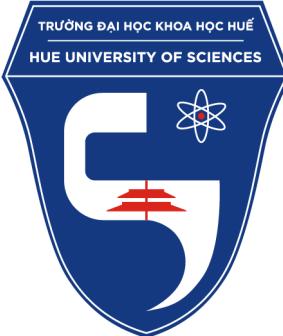


ĐẠI HỌC HUẾ  
TRƯỜNG ĐẠI HỌC KHOA HỌC  
KHOA CÔNG NGHỆ THÔNG TIN  
\*\*\*



**KHÓA LUẬN  
TỐT NGHIỆP ĐẠI HỌC**

Đề tài:

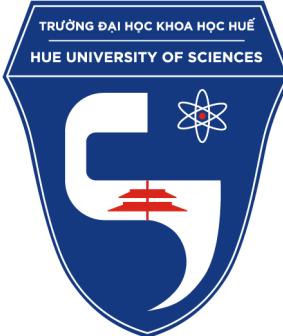
**TÌM HIỂU TIMESFORMER VÀ  
ỨNG DỤNG BÀI TOÁN TÓM TẮT VIDEO**

Sinh viên thực hiện: **NGUYỄN HOÀI NAM**

Khóa: **K44 - HỆ CHÍNH QUY**

Huế, 05 - 2024

ĐẠI HỌC HUẾ  
TRƯỜNG ĐẠI HỌC KHOA HỌC  
KHOA CÔNG NGHỆ THÔNG TIN  
\*\*\*



**KHÓA LUẬN  
TỐT NGHIỆP ĐẠI HỌC  
NGÀNH CÔNG NGHỆ THÔNG TIN**

Đề tài:

**TÌM HIỂU TIMESFORMER VÀ  
ỨNG DỤNG BÀI TOÁN TÓM TẮT VIDEO**

Sinh viên thực hiện: **NGUYỄN HOÀI NAM**

Khóa: **K44 - HỆ CHÍNH QUY**

Giáo viên hướng dẫn: **TS. LÊ QUANG CHIẾN**

Huế, 05 - 2024

# LỜI CẢM ƠN

Trong cuộc đời mỗi người, những năm tháng sinh viên luôn là một trong những giai đoạn đáng nhớ và ý nghĩa nhất. Qua bao nhiêu nỗ lực, thách thức và niềm vui, tôi đã hoàn thành khóa luận tốt nghiệp - một cột mốc quan trọng đánh dấu sự kết thúc của hành trình học tập tại trường Đại học Khoa học, Đại học Huế. Trước khi bước vào chặng đường mới, tôi xin được gửi lời cảm ơn chân thành và sâu sắc nhất đến tất cả những người đã đồng hành và hỗ trợ tôi trong suốt quá trình học tập và nghiên cứu.

Đầu tiên, tôi xin bày tỏ lòng biết ơn sâu sắc đến Ban Giám hiệu nhà trường. Các thầy cô đã tạo điều kiện thuận lợi nhất cho chúng tôi có môi trường học tập tốt, hiện đại và đầy đủ tiện nghi. Nhờ có sự lãnh đạo tận tâm và chiến lược phát triển đúng đắn của Ban Giám hiệu, tôi và các bạn sinh viên khác mới có cơ hội tiếp cận với nền giáo dục chất lượng, được trang bị đầy đủ kiến thức và kỹ năng cần thiết để bước vào cuộc sống và sự nghiệp tương lai.

Tiếp theo, tôi xin được gửi lời cảm ơn chân thành đến các thầy cô giáo trong khoa Công nghệ thông tin. Những bài giảng sâu sắc, những giờ học bổ ích và những lời khuyên chân thành của các thầy cô đã giúp tôi tích lũy được nhiều kiến thức quý báu. Không chỉ truyền đạt kiến thức, các thầy cô còn dạy chúng tôi cách suy nghĩ, cách học hỏi và cách làm người. Những giá trị này sẽ luôn là hành trang quý báu trong suốt cuộc đời tôi.

Đặc biệt, tôi xin gửi lời tri ân sâu sắc nhất đến giáo viên hướng dẫn của tôi, thầy giáo - TS. Lê Quang Chiến. Thầy không chỉ là người thầy mà còn là người bạn, người đồng hành tận tụy cùng tôi trong suốt quá trình thực hiện khóa luận. Những góp ý, nhận xét của thầy đã giúp tôi hoàn thiện và phát triển ý tưởng nghiên cứu của mình. Sự động viên, khích lệ của thầy mỗi khi tôi gặp khó khăn, ốm đau đã trở thành nguồn động lực to lớn giúp tôi vượt qua mọi thử thách. Tôi hiểu rằng, dù thành công hay thất bại, sự phát triển của tôi có một phần không nhỏ từ sự giúp đỡ và hỗ trợ của thầy.

Ngoài ra, tôi cũng muốn gửi lời cảm ơn đến các anh chị khoá trên, các

bạn sinh viên cùng khóa. Nhờ có sự hỗ trợ, chia sẻ và giúp đỡ của mọi người, tôi mới có thể hoàn thành tốt các bài tập, dự án và cuối cùng là khóa luận tốt nghiệp. Những buổi thảo luận nhóm, những giờ làm việc chung đã giúp tôi học hỏi được rất nhiều điều và rèn luyện kỹ năng làm việc nhóm, kỹ năng giao tiếp.

Bên cạnh đó, tôi cũng xin gửi lời cảm ơn đặc biệt đến các anh chị trong doanh nghiệp FPT Software Quy Nhơn (QAI). Thời gian thực tập tại QAI đã mang lại cho tôi rất nhiều kinh nghiệm thực tiễn quý báu. Các anh chị không chỉ hướng dẫn tôi về công việc chuyên môn mà còn chia sẻ những kinh nghiệm quý giá về môi trường làm việc thực tế, giúp tôi hiểu rõ hơn về cách áp dụng kiến thức đã học vào thực tế. Sự tận tình, cởi mở và chuyên nghiệp của các anh chị đã giúp tôi trưởng thành hơn rất nhiều. Những buổi làm việc cùng các anh chị đã mở ra cho tôi những góc nhìn mới, những kỹ năng mới và tạo động lực để tôi hoàn thiện bản thân mỗi ngày. Sự hỗ trợ của các anh chị đã đóng góp không nhỏ vào việc hoàn thành khóa luận này.

Không thể không nhắc đến gia đình, những người luôn đứng sau ủng hộ và động viên tôi. Ba mẹ đã dành cho tôi tình yêu thương vô bờ bến, luôn là chỗ dựa vững chắc và động lực để tôi cố gắng trong mọi hoàn cảnh. Sự quan tâm, lo lắng của ba mẹ đã giúp tôi có thêm niềm tin và sức mạnh để vượt qua mọi khó khăn trong suốt quá trình học tập.

Một lần nữa, tôi xin gửi lời cảm ơn chân thành và sâu sắc nhất đến Ban Giám hiệu, các thầy cô giáo và toàn thể các bạn sinh viên trường Đại học Khoa học, cũng như các anh chị trong doanh nghiệp QAI. Tôi xin hứa sẽ luôn cố gắng học tập, làm việc và sống tốt để không phụ lòng mong đợi và tình cảm mà mọi người đã dành cho tôi.

Bản thân tôi đã cố gắng hết sức trong quá trình thực hiện đề tài này nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Kính mong quý thầy cô góp ý, chỉ dẫn.

Xin chân thành cảm ơn!

# DANH MỤC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

Từ viết tắt	Điễn giải	Dịch nghĩa
AI	Artificial intelligence	Trí tuệ nhân tạo
VS	Video Summarization	Tóm tắt video
R,G,B	Red, Green, Blue	Đỏ, xanh lá, xanh dương
CNN	Convolutional Neural Networks	Mạng nơ-ron tích chập
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
GPU	Graphics Processing Unit	Đơn vị xử lý đồ họa
CCTV	Closed Circuit Television	Máy quay một địa điểm cụ thể
SOTA	State of The Art	Hiện đại
FPS	Frame per second	Số lượng khung hình trên mỗi giây
PX	Pixel	Pixel
.	Dot-product	Tích vô hướng
PRE	Precision	tỷ lệ giữa số kết quả dự đoán đúng là True trên tổng số kết quả dự đoán là True
REC	Recall	tỷ lệ giữa số kết quả dự đoán đúng là True trên tổng số kết quả thực sự là True

# Danh sách bảng

2.1	Kết quả độ chính xác của các cách chọn positional embedding trên tập dữ liệu K400 và SSv2. . . . .	20
3.1	Bảng tổng hợp thông tin một số tập dữ liệu cho bài toán tóm tắt video. . . . .	33
3.2	Bảng tổng hợp thiết lập tham số thí nghiệm. . . . .	36
3.3	Thông tin thiết bị thí nghiệm. . . . .	38
3.4	Bảng thống số kết quả thí nghiệm Timesformer trên tập dữ liệu SumMe theo thuật toán phân cụm và chọn lọc (1). . . . .	40
3.5	Bảng thống số kết quả thí nghiệm Timesformer trên tập dữ liệu SumMe theo thuật toán phân cụm và chọn lọc (2) và (3). . .	41
3.6	Bảng so sánh kết quả thí nghiệm với các nghiên cứu trước đây trên tập dữ liệu SumMe. . . . .	42

# Danh sách hình vẽ

1.1	Minh họa đầu vào của bài toán tóm tắt video. . . . .	4
1.2	Minh họa đầu ra của bài toán tóm tắt video. . . . .	5
1.3	Minh họa các lĩnh vực ứng dụng của bài toán tóm tắt video. . . . .	5
1.4	Minh họa ứng dụng của bài toán tóm tắt video trong lĩnh vực thể thao. (Nguồn: Youtube Kplus Sports) . . . . .	6
1.5	Minh họa ứng dụng của bài toán tóm tắt video trong lĩnh vực camera giám sát. (Nguồn: Youtube) . . . . .	7
1.6	Minh họa ứng dụng của bài toán tóm tắt video trong lĩnh vực tin tức. (Nguồn: <a href="https://retv-project.eu/2020/04/16/summarizing-videos-on-the-web/">https://retv-project.eu/2020/04/16/summarizing-videos-on-the-web/</a> ) . . . . .	8
1.7	Minh họa bộ dữ liệu quy mô lớn ImageNet [3]. . . . .	9
1.8	Minh họa các ý kiến đánh giá khác nhau trên một bản video đã tóm tắt. . . . .	10
1.9	Minh họa chi phí thiết bị GPU hiện đại ngày nay. (Nguồn: Google)	10
2.1	Minh họa chi phí tính toán của Transformer trên dữ liệu hình ảnh.	14
2.2	Minh họa bước phân rã hình ảnh thành các phần nhỏ hơn gọi là patch. . . . .	14
2.3	Minh họa bước lấy mẫu đầu vào mô hình từ video gốc. . . . .	16
2.4	Minh họa bước lấy mẫu đầu vào mô hình từ video gốc. . . . .	17
2.5	Minh họa bước phân rã một input clip thành các patch. . . . .	18
2.6	Minh họa bước chuyển đổi các patches thành các embedding vectors. . . . .	19
2.7	Minh họa kiến trúc tổng quan lớp Encoder của Timesformer. .	20





