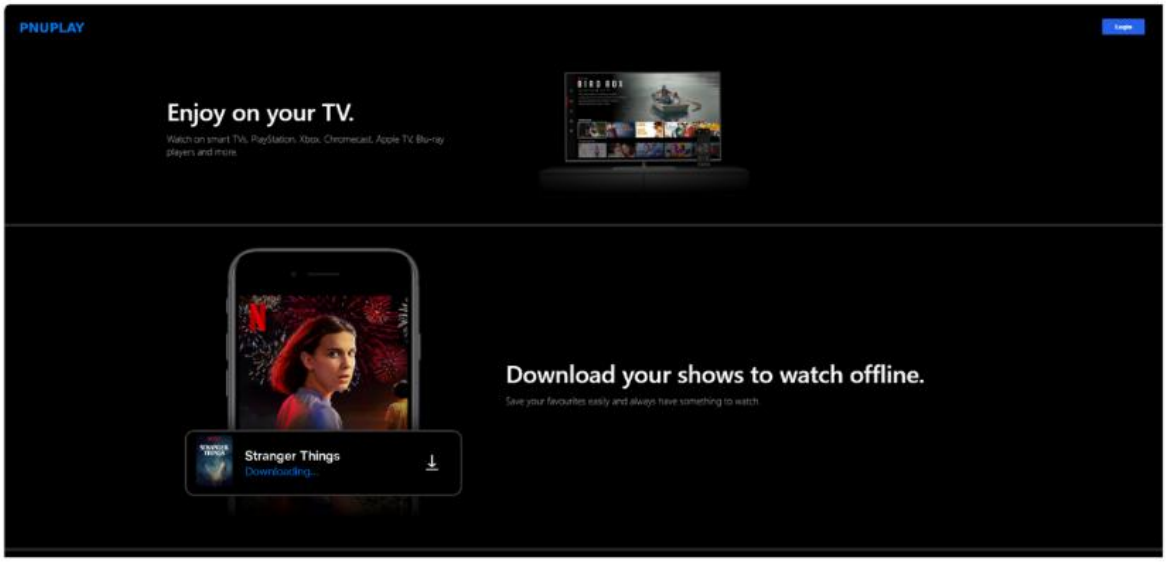
3.7.5. 웹 서비스 구현 화면

GitHub에 공개된 넷플릭스 클론 UI를 기반으로 전체적인 화면 구성을 먼저 구현한 뒤, UI 디자인을 프로젝트에 맞게 수정하였다. 또한 dash.js를 활용하여 서버에서 받은 MPD 파일을 기반으로 클라이언트에서 적응형 스트리밍을 수행할 수 있도록 구현하였다. 이를 통해 사용자는 영상 목록(썸네일)에서 원하는 콘텐츠를 선택하면 스트리밍이 가능하며, 네트워크 상태에 따라 자동으로 화질이 조정되는 DASH 기반 적응형 스트리밍 기능을 경험할 수 있다.

3.7.5.1. 서비스 시작 화면

[그림 54]는 사용자가 웹 서비스를 처음 접속했을 때 나타나는 초기 페이지로, 서비스의 핵심 콘텐츠를 소개하면서 로그인 또는 회원가입을 유도한다. 상단의 Login 버튼을 통해 로그인 할 수 있고 메인 영역의 Get Started 버튼을 통해 회원가입 창으로 이동할 수 있다.


PNUPLAY
Login
Unlimited movies, TV shows and more.
Watch anywhere Cancel anytime
Ready to watch? Enter your email to create or restart your membership.
Email Address
Get Started


PNUPLAY
Enjoy on your TV.
Watch on smart TVs, PlayStation, Xbox, Chromecast, Apple TV, Blu-ray players and more.
Stranger Things
Downloading...
Download your shows to watch offline.
Save your favourites easily and always have something to watch.

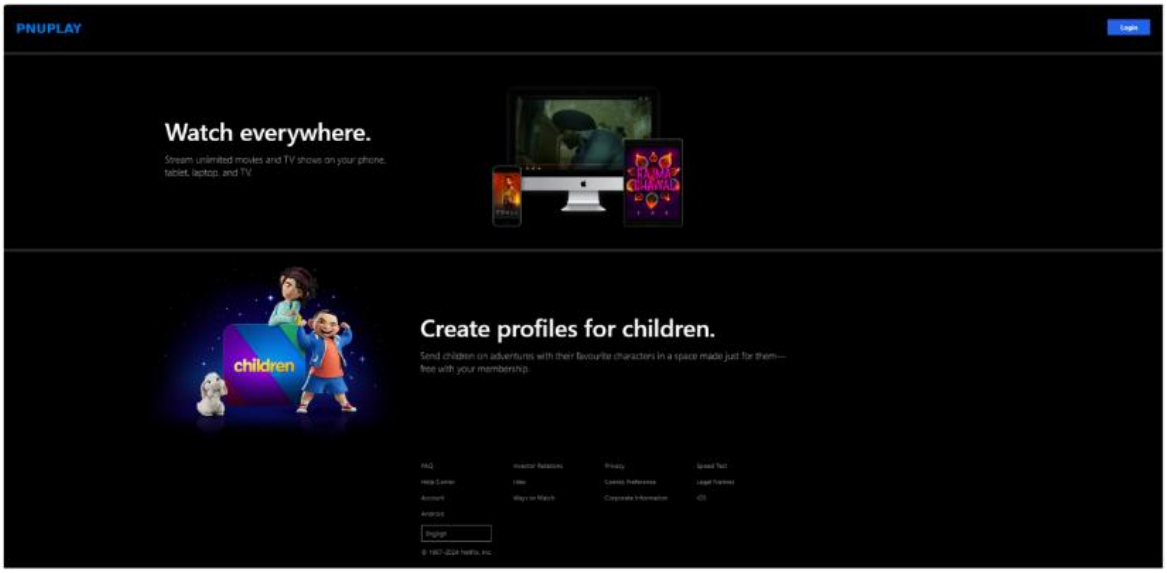

PNUPLAY
Watch everywhere.
Stream unlimited movies and TV shows on your phone, tablet, laptop, and TV.
children
Create profiles for children.
Send children on adventures with their favourite characters in a space made just for them—free with your membership.

그림 52. 서비스 시작 화면

3.7.5.2. 회원가입 화면

[그림 55]는 [그림 54]에서 Get Started 버튼을 클릭하면 나타나는 회원가입 화면으로, 사용자는 이메일과 비밀번호를 입력해서 계정을 생성할 수 있다. 이 때 이메일은 '@' 문자를 포함한 올바른 이메일 형식이어야 하고, 비밀번호는 최소 6자 이상 입력해야 정상적으로 계정 생성이 가능하다.

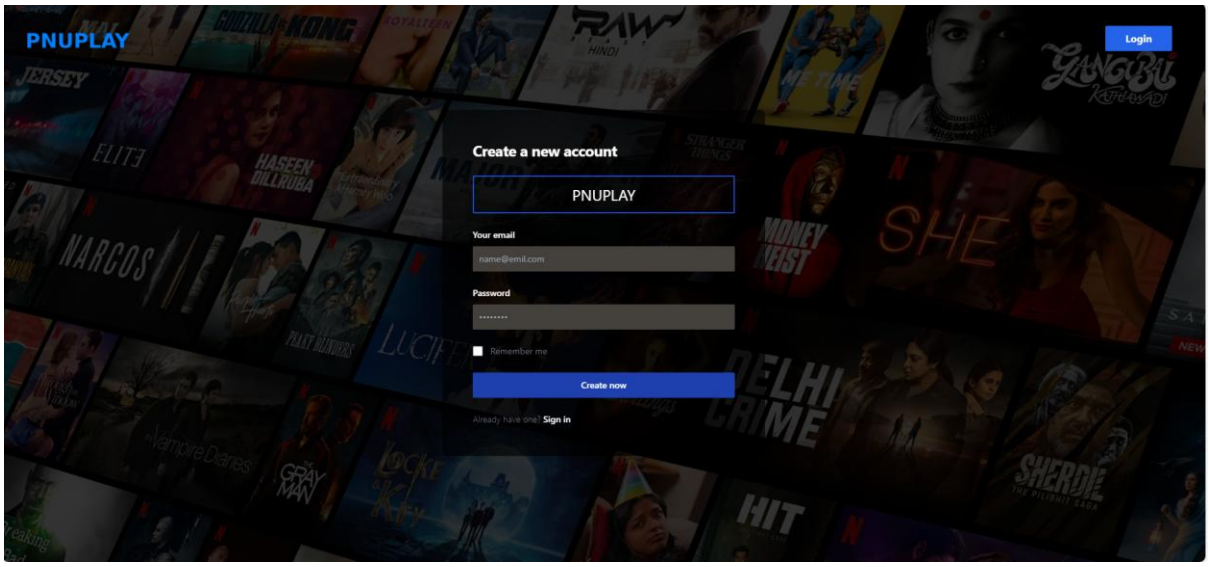


그림 53. 회원가입 화면

3.7.5.3. 로그인 화면

[그림 54]와 [그림 55]의 Login 버튼이나 [그림 55]의 Sign in을 클릭하면, [그림 56]에서 보이는 로그인 화면이 나타난다. 사용자는 [그림 56]에서 이메일과 비밀번호를 입력하여 로그인 할 수 있고, Sign in with Google 버튼을 클릭하여 실제로 가지고 있는 구글 계정을 통해 로그인할 수 있다. [그림 57]는 Sign in with Google 버튼을 클릭하였을 때 구글 로그인을 진행하는 화면이다. 또한 [그림 55]와 [그림 56]에 있는 Remember me 옵션을 선택하면 이후 접속 시 로그인 정보를 유지하여 자동으로 로그인 된다.

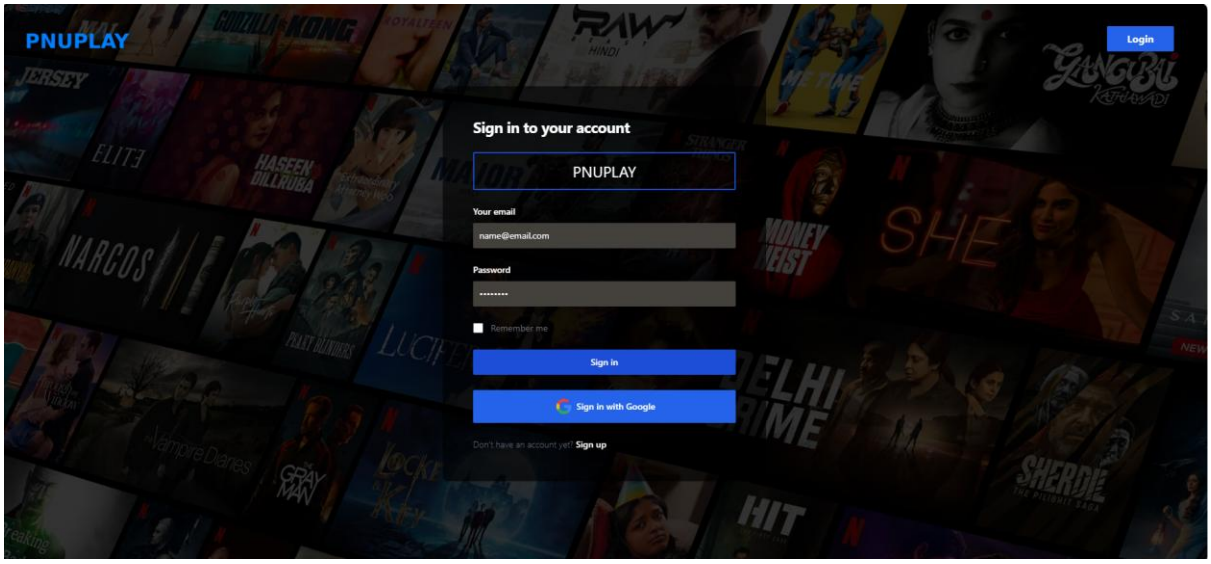


그림 54. 로그인 화면

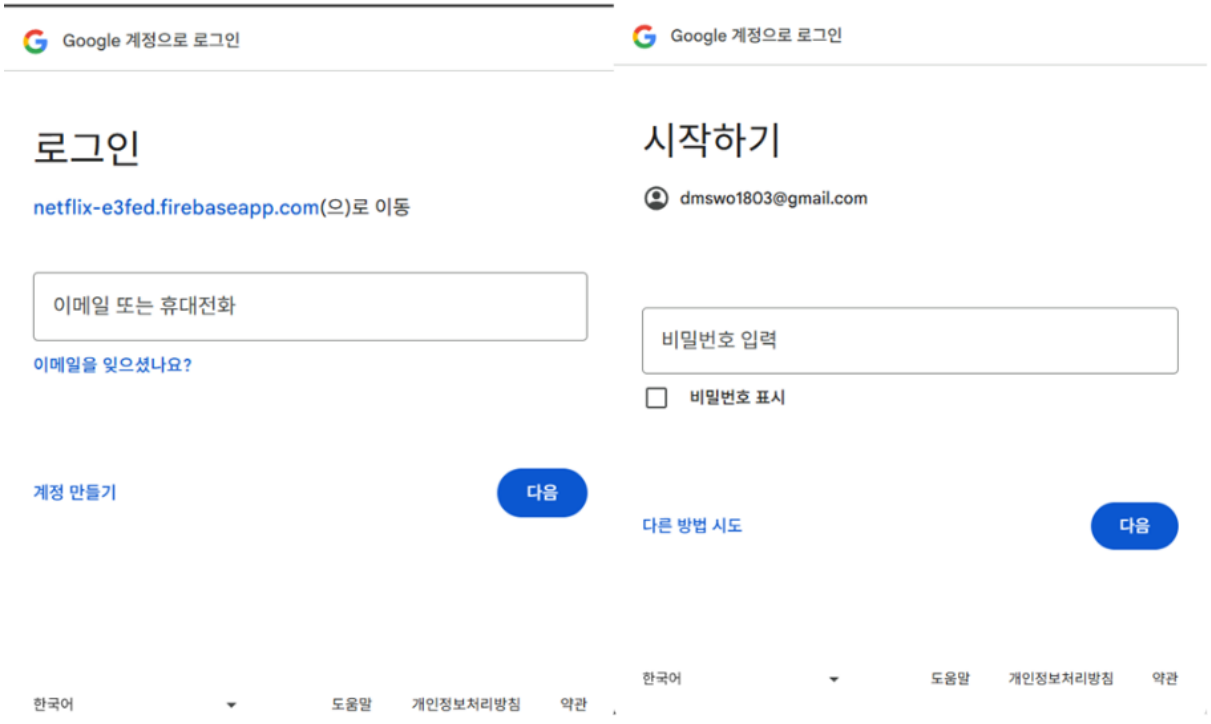


그림 55. 구글 로그인 화면

3.7.5.4. 프로필 설정 화면

[그림 58]은 이름과 프로필 이미지를 변경할 수 있는 프로필 설정 화면이다. Who is Watching? 밑에 있는 4개의 기본 이미지 중 하나를 클릭하는 즉시 프로필 사진이 클릭된 사진으로 변경된다. [그림 58]은 변경 전의 사진이고 [그림 59]에서 다른 이미지를 클릭해서 바뀐 것을 볼 수 있다. 그리고 다른 이름을 입력하자 Back to Home 버튼이 Save

and continue로 바뀌고 이 버튼을 클릭하면 정보가 변경되고 홈화면으로 이동하게 된다.