# Mục lục

<b>LỜI MỞ ĐẦU</b>	<b>1</b>
<b>1 GIỚI THIỆU</b>	<b>2</b>
1.1 Mô tả bài toán . . . . .	2
1.2 Đầu vào của bài toán . . . . .	4
1.3 Đầu ra của bài toán . . . . .	5
1.4 Ứng dụng của tóm tắt video . . . . .	5
1.5 Các thách thức của tóm tắt video . . . . .	8
1.6 Mục tiêu của đồ án . . . . .	11
1.7 Đóng góp chính của đồ án . . . . .	11
1.8 Cấu trúc của đồ án . . . . .	12
<b>2 TỔNG QUAN TIMESFORMER VÀ CÁC PHƯƠNG PHÁP SELF-ATTENTIONS</b>	<b>13</b>
2.1 Ý tưởng . . . . .	13
2.2 Dữ liệu đầu vào . . . . .	15
2.3 Kiến trúc tổng quan . . . . .	17
2.3.1 Phân rã . . . . .	17
2.3.2 Nhúng tuyến tính . . . . .	19
2.3.3 Tính toán Query-Key-Value . . . . .	20
2.3.4 Tính toán self-attention . . . . .	22
2.3.5 Mã hoá (Encoding) . . . . .	22
2.3.6 Class Embedding . . . . .	24
2.4 Các phương pháp Self-attentions . . . . .	24

2.4.1	Space Attention (S) . . . . .	25
2.4.2	Joint Space-Time Attention (ST) . . . . .	26
2.4.3	Divided Space-Time Attention (T+S) . . . . .	27
2.4.4	Sparse Local Global Attention (L+G) . . . . .	30
2.4.5	Axial Attention (T+W+H) . . . . .	31
<b>3</b>	<b>THÍ NGHIỆM VÀ ĐÁNH GIÁ</b>	<b>32</b>
3.1	Thiết lập thí nghiệm . . . . .	32
3.1.1	Tổng quan các bước thí nghiệm và thiết lập tham số . . . . .	32
3.1.2	Dữ liệu thí nghiệm . . . . .	37
3.1.3	Cấu hình phần cứng thiết bị thí nghiệm . . . . .	38
3.2	Kết quả thí nghiệm . . . . .	38
3.2.1	Phương pháp đánh giá . . . . .	38
3.2.2	Kết quả thí nghiệm . . . . .	40
3.3	Phân tích và thảo luận . . . . .	43
<b>4</b>	<b>KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN</b>	<b>46</b>
4.1	Kết luận . . . . .	46
4.2	Hướng phát triển . . . . .	47

# LỜI MỞ ĐẦU

Trong thời đại công nghệ số phát triển vượt bậc, khối lượng thông tin mà con người tiếp nhận hàng ngày đang gia tăng một cách chóng mặt. Trong đó, một lượng lớn dữ liệu video liên tục được sản sinh hàng ngày, từ video trên mạng xã hội, video giáo dục, video giải trí đến video an ninh,... Vì vậy, video đang trở thành phương tiện truyền tải nội dung phổ biến và hiệu quả nhất. Tuy nhiên, đi kèm theo với sự gia tăng về số lượng và đa dạng của dữ liệu video, việc tìm kiếm, trích xuất thông tin và hiểu nội dung của chúng đang trở nên phức tạp và tốn kém thời gian.

**Video Summarization**, hay tóm tắt video, đã nổi lên như một giải pháp tiềm năng để giải quyết vấn đề này. Tóm tắt video là quá trình tổng hợp các khung hình quan trọng được lấy ra từ video gốc thành một phiên bản ngắn hơn, nhưng vẫn giữ lại thông tin quan trọng và những điểm nổi bật của video gốc. Mục đích chính của tóm tắt video là giúp người xem tiết kiệm thời gian và công sức bằng cách cung cấp một cái nhìn tổng quan và nhanh chóng về nội dung chính của video.

Trong bối cảnh công nghệ phát triển nhanh chóng, các hệ thống tóm tắt video hiện đại thường áp dụng các thuật toán học sâu và mạng CNN để phân tích và hiểu nội dung video. Các hệ thống này có khả năng tự động xác định và trích xuất các khung hình hoặc đoạn video quan trọng dựa trên nhiều yếu tố như hình ảnh, âm thanh, và văn bản. Ngoài ra, một số hệ thống còn kết hợp khả năng xử lý ngôn ngữ tự nhiên để diễn giải lại nội dung video một cách mạch lạc và dễ hiểu.

Công nghệ tóm tắt video đang mở ra những khả năng mới trong việc tiêu thụ và quản lý thông tin số. Với sự tiến bộ không ngừng của trí tuệ nhân tạo và các thuật toán học sâu, tương lai của tóm tắt video hứa hẹn sẽ mang lại nhiều tiện ích và giá trị cho người dùng và doanh nghiệp.

# Chương 1

## GIỚI THIỆU

### 1.1 Mô tả bài toán

Tóm tắt video có thể được hiểu là quá trình rút gọn một video dài thành một video ngắn hơn hoặc một loạt các hình ảnh đại diện, mà vẫn duy trì được các thông tin cốt lõi và ý nghĩa chính của video gốc. Mục tiêu của quá trình này là làm sao để bản tóm tắt phản ánh chính xác và đầy đủ nội dung quan trọng mà người dùng quan tâm, giảm thiểu thời gian cần thiết để hiểu nội dung. Có hai phương pháp chính trong bài toán tóm tắt video:

- **Tóm tắt trích xuất (extractive summarization):** Phương pháp này tập trung vào việc chọn ra các khung hình (frames) hoặc đoạn video (segments) quan trọng từ video gốc để tạo thành bản tóm tắt. Các đoạn được chọn thường là những phần có nhiều hành động, âm thanh nổi bật, hoặc chứa thông tin then chốt. Phương pháp này chủ yếu dựa trên các đặc trưng bề mặt của video như màu sắc, chuyển động, và âm thanh.
- **Tóm tắt diễn giải (abstractive summarization):** Phương pháp này đòi hỏi hệ thống phải hiểu và diễn đạt lại nội dung video bằng ngôn ngữ tự nhiên hoặc hình thức khác, giúp người xem dễ dàng hiểu được ý nghĩa tổng quát của video. Đây là một nhiệm vụ phức tạp vì nó yêu cầu khả năng hiểu ngữ cảnh và tạo ra các diễn giải mới dựa trên nội dung gốc.

Gần đây, các bài toán trong lĩnh vực NLP thường sử dụng một phương pháp mang tên **self-attention**. Lý do chính là vì phương pháp này có thể giữ

được các thông tin của ngữ cảnh (mặc dù độ dài của sequence đầu vào có thể rất lớn) mà vẫn cho phép tính toán song song (parallelization) - điểm mạnh của việc sử dụng GPU. Điển hình, ta có ví dụ về kiến trúc mạng **Transformer**, một kiến trúc đã làm thay đổi toàn bộ hướng tiếp cận về các bài toán xử lý ngôn ngữ tự nhiên trong thời gian vài năm đổ lại. Vậy liệu ta có thể sử dụng lợi thế đó vào các bài toán trong lĩnh vực Thị giác máy tính, như là bài toán **Video Summarization**?

Tại sao lại là **Transformer**? Như đã biết, video là một chuỗi các frame ảnh liên tiếp, có liên quan tới nhau. Video thường sẽ chứa hai chiều là không gian (Spatial) - nội dung của từng frame trong video và thời gian (Temporal) - sự liên kết của các frame trong video. Chúng ta có thể dễ dàng nhận thấy được nó tương tự như dạng dữ liệu sequence, giống với một câu văn: một frame ảnh ứng với một từ trong câu, vị trí của từ trong câu tương ứng với vị trí của frame trong video. Do đó, có thể hoàn toàn áp dụng **Transformer** vào video như dạng dữ liệu sequence.

Năm 2021, bài báo có tiêu đề: "**Is Space-Time Attention All You Need for Video Understanding?**" [1] đã được phát hành bởi đội ngũ nghiên cứu của Facebook. Trong đó, tác giả đã giới thiệu một mô hình có tên **Timesformer**, mô hình này hoạt động dựa trên Transformer, hứa hẹn sẽ hoạt động tốt hơn các mô hình trước đó đồng thời hiệu quả hơn về mặt tính toán trong lĩnh vực Video Understanding.

Để giải quyết những thách thức trong bài toán tóm tắt video, mô hình **Timesformer** với khả năng học cấu trúc không gian thời gian của video thông qua việc áp dụng cơ chế self-attention hứa hẹn sẽ giải quyết được các thách thức to lớn của bài toán này. Cụ thể, ta sẽ sử dụng phương pháp **tóm tắt trích xuất** với **Timesformer** đóng vai trò trích xuất đặc trưng hình ảnh từ các khung hình trong video gốc, từ đó, nắm bắt được những khung hình quan trọng, giúp việc tóm tắt trở nên hiệu quả, chính xác và đảm bảo được nội dung chính mà video gốc muốn truyền tải.

## 1.2 Đầu vào của bài toán

Đầu vào của bài toán này là các video, được biểu diễn dưới dạng chuỗi các khung hình (frames). Mỗi video là một chuỗi các khung hình, mỗi khung hình thể hiện một trạng thái của video tại một thời điểm cụ thể. Đối với mỗi video, thông tin về độ phân giải, tỷ lệ khung hình, và độ dài của video cũng được cung cấp.

Các video có thể có độ phức tạp và đa dạng khác nhau, từ video ngắn vài phút đến video dài có thể kéo dài hàng giờ. Ngoài ra, nội dung và ngữ cảnh cũng có thể khác nhau, từ các video giáo dục, giải trí đến các video an ninh, ... (hình 1.1).