그리고 [그림 58]의 SignOut 버튼을 통해 로그아웃 할 수 있다.

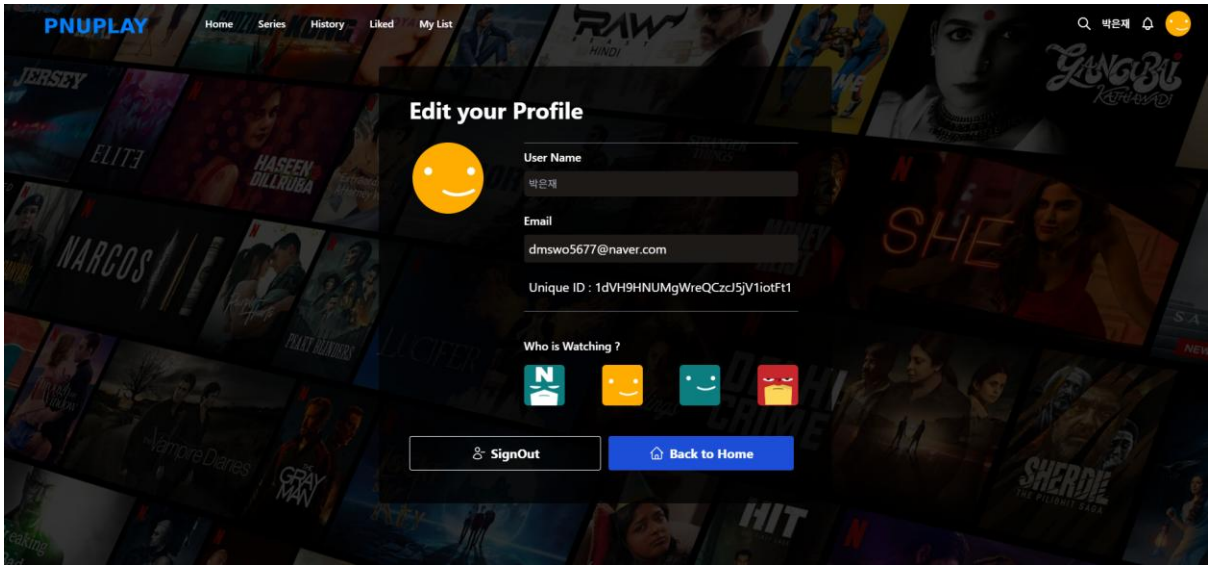


그림 56. 프로필 설정 화면

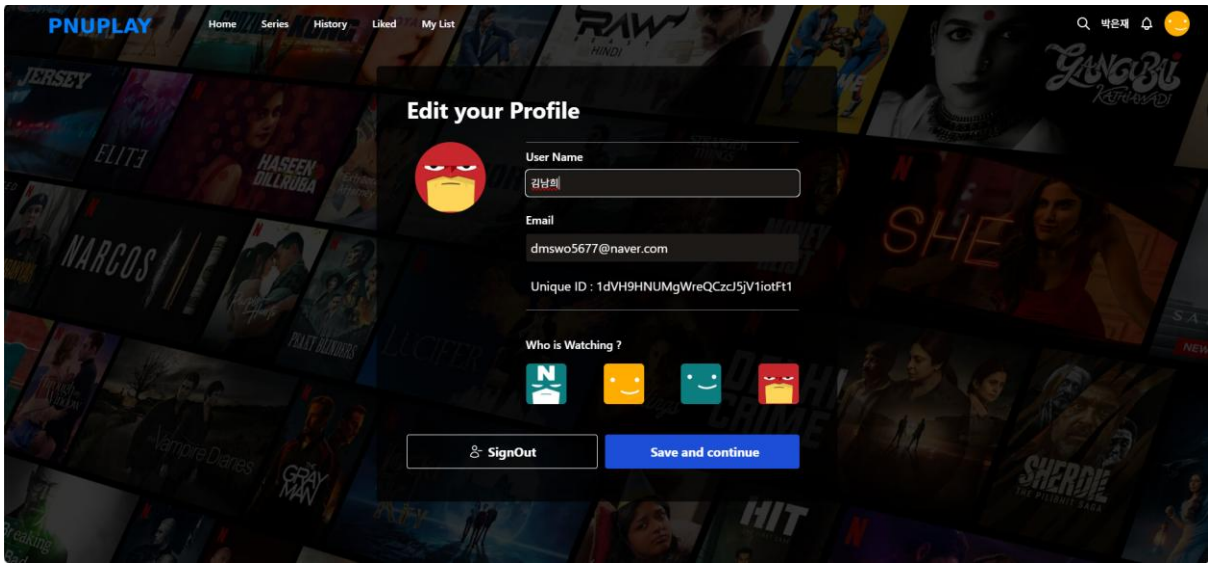


그림 57. 프로필 설정 화면

3.7.5.5. 홈 화면

[그림 60]은 홈 화면 상단 부분으로 대표 추천 영상이 뜨고, 접속할 때마다 다른 영상이 자동으로 표시된다. 그리고 오른쪽 상단에 사용자 프로필이 표시되며, 프로필 이미지에 마우스를 올리면 나타나는 Sign Out 버튼을 통해 로그아웃 할 수 있다.

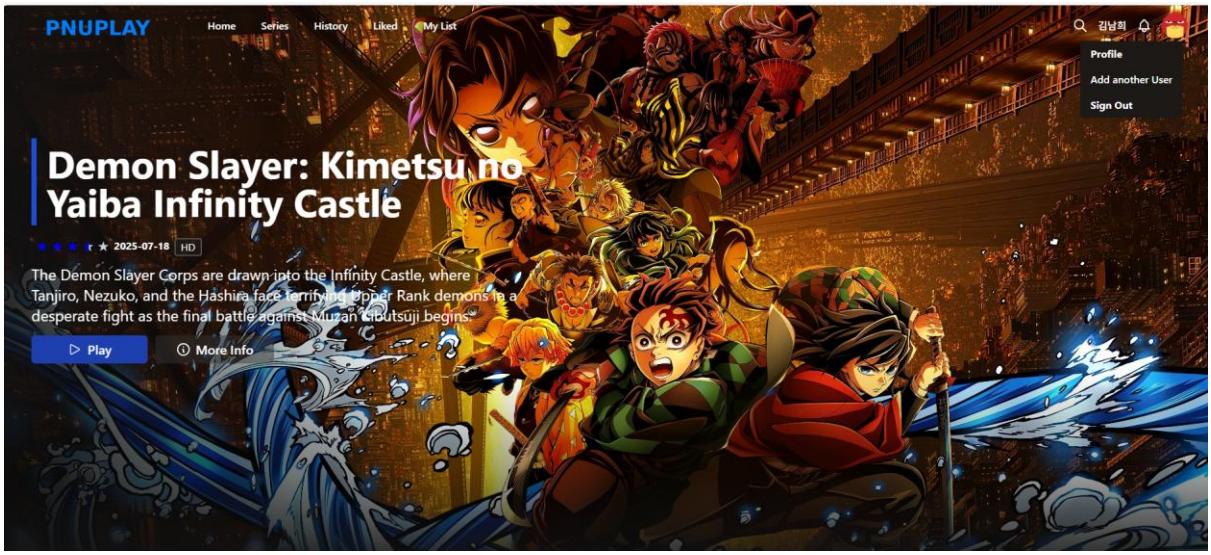


그림 58. 홈 화면

[그림 61]은 홈 화면 중앙 부분으로 장르/종류별로 영상이 나열된다.

특정 영상 위에 마우스를 올리면, 해당 영상의 정보와 네 개의 버튼이 나타난다. 순서대로 영상 재생, 나중에 볼 목록에 추가, 좋아요 누르기, 자세히 보기 기능을 한다.

첫번째 버튼을 클릭하면 [그림 62]처럼 해당 영상을 유튜브에 있는 짧은 영상으로 재생할 수 있다. 마지막 버튼을 클릭하면 [그림 63]처럼 상세 페이지가 뜬다.

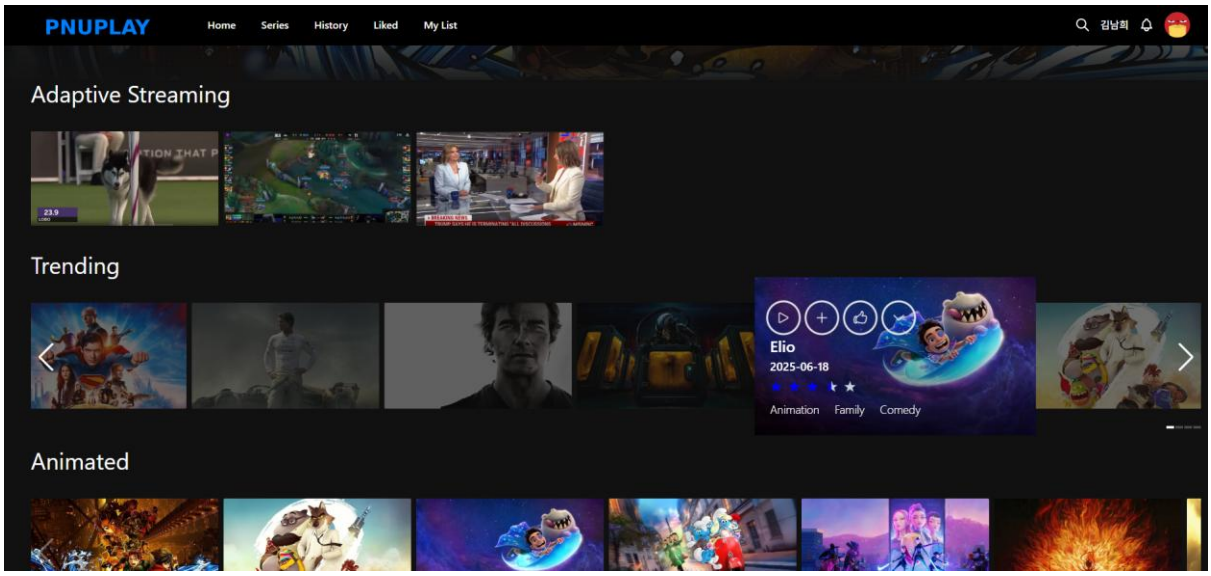


그림 59. 홈 화면

그림 60. 홈 화면

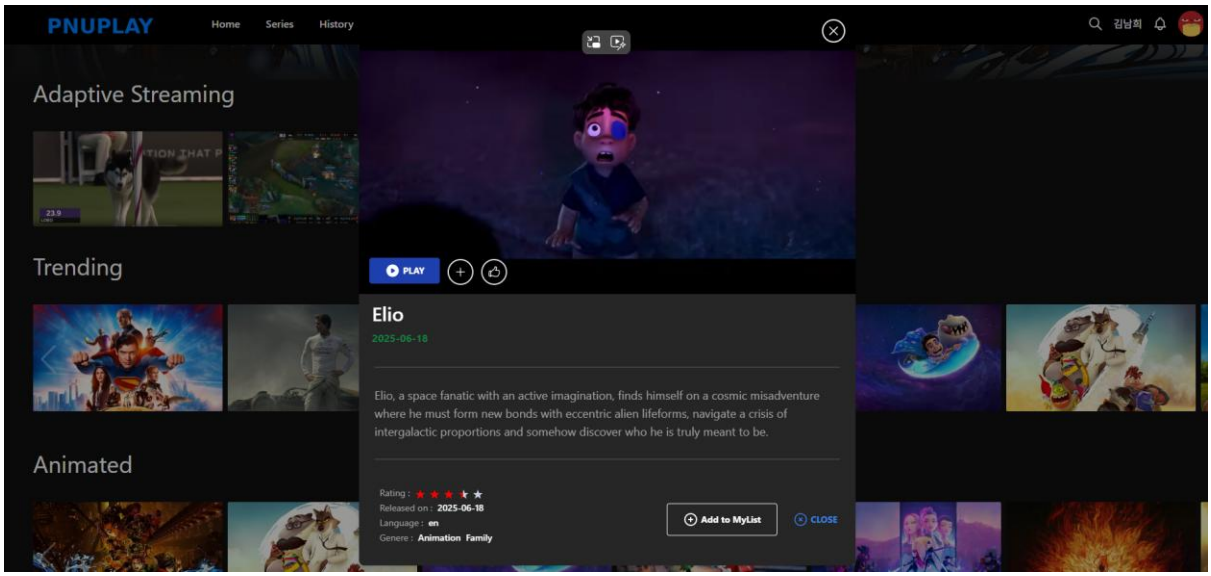


그림 61. 홈 화면

3.7.5.6. 적응형 스트리밍 화면

[그림 64]에서 Adaptive Streaming 영역은 적응형 스트리밍을 위해 별도로 추가하였다. 이 영역에서 직접 준비한 세 개의 영상 중 하나를 선택하면 [그림 65, 66]의 재생 페이지로 이동한다. 재생 페이지에서는 dash.js가 MPD를 파싱 해 네트워크 상태에 따라 적절한 품질의 세그먼트를 선택해서 재생한다. 그리고 [그림 61]에서 해상도 전환 뿐 아니라 비트레이트가 초단위로 변화하는 것을 볼 수 있다.