Hình 1.1: Minh họa đầu vào của bài toán tóm tắt video.

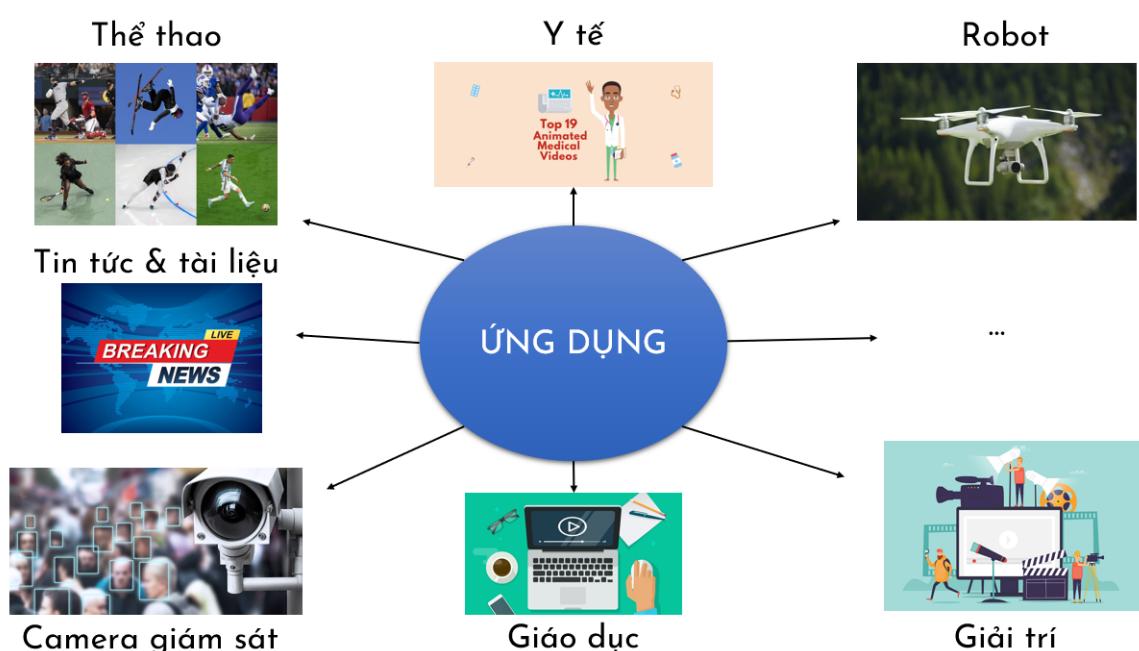
## 1.3 Đầu ra của bài toán

Đầu ra của bài toán này là một bản tóm tắt video, tức là một phiên bản ngắn gọn của video gốc, giữ lại những thông tin quan trọng và điểm nổi bật của video đó (hình 1.2).



Hình 1.2: Minh họa đầu ra của bài toán tóm tắt video.

## 1.4 Ứng dụng của tóm tắt video



Hình 1.3: Minh họa các lĩnh vực ứng dụng của bài toán tóm tắt video.

Từ hình 1.3, có thể thấy tóm tắt video có nhiều ứng dụng tiềm năng trong nhiều lĩnh vực khác nhau, mở ra rất nhiều cơ hội mới. Từ giáo dục, giải trí, an ninh đến nghiên cứu khoa học, việc tự động tóm tắt video có thể giúp nâng cao hiệu quả làm việc và trải nghiệm người dùng trong nhiều ngữ cảnh khác nhau. Với công nghệ ngày càng tiến bộ, những phương pháp mới như học máy và học sâu, đang mở ra những khả năng mới cho việc tạo ra các bản tóm tắt video chất lượng cao hơn.

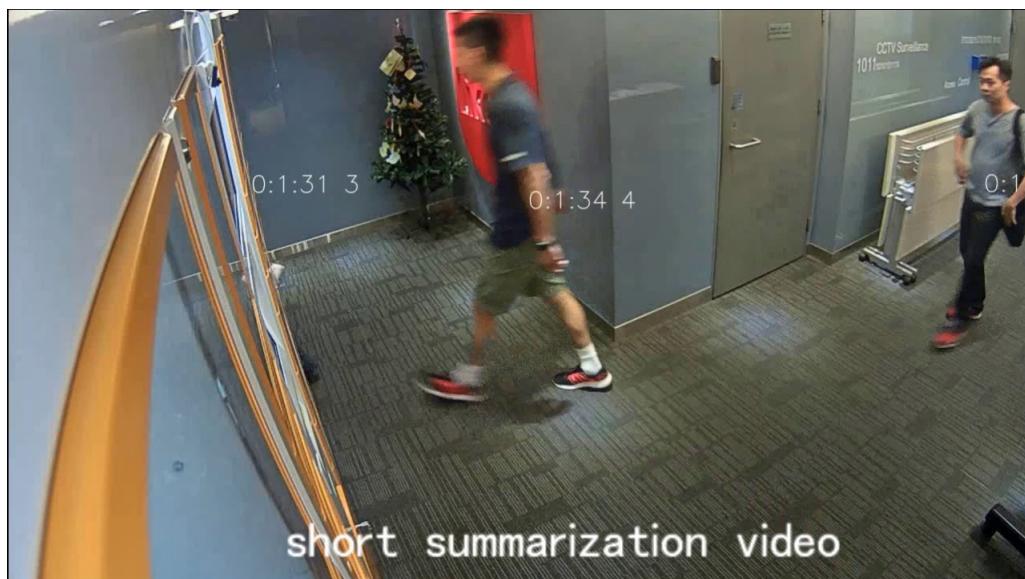
- **Thể thao:** Trong các sự kiện thể thao, video tóm tắt các pha hành động nổi bật, giúp khán giả xem lại những khoảnh khắc đáng nhớ mà không cần theo dõi toàn bộ trận đấu.



Hình 1.4: Minh họa ứng dụng của bài toán tóm tắt video trong lĩnh vực thể thao.  
(Nguồn: Youtube Kplus Sports)

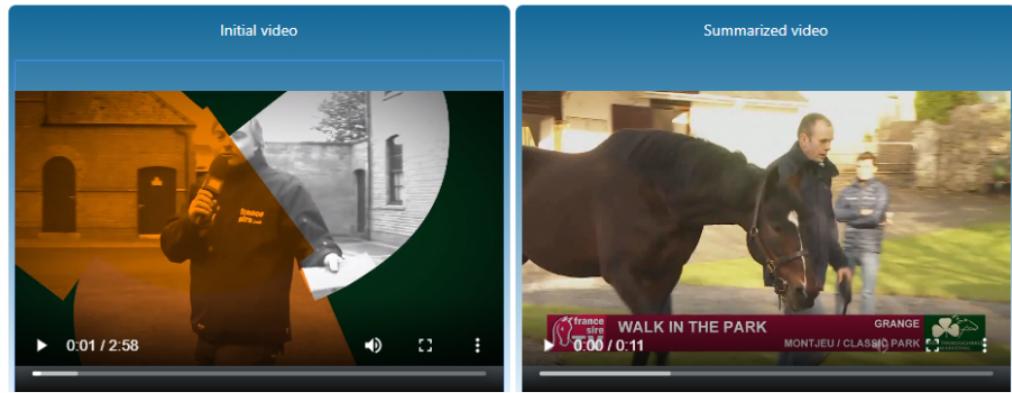
- **Y tế:** Các video y tế như video nội soi, bản ghi các thủ tục y tế và phẫu thuật, video nội soi chẩn đoán, v.v. mở ra nhiều cơ hội cho việc tóm tắt video tự động. Việc tóm tắt các video y khoa dài cho phép các bác sĩ y khoa thực hiện kiểm tra chi tiết các thủ tục y tế và hỗ trợ giảng dạy các thủ tục cho sinh viên y khoa. Điều này cũng sẽ cho phép các chuyên gia y tế tìm kiếm các trường hợp tương tự hoặc xem xét chi tiết các trường hợp cũ ngay lập tức.

- **Robot:** Máy ảnh đang được tích hợp vào các thiết bị được phát triển gần đây như máy bay không người lái và robot trang bị cho chúng để quay video ở nhiều nơi mà con người không thể tiếp cận được. Việc tóm tắt những video này sẽ giúp bạn dễ hiểu hơn về loại video mới này.
- **Camera giám sát:** Camera CCTV được lắp đặt và các video giám sát đang được ghi lại suốt ngày đêm ở những nơi quan trọng về an ninh, dẫn đến một lượng lớn dữ liệu video được tạo ra vô tận. Một bản tóm tắt video giúp dễ dàng giải thích các tình huống thực tế liên quan đến vấn đề an toàn, đặc biệt khi nhiều camera được sử dụng để ghi lại video giám sát ở một địa điểm.



Hình 1.5: Minh họa ứng dụng của bài toán tóm tắt video trong lĩnh vực camera giám sát.  
(Nguồn: Youtube)

- **Tin tức và tài liệu:** Việc tóm tắt các video tin tức một cách tự động cho phép chúng ta nhanh chóng tìm ra các mẩu quan trọng được hiển thị trong tin tức. Ngành truyền thông và báo chí có thể sử dụng tóm tắt video để thu hút khán giả và tăng cường hiệu quả truyền tải thông tin.



Hình 1.6: Minh họa ứng dụng của bài toán tóm tắt video trong lĩnh vực tin tức.  
(Nguồn: <https://retv-project.eu/2020/04/16/summarizing-videos-on-the-web/>)

- **Giáo dục:** Trong lĩnh vực giáo dục, tóm tắt video có thể đóng vai trò quan trọng trong việc giúp học sinh và giáo viên tiết kiệm thời gian và nâng cao hiệu quả học tập.
- **Giải trí:** Trong lĩnh vực giải trí, tóm tắt video mang lại nhiều lợi ích, từ việc giúp khán giả chọn lựa nội dung đến việc quảng bá sản phẩm như cung cấp cái nhìn tổng quan về nội dung phim hoặc chương trình TV, giúp khán giả quyết định xem có nên đầu tư thời gian để xem toàn bộ hay không.