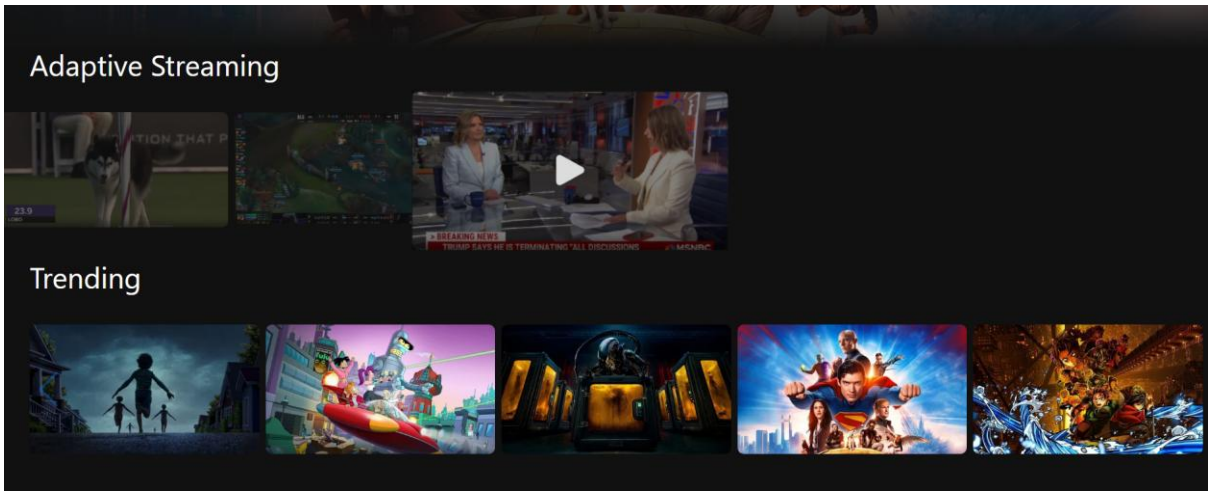


그림 62. 적응형 스트리밍 화면

그림 63. 적응형 스트리밍 화면

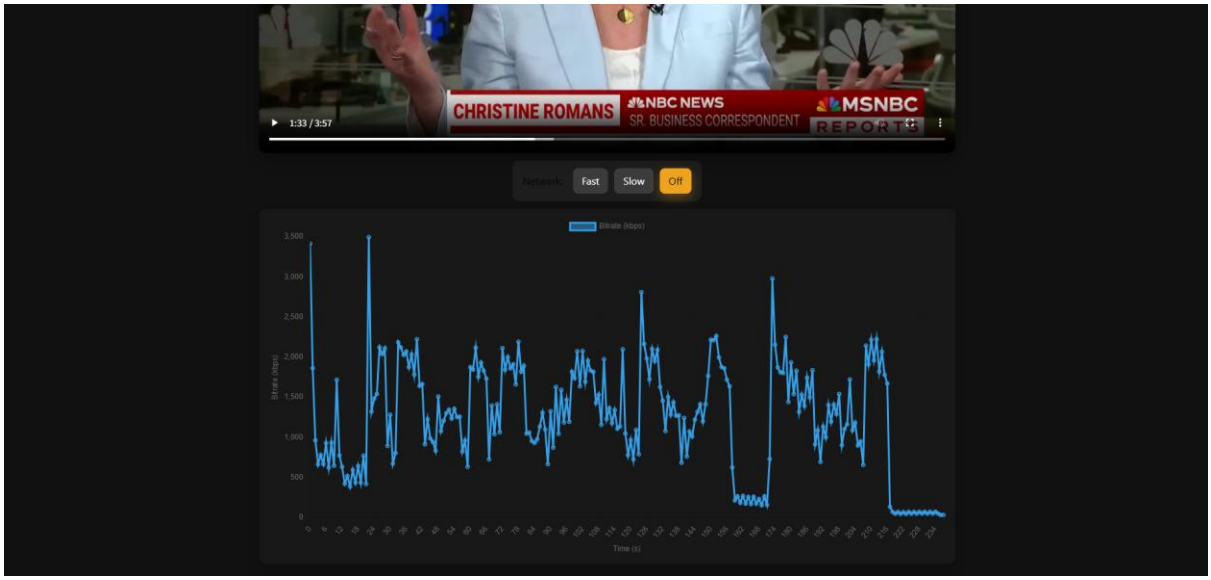


그림 64. 적응형 스트리밍 화면

3.7.5.7. 시청 기록 조회, 삭제 화면

[그림 67, 68]에서 시청한 영상 목록을 조회할 수 있다.

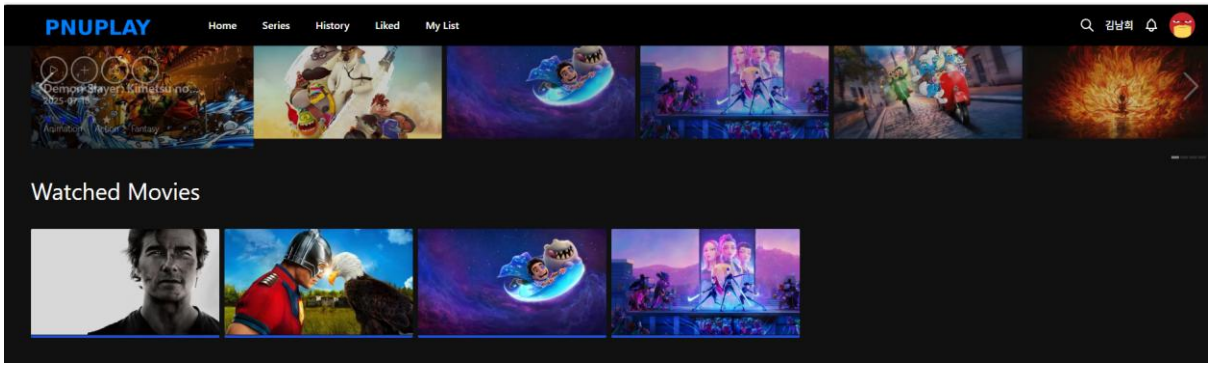


그림 65. 시청 기록 조회 화면

그리고 상단 메뉴의 History를 클릭했을 때 나타나는 [그림 68]에서 볼 수 있듯이 특정 영상에 마우스를 가져다 대면 버튼 여러 개가 나오는데, 그 중 세 번째 '삭제' 버튼을 클릭하면 해당 시청 기록을 삭제할 수 있다.

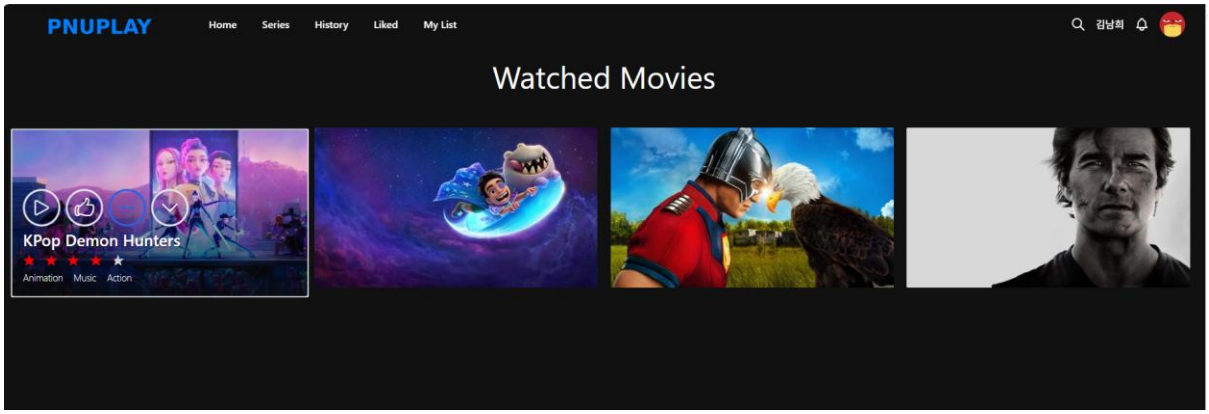


그림 66. 시청 기록 삭제 화면

3.7.5.8. 좋아요를 누른 영상 조회, 삭제 화면

[그림 63, 68] 등에서 볼 수 있는 좋아요 버튼을 클릭하면 해당 영상을 상단 메뉴의 Liked를 클릭했을 때 나타나는 화면에서 볼 수 있다.