## 1.5 Các thách thức của tóm tắt video

Bài toán tóm tắt video là một lĩnh vực đầy hứa hẹn trong trí tuệ nhân tạo và xử lý video, nhưng nó cũng đối mặt với nhiều thách thức phức tạp. Các thách thức này xuất phát từ bản chất đa dạng và phức tạp của video cũng như yêu cầu cao về tính chính xác, mạch lạc và ý nghĩa của bản tóm tắt. Ví dụ như:

- **Nguồn dữ liệu đa dạng, rộng lớn nhưng khó tận dụng được:** Các bộ dữ liệu lớn có nhãn rõ ràng đóng một vai trò quan trọng trong cộng đồng thị giác máy tính. Nếu không có các bộ dữ liệu mở chất lượng cao như ImageNet (hình 1.8), MSCOCO và Kinetics, nhiều nghiên cứu sẽ không thể thành công như ngày nay. Mặt khác, tóm tắt video không có bộ dữ liệu quy mô lớn như vậy, chủ yếu là do chi phí xây dựng và chú thích bộ dữ liệu tóm tắt video rất lớn. Việc tạo các bản tóm tắt tham khảo hoặc

video có chủ thích dày đặc tồn nhiều công sức và khó mở rộng quy mô.



Hình 1.7: Minh họa bộ dữ liệu quy mô lớn ImageNet [3].

- **Khó khăn trong việc xác định các thước đo đánh giá phù hợp:** Các nhà nghiên cứu chưa thiết lập được cách đánh giá chất lượng của các video tóm tắt. Chất lượng của bản tóm tắt video mang tính chủ quan cao và có thể có nhiều bản tóm tắt hay cho một video. Hơn nữa, các thuộc tính mong muốn khác nhau tùy thuộc vào ứng dụng và miền video. Do thiếu các phương pháp đánh giá đáng tin cậy nên việc so sánh các phương pháp tóm tắt video rất khó khăn. Tương tự như các nhiệm vụ sáng tạo khác như tạo hình ảnh, đánh giá định lượng tính thẩm mỹ của video cũng là một vấn đề quan trọng nhưng ít được nghiên cứu trong lĩnh vực này.

## Video tóm tắt



Hình 1.8: Minh họa các ý kiến đánh giá khác nhau trên một bản video đã tóm tắt.

- **Yêu cầu phần cứng:** Để làm việc với dữ liệu video, Timesformer đòi hỏi một hệ thống GPU mạnh mẽ để đảm bảo tính độ chính xác, hiệu quả của bài toán. Do đó, dẫn tới áp lực về chi phí triển khai cho các dự án thực tế.



Hình 1.9: Minh họa chi phí thiết bị GPU hiện đại ngày nay. (Nguồn: Google)

## 1.6 Mục tiêu của đồ án

Các mục tiêu đặt ra của đồ án này bao gồm:

- **Thứ nhất**, phân tích, hiểu và trình bày được cách Timesformer sử dụng cơ chế self-attention để xử lý thông tin thời gian và không gian trong video. Để làm được điều này, đồ án sẽ đi sâu vào các lớp và thành phần của mô hình.
- **Thứ hai**, sử dụng Timesformer để trích xuất đặc trưng ảnh, kết hợp phân cụm K-means để xây dựng hệ thống có khả năng tóm tắt video một cách chính xác và hiệu quả.
- **Cuối cùng**, thực nghiệm hệ thống với bộ dữ liệu điểm chuẩn **SumMe** và so sánh kết quả với các phương pháp khác, nhằm đánh giá độ chính xác, độ đầy đủ và tính mạch lạc của bản tóm tắt. Ngoài ra, phân tích kết quả thí nghiệm để điều chỉnh và tối ưu hóa các tham số.

## 1.7 Đóng góp chính của đồ án

### Đồ án **Tìm Hiểu Timesformer và Ứng Dụng Cho Bài Toán Tóm Tắt Video**

**Video** là một công trình nghiên cứu mang tính cá nhân của tôi trong việc áp dụng các mô hình hiện đại để giải quyết một vấn đề phức tạp và có ý nghĩa thực tiễn cao. Tuy nhiên, tôi mong muốn và hi vọng rằng, đồ án của mình sẽ giúp đỡ nhiều nhất có thể cho cộng đồng. Dưới đây là một số những đóng góp chính của đồ án:

- **Khám phá mô hình Timesformer:** Mô hình Timesformer có thể được xem là một mô hình SOTA trong lĩnh vực Video Understanding, mô hình này được xem là một bước tiến quan trọng trong việc xử lí video. Đồ án thực hiện phân tích sâu các thành phần hoạt động và cơ chế self-attention của Timesformer, điều này giúp làm rõ cách mô hình nắm bắt các mối quan hệ phức tạp không gian - thời gian giữa các khung hình (có hình minh họa các cơ chế này ở phần sau của đồ án). Hi vọng rằng việc phân

tích và minh họa rõ cơ chế hoạt động của mô hình này sẽ giúp các sinh viên, nhà nghiên cứu khác nắm rõ về Timesformer.

- **Triển khai Timesformer cho bài toán tóm tắt video:** Đồ án sử dụng Timesformer với vai trò trích xuất đặc trưng hình ảnh, xây dựng lên hệ thống tóm tắt video tạo ra các bản tóm tắt ngắn gọn nhưng đầy đủ thông tin của video gốc. Với cơ chế học không gian - thời gian đầy tiềm năng, hi vọng rằng hệ thống tóm tắt video sẽ hoạt động hiệu quả và mang đến một công cụ hỗ trợ đắc lực ứng dụng cho nhiều lĩnh vực khác nhau.

Tóm lại, bằng việc công bố các kết quả thực nghiệm, những phát hiện và đóng góp từ đồ án sẽ là cơ sở quan trọng để các nghiên cứu tiếp theo có thể tiếp tục phát triển và hoàn thiện các ứng dụng của Timesformer trong xử lý video.

## 1.8 Cấu trúc của đồ án

**Chương 1** Giới thiệu, mô tả tổng quan bài toán tóm tắt video.

**Chương 2** Trình bày tổng quan về mô hình Timesformer và các phương pháp self-attentions. Trong đó, tôi mô tả một cách khái quát các nội dung, kiến trúc, công thức và các đánh giá của tác giả.

**Chương 3** Thử nghiệm, đánh giá, thảo luận với kết quả thực nghiệm đạt được.

**Chương 4** Kết luận chung về mô hình Timesformer và bài toán tóm tắt video, cuối cùng, đưa ra phương hướng phát triển tiếp theo cho đồ án.

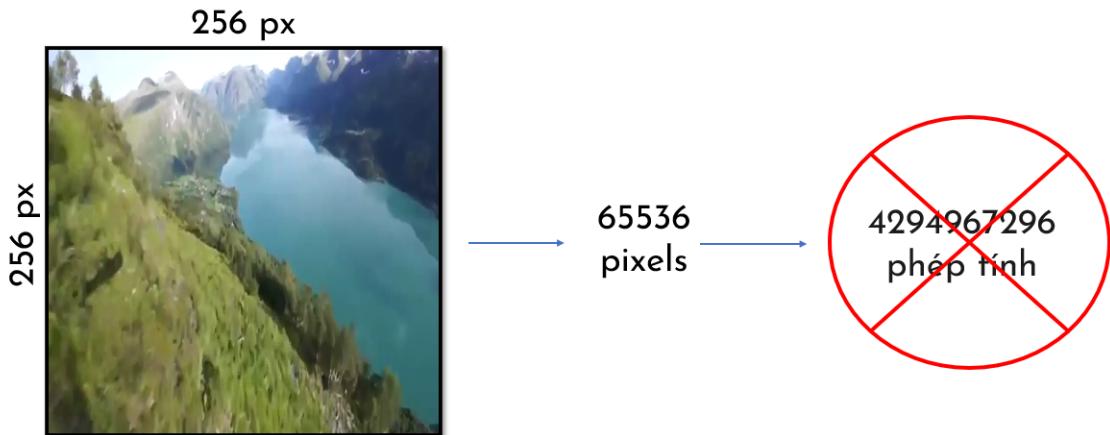
## Chương 2

# TỔNG QUAN TIMESFORMER VÀ CÁC PHƯƠNG PHÁP SELF-ATTENTIONS

### 2.1 Ý tưởng

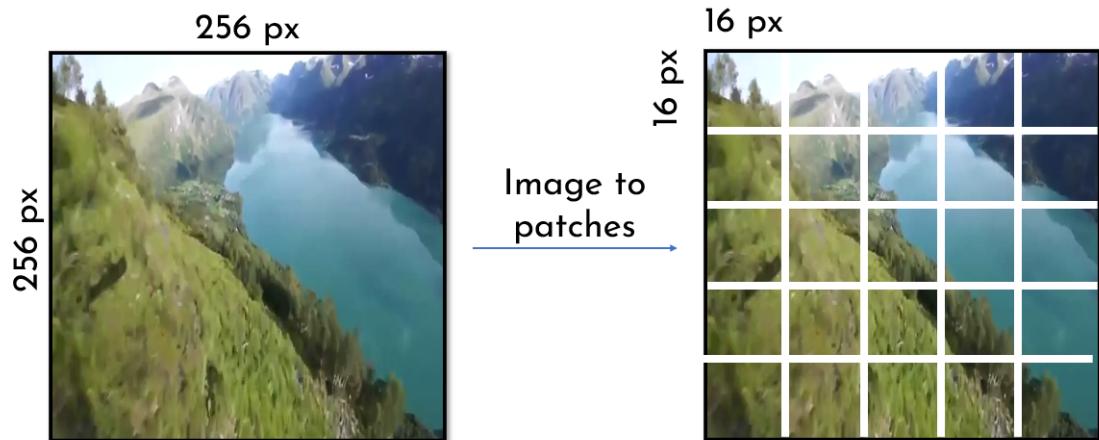
**Timesformer** là một kiến trúc mạng neural network ứng dụng cho bài toán phân loại video. Mô hình hoạt động dựa trên kiến trúc Transformer, kiến trúc đã rất thành công trong các tác vụ xử lý ngôn ngữ tự nhiên. Tuy nhiên, trong khi Transformer ban đầu được thiết kế cho dữ liệu tuần tự, Timesformer được thiết kế cho dữ liệu video, có cả chiều không gian và thời gian. Điều này cho phép mô hình tìm hiểu sự phụ thuộc giữa các khung hình trong video và giữa các phần khác nhau trong mỗi hình ảnh.