[그림 69]에서 싫어요 버튼을 클릭하면 좋아요를 누른 영상 목록에서 해당 영상을 삭제할 수 있다.

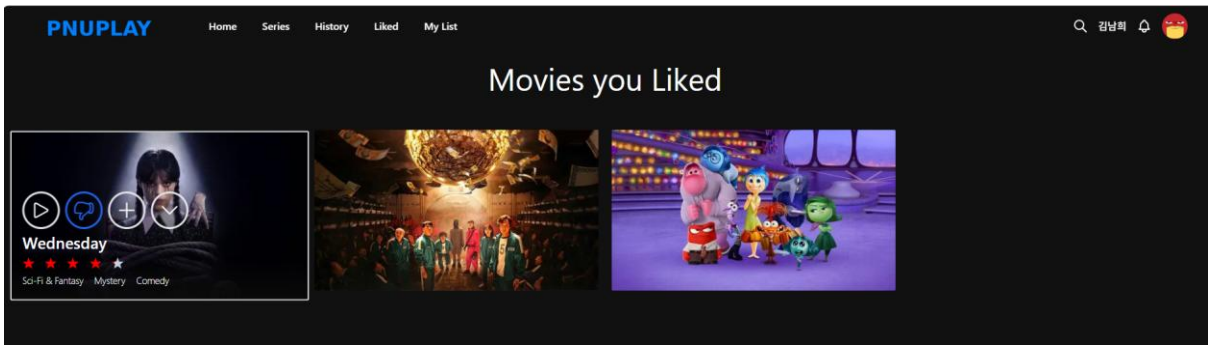


그림 67. 좋아요를 누른 영상 목록 화면

3.7.5.9. 나중에 볼 영상 조회, 삭제 화면

[그림 61, 63] 등에서 볼 수 있는 플러스 버튼을 클릭하면 해당 영상을 상단 메뉴의 My List를 클릭했을 때 나타나는 화면에서 볼 수 있다.

[그림 70]에서 마이너스 버튼을 클릭하면 나중에 볼 영상 목록에서 해당 영상을 삭제할 수 있다.

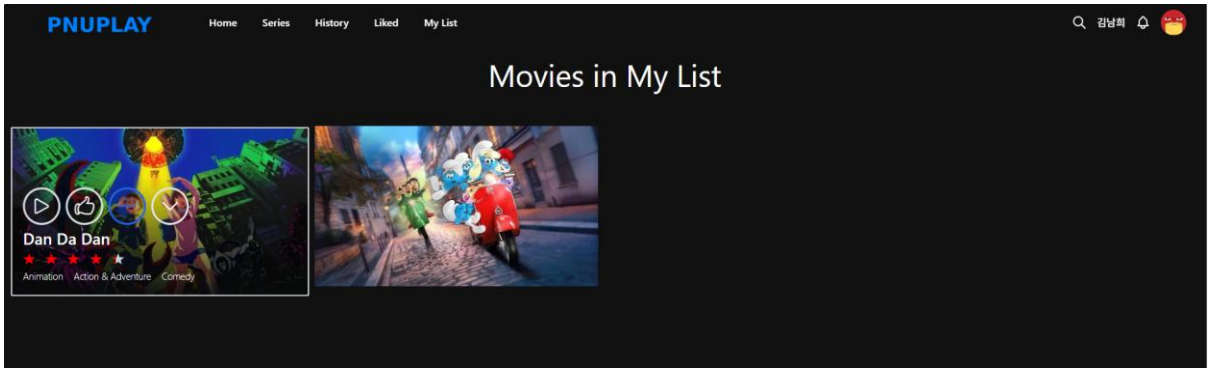


그림 68. 나중에 볼 영상 목록 화면

[그림 71]처럼 영상 제목으로 검색해 원하는 영상을 찾을 수 있다.

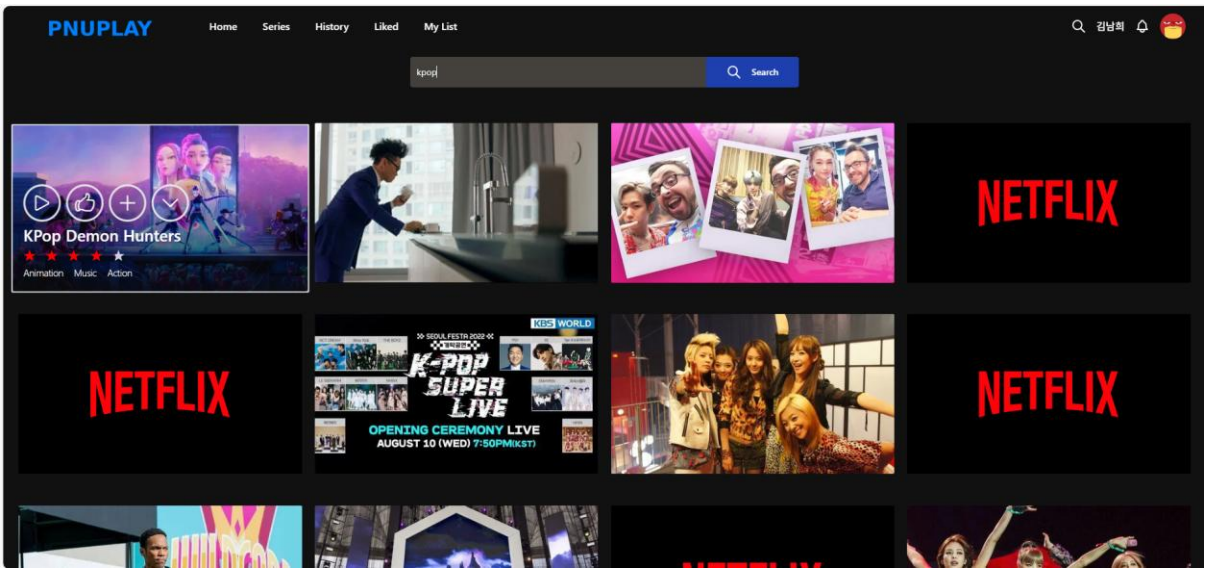


그림 69. 영상 검색 화면

3.8. 배포

본 시스템은 AWS EC2(Ubuntu 22.04) 환경에서 Docker Compose를 이용해 배포하였다. 서버는 FastAPI 백엔드와 nginx 리버스 프록시로 구성되며, React 클라이언트는 빌드 후 nginx를 통해 정적 페이지 형태로 제공된다. 전체 서비스는 컨테이너 단위로 실행되어 관리가 쉬우며, 사용자는 https://pnuplay.duckdns.org/ 주소를 통해 시스템을 이용할 수 있다.

4. 연구 결과 분석 및 평가

4.1. 영상 특성에 따른 스트리밍 동작

dash.js 기반 스트리밍 중, 클라이언트가 수신한 영상 세그먼트들의 초 단위 비트레이트를 측정해 전체 재생 시간 동안의 비트레이트 변화를 그래프로 시각화한 것을 볼 수 있다.