Giả sử, video có độ phân giải là  $256 \times 256$ , số lượng pixel trong 1 frame sẽ là  $256 \times 256 = 65536$  pixels. Khi áp dụng cơ chế self-attention, mỗi một pixel cần phải được so sánh với tất cả các pixel còn lại trong cùng một frame, dẫn đến độ phức tạp thuật toán là  $O(n^2)$ , khi đó số lượng phép tính sẽ là  $65536^2 = 4294967296$  phép tính. Như vậy, có hơn 4 tỷ phép tính cho 1 frame có độ phân giải  $256 \times 256$ . Nếu video đó dài 10 giây, mỗi giây có 30 frame (FPS = 30) thì chúng ta sẽ cần  $(256 \times 256 \times 30 \times 10)^2 = 386547056640000$  phép tính, gần 400 nghìn tỷ phép tính cho một video ngắn 10s và độ phân giải gần bằng 240, điều này rõ ràng là phi thực tế và lãng phí (hình 2.1).



Hình 2.1: Minh họa chi phí tính toán của Transformer trên dữ liệu hình ảnh.

Vì vậy, Timesformer đã áp dụng 2 cách thức nhằm giải quyết vấn đề trên của Transformer:



Hình 2.2: Minh họa bước phân rã hình ảnh thành các phần nhỏ hơn gọi là patch.

- **Thứ nhất**, Timesformer chia frame thành các phần nhỏ hơn đồng đều nhau, gọi là patch, mỗi patch sẽ đại diện cho một phần tử (token) để tính toán self-attention (hình 2.2). Đây là phương pháp đã được áp dụng trong mạng ViT [5] nhằm giảm số lượng phần tử cần tính trong self-attention mà vẫn giữ được đầy đủ thông tin cần thiết. Trong bài báo, tác giả đã đặt kích thước của từng patch là 16 x 16, tương tự với kích thước patch có

trong bài báo gốc của ViT.

- **Thứ hai**, Timesformer sử dụng cơ chế self-attention theo chiều không gian và thời gian, gọi là “**Divided Space-Time Attention**”. Trong chiều thời gian, mỗi patch tập trung vào các patch có cùng vị trí không gian trong các frame còn lại, mặt khác, trong chiều không gian, mỗi patch tập trung vào các patch trong cùng frame chứa patch đang xét. Có tổng cộng 5 phương pháp tính toán self-attention đã được giới thiệu trong bài báo.

## 2.2 Dữ liệu đầu vào

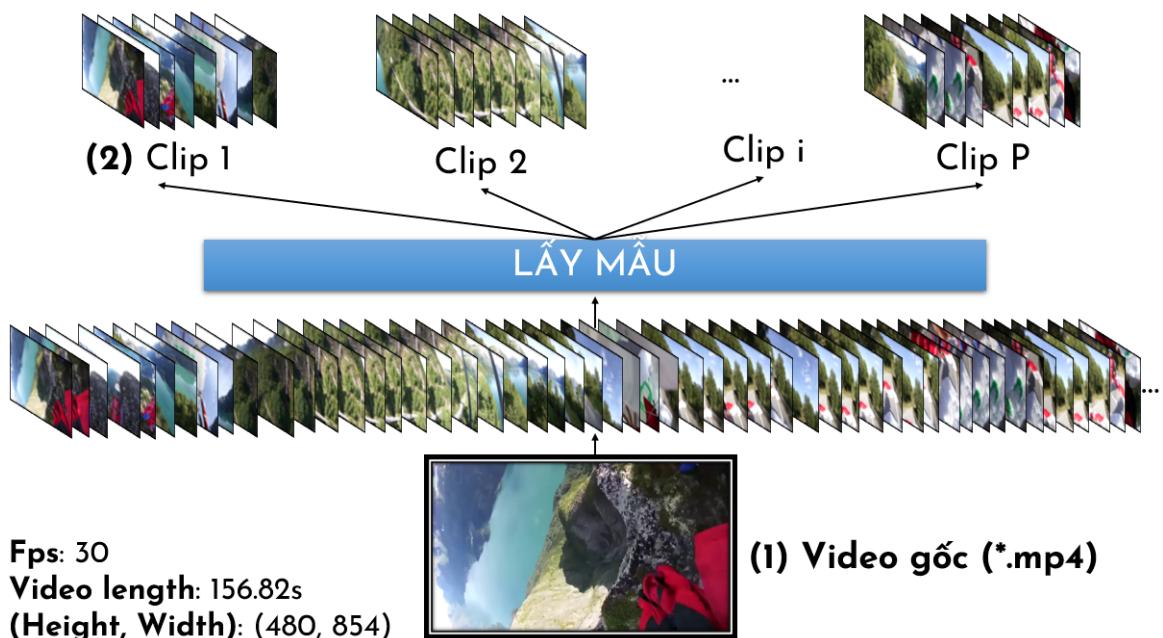
Đầu vào của mạng là một clip có kích thước  $(H, W, 3, F)$  cùng với một vài thông số:  $N, P$ . Ngoài ra, mặc định các clip đều có mức FPS là 30. Chi tiết hơn về những thông số trên như sau:

- **(H, W)** chính là độ phân giải của clip,  $(H, W)$  sẽ được resize sao cho có cùng kích thước (frame sẽ có dạng hình vuông), ví dụ như  $224 \times 224$ .
- **3** là số kênh màu (RGB).
- **F** là số frame của clip.
- **P** là kích thước của từng patch.
- **N** là tổng số patch có trong 1 frame, công thức tính là  $\frac{H*W}{P*P}$ .

Từ các thông số trên, tác giả đã giới thiệu 3 mạng Timesformer với các thiết lập khác nhau, với các tên gọi là Timesformer-baseline, Timesformer-HR xử lí video có độ phân giải cao và Timesformer-L xử lí video dài. Chi tiết như sau:

- **Timesformer-baseline**:  $224 \times 224 \times 3 \times 8$ , F/Fs: 1/32.  
(F/Fs là tần số lấy mẫu. Ví dụ, một video gốc có 256 frames, nếu F/Fs là 1/32 thì sẽ có tổng cộng 8 frames được lấy ra).
- **Timesformer-HR**:  $448 \times 448 \times 3 \times 8$ , F/Fs: Không có sẵn.
- **Timesformer-L**:  $224 \times 224 \times 3 \times 96$ , F/Fs: 1/4.

Dễ dàng nhận thấy, với các thông số trên, thời gian của clip gốc trước khi cắt thành input clip của mạng **Timesformer-baseline** sẽ là  $\frac{F/(F/F_s)}{FPS} = \frac{8/(1/32)}{30} \sim 8.5s$ , tương tự với hai mạng còn lại. Đối với các tập dữ liệu được sử dụng làm điểm chuẩn (benchmark) cho bài toán Action Recognition như Kinetics hay SSv2, với thời gian mỗi clip chỉ rơi vào khoảng 10s, là phù hợp với đầu vào của Timesformer. Tuy nhiên, thời lượng trên là quá ngắn so với độ dài video thực tế, vì vậy, giải pháp của tác giả là chia video gốc thành các clip ngắn hơn với độ dài của các clip đó sẽ có giá trị giống với đầu vào của mạng, sau đó sẽ tính toán kết quả với từng clip cắt được và tính trung bình để ra kết quả cuối cùng. Cụ thể, tác giả thực hiện việc lấy mẫu input clip từ video gốc như hình 2.3 dưới đây:



Hình 2.3: Minh họa bước lấy mẫu đầu vào mô hình từ video gốc.

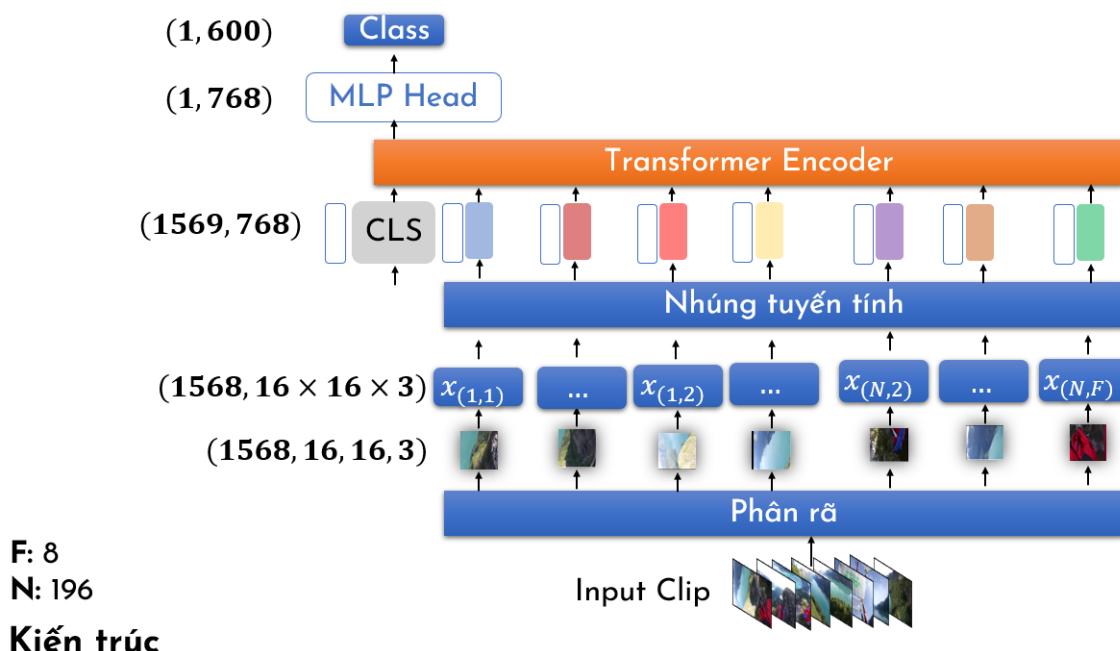
Từ **video gốc (1)**, đầu tiên, tiến hành lấy ra các frame có vị trí chia hết cho 32, nói cách khác, cứ mỗi lần duyệt qua 32 frames thì sẽ lấy ra 1 frame (điều này biểu diễn cho chỉ số  $F/F_s$  là  $1/32$ ). Tiếp theo, sau khi có được một chuỗi các frame được chọn, gọi là các key frame, thực hiện gom lần lượt theo thứ tự một lần 8 key frames tạo thành **input clip (2)** cho mô hình Timesformer. Ví dụ ở hình 1, tổng số frames của video gốc là  $video-length * FPS = 156.82 * 30 = 4707$  frames, qua khối **Lấy Mẫu** thu được  $4707 * \frac{1}{32} = 147$  frames, cuối cùng, thực

hiện gom các key frame và thu được  $\frac{147}{8} = 18$  input clips (mỗi clip có 8 frames).