- 고모션, 고복잡도 영상

  고모션, 고복잡도 영상의 경우 시각적 마스킹 효과로 인해 CRF가 전체적으로 높게 예측되었으며, 원본 대비 비트레이트 절감율이 높다.

```
[husky]=== 영상 특성 추출 및 인코딩 파라미터 최적화 ===
[segment_0 원본: 1280x720] CRF=35, maxrate=8291k
[segment_0 640x360] CRF=34, maxrate=2073k
[segment_0 854x480] CRF=34, maxrate=3688k
[segment_0 1280x720] CRF=35, maxrate=8291k
[segment_0 1920x1080] CRF=35, maxrate=18655k
[segment_1 원본: 1280x720] CRF=35, maxrate=8287k
[segment_1 640x360] CRF=34, maxrate=2072k
[segment_1 854x480] CRF=34, maxrate=3686k
[segment_1 1280x720] CRF=35, maxrate=8287k
[segment_1 1920x1080] CRF=35, maxrate=18646k
```

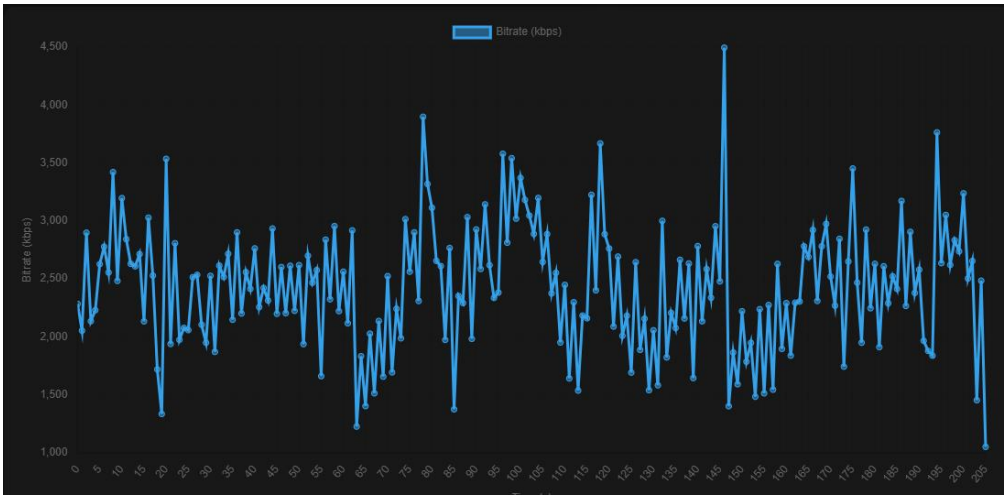


그림 70. 고모션, 고복잡도 영상 인코딩 비트레이트

- 저모션, 저복잡도 영상

  저모션, 저복잡도 영상의 경우 기본적으로 비트레이트가 낮게 나오지만, 시각적 마스킹 효과가 거의 적용되지 않아 CRF가 전체적으로 낮게 나오며 원본과 비슷한 수준을 유지한다.

```
[news]=== 영상 특성 추출 및 인코딩 파라미터 최적화 ===
[segment_0 원본: 1280x720] CRF=32, maxrate=6217k
[segment_0 640x360] CRF=31, maxrate=1554k
[segment_0 854x480] CRF=31, maxrate=2765k
[segment_0 1280x720] CRF=32, maxrate=6217k
[segment_0 1920x1080] CRF=32, maxrate=13988k
[segment_1 원본: 1280x720] CRF=32, maxrate=6230k
[segment_1 640x360] CRF=31, maxrate=1558k
[segment_1 854x480] CRF=31, maxrate=2771k
[segment_1 1280x720] CRF=32, maxrate=6230k
[segment_1 1920x1080] CRF=32, maxrate=14018k
```

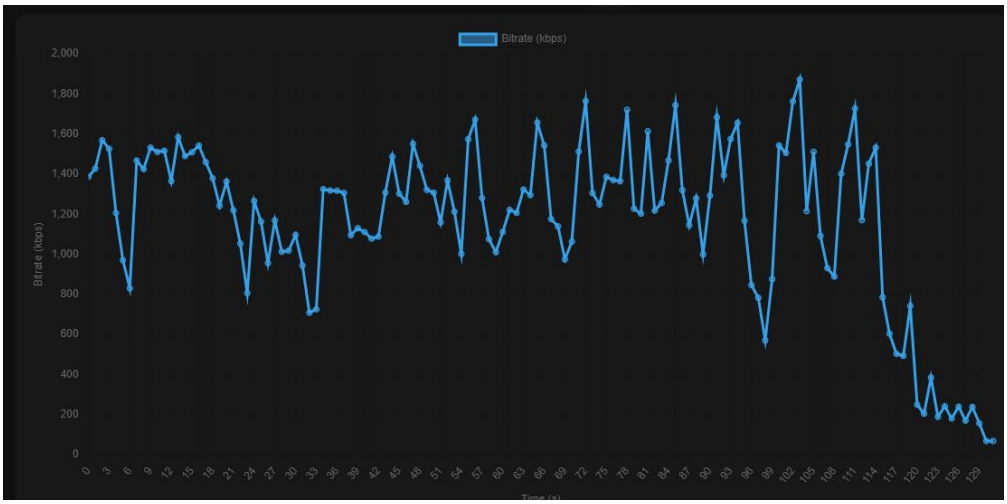


그림 71. 저모션, 저복잡도 영상 인코딩 비트레이트

4.2. 네트워크 상태 변경에 따른 스트리밍 동작

[그림 75]는 사용자가 영상을 시청하는 도중 버튼을 통해 네트워크 상태를 인위적으로 변경했을 때, 클라이언트가 자동으로 적절한 해상도의 세그먼트를 선택하는 과정을 보여준다. [그림 76]처럼 fast, slow, off 세 가지 네트워크 프로파일을 사전에 설정하여 테스트를 진행했다. 처음에는 off 상태(무제한 속도)로 시작하여 dash.js가 고화질 세그먼트인

1080p를 선택하였다. 이후 네트워크를 Slow로 변경하자, dash.js는 속도 저하를 감지하고 720p, 480p로 화질을 낮추었다. 마지막으로 Fast로 전환하니 다시 1080p 세그먼트를 선택하는 것을 확인할 수 있다. 이를 통해 dash.js가 MPEG-DASH의 적응형 스트리밍 구조에 따라, 네트워크 상황을 실시간으로 반영하여 최적의 화질을 선택한다는 것을 확인하였다.