Chú ý, trong quá trình cắt video gốc thành các input clip, cần dùng một số phép transform để biến đổi raw data thành data phù hợp, ví dụ như phép Resize, RandomCrop, Lambda, Normalize. Ngoài ra, có một số phép transform giúp tăng cường data như RandomHorizontalFlip, RandomShortSideScale.

## 2.3 Kiến trúc tổng quan

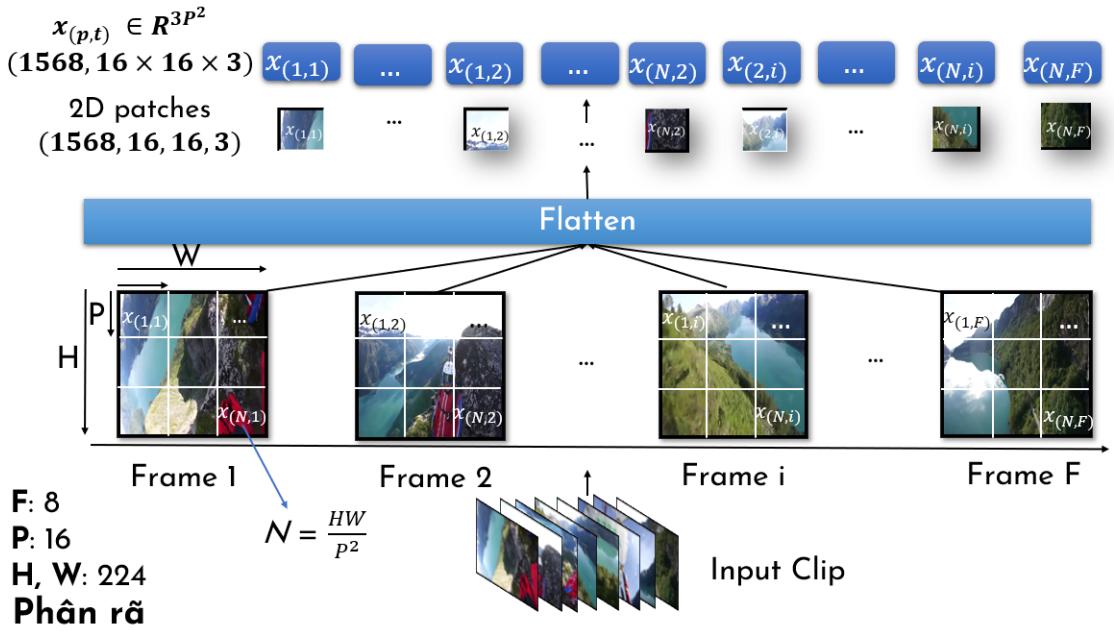
Về tổng quát, kiến trúc mô hình **Timesformer** được biểu diễn như hình 2.4 dưới đây (cấu hình theo mô hình pre-trained **Timesformer** trên tập dữ liệu Kinetics-600).



Hình 2.4: Minh họa bước lấy mẫu đầu vào mô hình từ video gốc.

### 2.3.1 Phân rã

Sau khi nhận đầu vào, mô hình tiến hành quá trình phân rã input clip như hình 2.5 dưới đây:



Hình 2.5: Minh họa bước phân rã một input clip thành các patch.

Mỗi frame trong input clip sẽ được chia thành N patches không chồng chéo (non-overlapping) theo công thức 2.1. Từ đó, input clip sẽ có tổng số patch là  $M = N * F$  với F là tổng số frame. Ví dụ trong hình 2.5, có  $M = N * F = \frac{H*W}{P*P} * F = \frac{224*224}{16*16} * 8 = 1568$  patch.

$$N = \frac{H * W}{P * P} \quad (2.1)$$

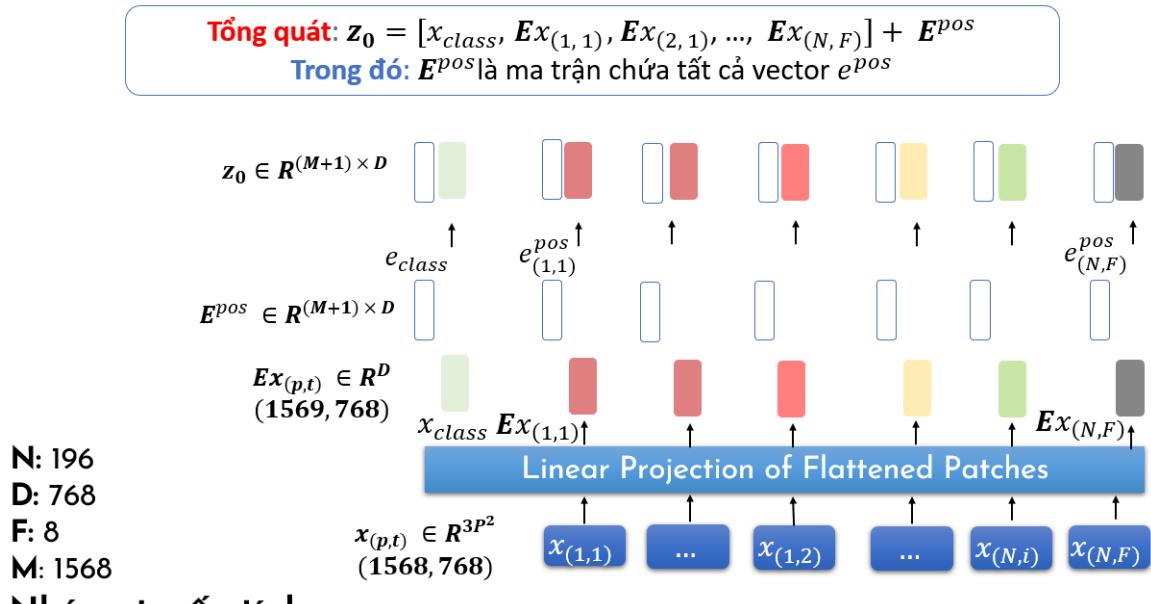
**Trong đó:**

- **H** là chiều cao của frame.
- **W** là chiều rộng của frame.
- **P** là kích thước của từng patch.

Sau đó, mô hình sẽ làm phẳng những patch này thành các vector  $x_{(p,t)} \in R^{3P^2}$  với  $p = 1, \dots, N$  là vị trí của patch đó trong frame t và  $t = 1, \dots, F$  là vị trí của frame chứa patch đó trong input clip.

### 2.3.2 Nhúng tuyến tính

Sau khi hoàn thành quá trình phân rã và làm phẳng, mô hình tiến hành nhúng tuyến tính bằng cách chuyển đổi các patch thành các embedding vectors như hình 2.6 dưới đây:



Hình 2.6: Minh họa bước chuyển đổi các patches thành các embedding vectors.

Cụ thể, mô hình tiến hành ánh xạ tuyến tính từng patch  $x_{(p,t)}$  thành embedding vector  $z_{(p,t)}^0 \in \mathbb{R}^D$  bằng cách kết hợp nhân với một ma trận learnable weight  $E \in \mathbb{R}^{D \times 3P^2}$  và cộng thêm vào một learnable positional embedding  $e_{(p,t)}^{pos} \in \mathbb{R}^D$  để mã hoá vị trí không gian và thời gian của mỗi patch theo như công thức 2.2.

$$z_{(p,t)}^0 = \mathbf{Ex}_{(p,t)} + e_{(p,t)}^{pos} \quad (2.2)$$

$z_0$ , chuỗi kết quả của các embedding vector  $z_{(p,t)}^0$  với  $p = 1, \dots, N$  và  $t = 1, \dots, F$  sẽ là đầu vào của Transformer Encoder, đóng vai trò tương tự như chuỗi các embedded words được đưa vào Transformer trong NLP. Ngoài ra, tương tự với BERT Transformer [4], tác giả cũng thêm vào vị trí đầu tiên của chuỗi một learnable vector  $z_{(0,0)}^0 \in \mathbb{R}^D$  đại diện cho embedding của classification token.

Ngoài ra, để nghiên cứu tầm quan trọng của việc chọn learnable positional embedding  $e_{(p,t)}^{pos}$ , tác giả tiến hành thử nghiệm Timesformer với một số biến thể,

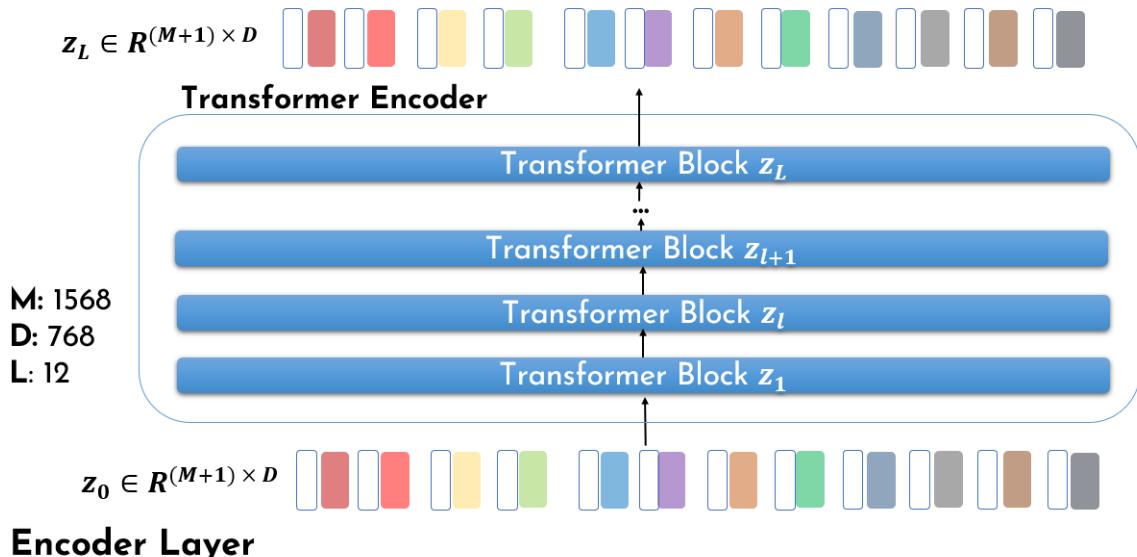
bao gồm: (1) no positional embedding, (2) space-only positional embedding và (3) space-time positional embedding. Dựa trên những kết quả thực nghiệm ở bảng 2.1, tác giả nhận thấy rằng (3) tạo ra độ chính xác tốt nhất trên cả tập dữ liệu Kinetics-400 (K400) [2] và Something-Something-V2 (SSv2) [6]. Có một điều thú vị, tác giả cũng nhận thấy rằng việc sử dụng (2) sẽ dẫn đến kết quả tốt trên tập dữ liệu K400, nhưng lại cho kết quả tệ hơn nhiều trên SSv2. Điều này là có lý vì tập dữ liệu K400 thiên về không gian hơn, trong khi SSv2 yêu cầu lý luận thời gian phức tạp, điều này đã được phân tích trong một nghiên cứu trước đây [14].

Bảng 2.1: Kết quả độ chính xác của các cách chọn positional embedding trên tập dữ liệu K400 và SSv2.

Positional Embedding	K400 (%)	SSv2 (%)
(1) None	75.4	45.8
(2) Space-only	77.8	52.8
(3) Space-Time	78.0	59.5

### 2.3.3 Tính toán Query-Key-Value

Trong mô hình Timesformer, lớp Encoder bao gồm  $L$  khối Encoding (Transformer Block), được biểu diễn như hình 2.7 dưới đây:



Hình 2.7: Minh họa kiến trúc tổng quan lớp Encoder của Timesformer.

Ở mỗi khối Encoding, giá trị vector query/key/value (q/k/v) được tính cho mỗi patch từ representation  $z_{(p,t)}^{(l-1)}$  encoded trước đó theo công thức 2.3, 2.4, 2.5,

quy trình được biểu diễn ở hình 2.8:

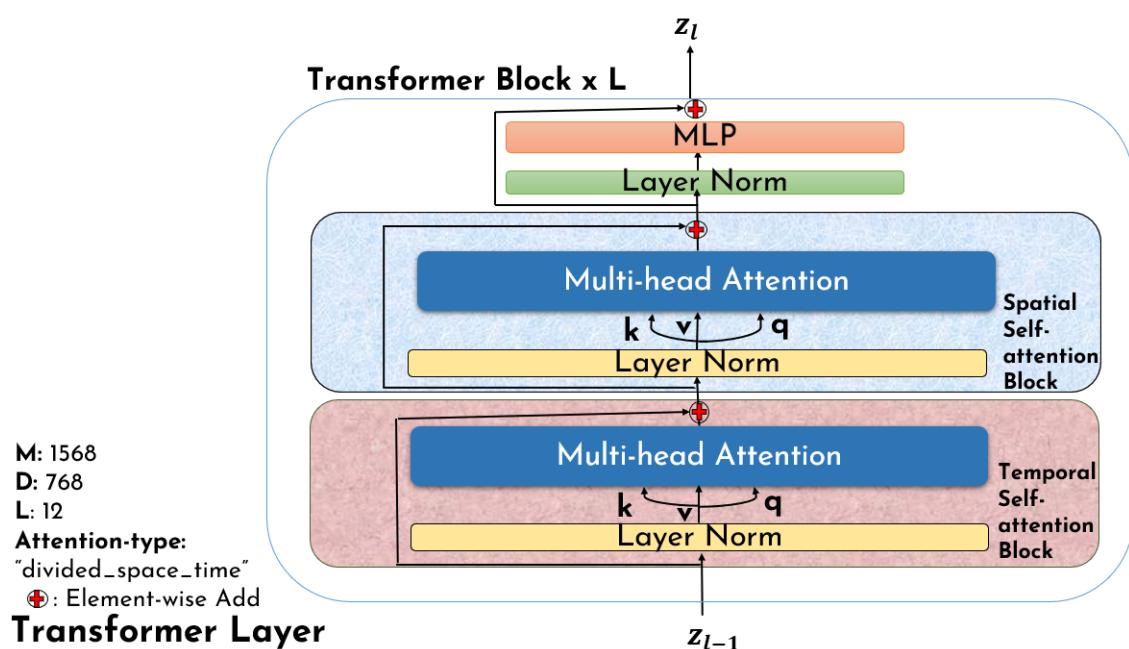
$$q_{(p,t)}^{(l,a)} = W_Q^{(l,a)} LN(z_{(p,t)}^{(l-1)}) \in R^{D_h} \quad (2.3)$$

$$k_{(p,t)}^{(l,a)} = W_K^{(l,a)} LN(z_{(p,t)}^{(l-1)}) \in R^{D_h} \quad (2.4)$$

$$v_{(p,t)}^{(l,a)} = W_V^{(l,a)} LN(z_{(p,t)}^{(l-1)}) \in R^{D_h} \quad (2.5)$$

**Trong đó:**

- **LN()** biểu thị cho hàm LayerNorm (viết tắt của Layer Normalization), là một kỹ thuật chuẩn hóa đầu vào hoặc đầu ra của mỗi lớp trong mạng nơ-ron. Mục tiêu của Layer Normalization là giúp duy trì đồng nhất giữa phân phối của đầu vào hoặc đầu ra của các lớp trong mạng nơ-ron, giúp cho việc huấn luyện mạng nơ-ron hiệu quả hơn.
- **I** là vị trí khối Transformer của patch đang xét.
- **a = 1, … , A** là chỉ số multiple attention heads với A là tổng số multiple attention heads.
- $W_Q, W_K, W_V$  lần lượt là ma trận Query, ma trận Value, ma trận Key.
- Chiều ẩn (the latent dimensionality) cho mỗi attention head được đặt thành  $D_h = \frac{D}{A}$ .



Hình 2.8: Minh họa kiến trúc Transformer Block và công thức tính q/k/v.

### 2.3.4 Tính toán self-attention

Trọng số self-attention được tính toán thông qua dot-product, trọng số self-attention  $\alpha_{(p,t)}^{(l,a)} \in R^{N*F+1}$  cho query của patch (p,t) được tính bởi công thức 2.6.

$$\alpha_{(p,t)}^{(l,a)} = SM\left(\frac{q_{(p,t)}^{(l,a)}}{\sqrt{D_h}} \cdot [k_{(0,0)}^{(l,a)} \{k_{(p',t')}^{(l,a)}\}_{p'=1,\dots,N; t'=1,\dots,F}]\right) \quad (2.6)$$

Trong đó:

- $SM()$  biểu thị cho hàm Softmax Activation, được sử dụng để chuyển đổi đầu ra của một mạng nơ-ron thành xác suất.
- $l$  là vị trí khối Transformer của patch đang xét.
- $a = 1, \dots, A$  là chỉ số multiple attention heads với  $A$  là tổng số multiple attention heads.
- $N$  là số patch trong 1 frame.
- $F$  là tổng số frame của input clip.
- $D_h$  là chiều ẩn (the latent dimensionality) cho mỗi attention head.

Chú ý rằng 2.6 đang là công thức dot-product ở cả hai chiều Spatial và Temporal. Nếu self-attention chỉ tính ở một chiều duy nhất như spatial-only hoặc temporal-only thì việc tính toán sẽ giảm đáng kể.

### 2.3.5 Mã hóa (Encoding)

Như vậy,  $z_{(p,t)}^{(l)}$  tại Encoding block  $l$  thu được bằng tính toán tổng trọng số của các  $v$  vector bằng cách sử dụng hệ số self-attention ở mỗi attention head theo công thức 2.7.

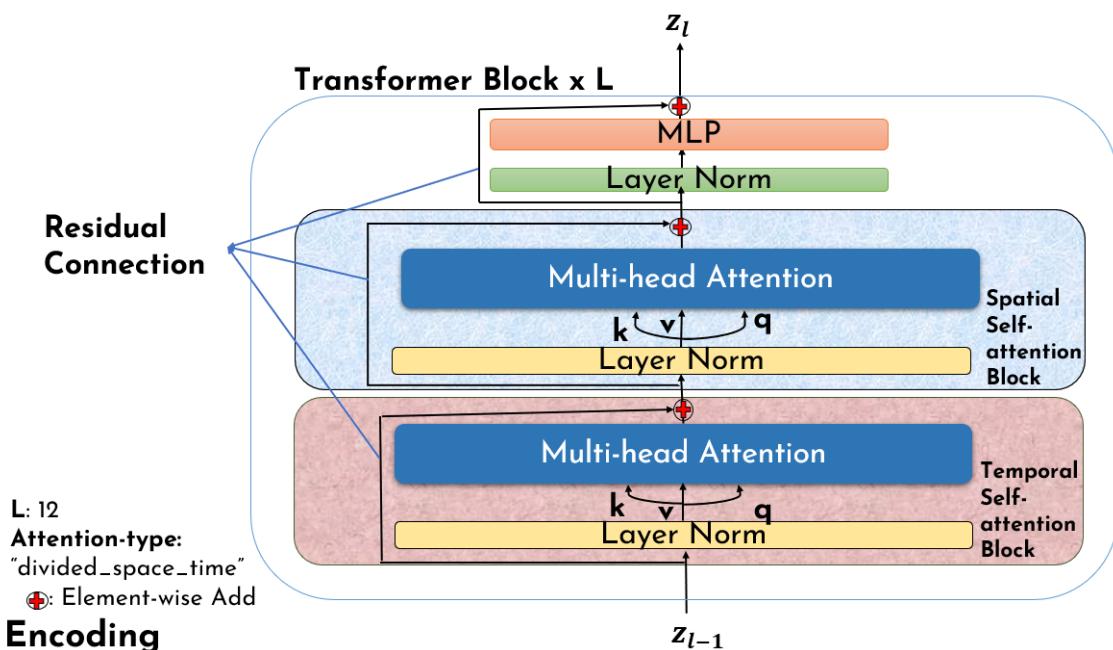
$$s_{(p,t)}^{(l,a)} = \alpha_{(p,t),(0,0)}^{(l,a)} v_{(0,0)}^{(l,a)} + \sum_{p'=1}^N \sum_{t'=1}^F \alpha_{(p,t),(p',t')}^{(l,a)} v_{(p',t')}^{(l,a)} \quad (2.7)$$

Trong đó:

- $\alpha$  là trọng số self-attention.
- $l$  là vị trí khối Transformer của patch đang xét.

- $a = 1, \dots, A$  là chỉ số multiple attention heads với  $A$  là tổng số multiple attention heads.
- $v$  là value vector.
- $N$  là số patch trong 1 frame.
- $F$  là tổng số frame của input clip.