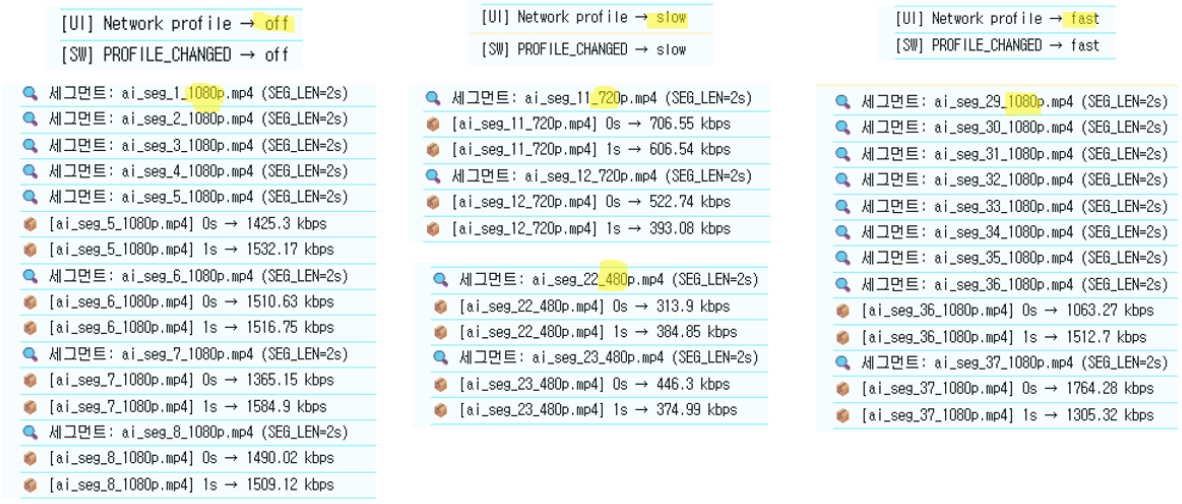


그림 72. 네트워크 상태에 따른 해상도 변화

```
const PROFILES = {
  off:  { bps: Infinity, latency: 0 },          // 제한 없음
  slow: { bps: 700_000, latency: 120 },        // 700kbps, 120ms RTT
  fast: { bps: 10_000_000, latency: 40 },       // 10Mbps, 40ms
};
```

그림 73. 네트워크 프로파일

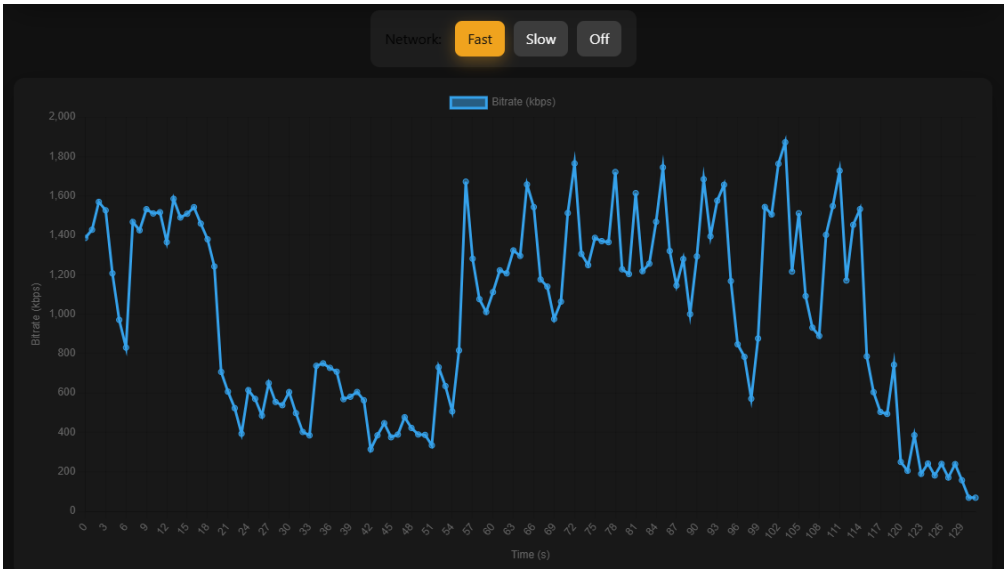


그림 74. 인코딩 비트레이트

5. 결론 및 향후 연구 방향

5.1. 결론

1. 영상 특성 기반 최적화 : 세그먼트별 영상 특성을 분석하여 최적의 인코딩 파라미터(CRF, Maxrate)를 적용함으로써, 기존 방식 대비 비트레이트를 줄이면서도 체감 화질 저하가 발생하지 않도록 하였다.
또한 생성되는 세그먼트 수를 줄여 서버 저장 공간과 인코딩 시간을 절약하여, 동시 접속자가 많아져도 서버 트래픽 부담을 최소화할 수 있음을 확인하였다.

2. 적응형 네트워크 : 네트워크 환경이 갑작스럽게 불안정해지더라도 dash.js의 적응형 스트리밍 기능을 통해 자동으로 해상도를 낮추어 선택함으로써, 끊김 없는 안정적인 스트리밍을 제공한다.

3. 웹 서비스: 이러한 기능을 통합하여 PNUPLAY 웹 서비스로 구현하였고, 이를 통해 전체 시스템의 동작을 확인할 수 있다.

5.2. 향후 연구 방향

1. 현재는 모션 벡터, 매크로블록, 공간 복잡도 등 기본적인 영상 특성을 활용했으나, 향

후에는 오디오 정보, 자막·텍스트 메타데이터, 시청 로그와 같은 멀티모달[1] 데이터를 함께 고려하여 더욱 정교한 인코딩 최적화를 시도할 계획이다. 예를 들어, 대화 장면은 오디오 품질을 우선, 스포츠 장면은 영상 디테일을 우선하는 식으로 구현할 수 있다.

2. 클라이언트 측에서 QoE(Quality of Experience) 기반 지표(버퍼링 발생 시간, 시청자 만족도 등)를 반영하여, 단순한 비트레이트 최적화가 아니라 실제 사용자 경험을 고려한 적응형 스트리밍을 구현할 예정이다.

6. 구성원별 역할 및 개발 일정

학번	이름	구성원별 진척도
202155515	김남희	영상 특성 추출 훈련 데이터셋 생성 1. VMAF 품질 평가 2. 최적 인코딩 파라미터 탐색 인코딩 파라미터 최적화 모델 구현 및 훈련 최적화 모델 스트리밍 시스템 적용
202155553	박은재	영상 스트리밍 웹 서비스 1. DASH 기반 영상 스트리밍 구현 2. Firebase 기반 사용자 인증 및 데이터베이스 구축 인코딩 파이프라인 구현 AWS EC2 + Docker 기반 배포 환경 구축

[1] 멀티모달(Multi Modal): 텍스트, 이미지, 음성, 영상 등 다양한 형태의 정보를 함께 이해하고 처리하는 인공지능 기술

업무	5월				6월				7월				8월				9월				10월			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
스터디 및 착수 보고서 작성																								
ffmpeg 스트리밍 및 웹 어플리케이션 구현																								
영상 복잡도 분석 알고리즘 개발																								
네트워크 모니터링 및 인코딩 분석 데이터 수집																								
실시간 적응형 스트리밍 구현																								
개선 사항 분석 및 수정																								
최종 평가 및 배포																								
최종 보고서 및 발표 준비																								

그림 75. 개발 일정

7. 참고 문헌

[1] Velibor Adzic, Hari Kalva, Borko Furht, "Temporal visual masking for HEVC/H.265 perceptual optimization", 2013 Picture Coding Symposium (PCS), pp.430-433, 2013.

[2] V. Adzic, H. Kalva, and B. Furht, "Exploring visual temporal masking for video compression," Proc. 2013 IEEE International Conference on Consumer Electronics (ICCE), pp. 590-591, 2013

[3] S. Rimac-Drlje, D. Zagar, and G. Martinovic, "Spatial Masking and Perceived Video Quality in Multimedia Applications," Proc. 2009 16th International Conference on Systems, Signals and Image Processing, pp. 1-4, 2009.

[4] J. Vlaović, M. Vranješ, D. Grabić, and D. Samardžija, "Comparison of Objective Video Quality Assessment Methods on Videos With Different Spatial Resolutions," Proc. 2019 International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 287-292, 2019.

[5] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet", IEEE MultiMedia, vol. 18, no. 4, pp. 62-67, April 2011.