Sau khi có được Encoding từ các bước trên, tác giả tiến hành kết hợp với residual connection và đưa qua một lớp MLP rồi truyền kết quả cho lớp tiếp theo như trong hình 2.9 dưới đây:



Hình 2.9: Minh họa bước kết hợp residual connection và MLP để feed kết quả encoding cho Transformer Block tiếp theo.

Trong đó, **MLP**, hay còn gọi là Multilayer Perceptron, là một phần của kiến trúc mạng nơ-ron được sử dụng để ánh xạ và biến đổi các đặc trưng đầu vào. Ngoài ra, **residual connection**, hoặc còn gọi là **skip connection**, là một kỹ thuật được sử dụng trong các mạng nơ-ron sâu để giảm hiện tượng mất mát thông tin và giúp cho việc huấn luyện mạng nơ-ron trở nên dễ dàng hơn. Ý tưởng cơ bản của residual connection là thêm vào đầu ra của một lớp nơ-ron một lượng nhỏ của đầu vào ban đầu của lớp đó. Tức là, thay vì đưa đầu vào của một lớp vào một lớp tiếp theo trực tiếp, chúng ta sẽ thêm nó vào đầu ra của lớp đó thông qua một phép cộng.

$z_{(p,t)}^{(l)}$  cũng chính là giá trị đầu vào của khối self-attention tiếp theo.

### 2.3.6 Class Embedding

Classification token được lấy từ Encoding Block cuối cùng theo công thức 2.8, sau đó được đưa qua một lớp MLP cuối, dùng để phân loại về số lượng lớp tùy vào bài toán.

$$y = LN(z_{(0,0)}^{(L)}) \in R^D \quad (2.8)$$

**Trong đó:**

- **LN()** biểu thị cho hàm LayerNorm tương tự như công thức 2.5.
- **L** là số khối Transformer Block trong lớp Encoder.
- **D** là chiều của embedding vectors.

## 2.4 Các phương pháp Self-attentions

Như đã đề cập, tác giả đã thực nghiệm 5 phương pháp tính toán self-attention khác nhau, bao gồm:

- Space Attention (**S**).
- Joint Space-Time Attention (**ST**).
- Divided Space-Time Attention (**T+S**).
- Sparse Local Global Attention (**L+G**).
- Axial Attention (**T+W+H**).

## 2.4.1 Space Attention (S)

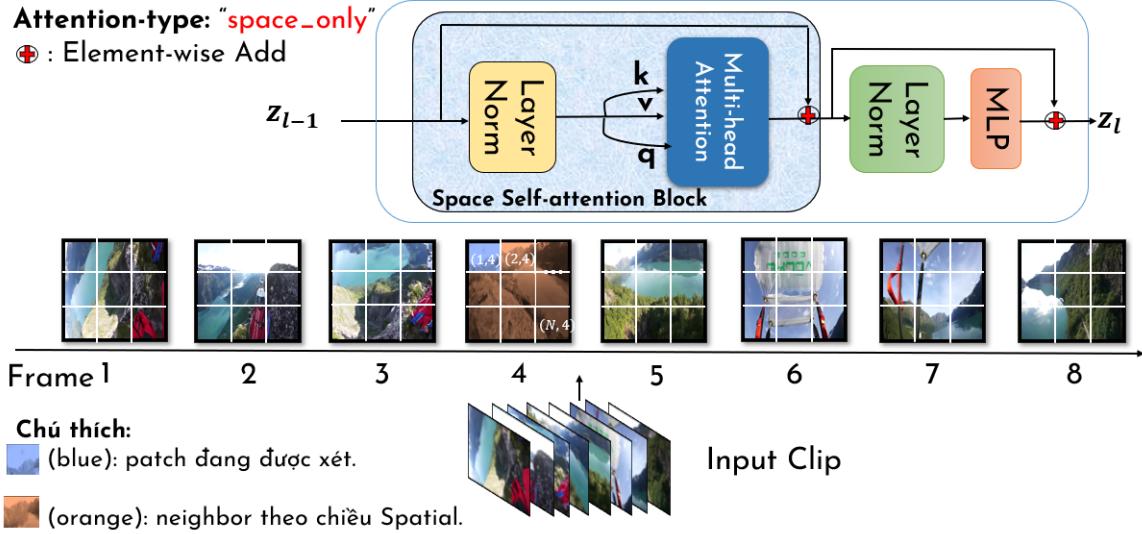
### Self-attention computation

F: 8

N: 196

Attention-type: "space\_only"

⊕ : Element-wise Add



Hình 2.10: Minh họa phương pháp tính self-attention Space Attention, chỉ chú ý đến không gian trong mỗi frame (chiều Spatial).

Thực hiện tính toán self-attention với **query (q)** của patch đang xét so sánh với tất cả **key (k)** của các patch ở cùng frame (chiều Spatial), gọi là neighbors, không quan tâm tới các frame khác trong input clip (chiều Temporal) theo như hình 2.10. Ngoài ra, còn có so sánh **q** của patch đang xét với **k** của classification token theo công thức 2.9.

$$\alpha_{(p,t)}^{(l,a)space} = SM\left(\frac{q_{(p,t)}^{(l,a)}}{\sqrt{D_h}} \cdot [k_{(0,0)}^{(l,a)} \{k_{(p',t)}^{(l,a)}\}_{p'=1,\dots,N}]\right) \quad (2.9)$$

Chú ý rằng công thức 2.9 được rút gọn từ công thức 2.6 bằng cách lược bỏ đi phần tính toán trên chiều Temporal. Ví dụ ở hình 2.10, với patch đang xét đang ở vị trí  $(p, t)$  là  $(1, 4)$  thì công thức sẽ là:

$$\alpha_{(1,4)}^{(l,a)space} = SM\left(\frac{q_{(1,4)}^{(l,a)}}{\sqrt{D_h}} \cdot [k_{(0,0)}^{(l,a)} \{k_{(p',4)}^{(l,a)}\}_{p'=1,\dots,N}]\right) \quad (2.10)$$

Việc thực hiện tính toán self-attention chỉ trên một chiều Spatial giúp giảm bớt đáng kể chi phí tính toán, chỉ có  $N + 1$  ( $N$  patches trong 1 frame và 1

classification token) so sánh query-key được thực hiện so với  $N * F + 1$  so sánh của phương pháp Joint Space-Time Attention sẽ được đề cập ở phần sau.

Cách tính này đã hoạt động tốt trên tập dữ liệu K400, bởi vì trong một nghiên cứu trước đây [14], đã chỉ ra trên tập dữ liệu K400, thông tin không gian quan trọng hơn so với thông tin thời gian để có được độ chính xác cao. Tuy nhiên, dễ dàng nhìn thấy rõ ràng việc tính toán như vậy đã bỏ qua sự phụ thuộc của giữa các frame theo thời gian. Vì vậy, trên hầu hết các thực nghiệm của tác giả, cách tiếp cận này có độ chính xác bị suy giảm so với việc tính toán self-attention trên cả hai chiều Temporal và Spatial, đặc biệt với các tập dữ liệu cần có sự phụ thuộc thời gian mạnh mẽ như tập dữ liệu SSv2.

## 2.4.2 Joint Space-Time Attention (ST)

### Self-attention computation

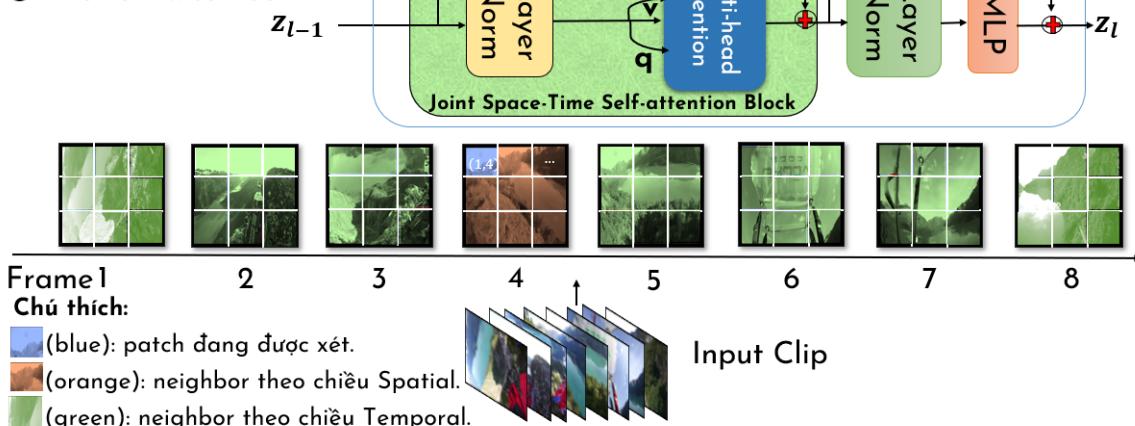
F: 8

N: 196

**Attention-type:**

“joint\_space\_time”

⊕ : Element-wise Add



Hình 2.11: Minh họa phương pháp tính self-attention Joint Space-Time Attention, kết hợp chú ý đến không gian trong mỗi frame (chiều Spatial) và các frame khác trong input clip (chiều Temporal).

Thực hiện tính toán self-attention trên cả hai chiều, cụ thể là với  $q$  của patch đang xét so sánh với tất cả  $k$  của các patch ở cùng frame (chiều Spatial) và tất cả các frame khác ở trong input clip (chiều Temporal) theo hình 2.11. Tương tự với phương pháp S, còn có so sánh  $q$  của patch đang xét với  $k$  của classification token theo công thức 2.6.

Như đã đề cập ở phần trước, phương pháp tính toán self-attention này đã khắc phục được nhược điểm là đảm bảo được sự phụ thuộc thời gian giữa các frame trong input clip. Tuy nhiên, việc thực hiện tính toán self-attention kết hợp cả hai chiều đồng thời khiến cho chi phí tính toán tăng cao, có  $N * F + 1$  ( $N$  patches \*  $F$  frames và 1 classification token) so sánh query-key được thực hiện.

Hơn nữa, phương pháp này cũng dẫn đến chi phí cao hơn đáng kể khi độ phân giải hoặc thời lượng input clip tăng lên. Trên thực tế, tác giả đã đề cập rằng cách tính này sẽ gây ra tình trạng tràn bộ nhớ GPU khi độ phân giải đạt 448 pixel hoặc khi số lượng frame tăng lên 32 và do đó, nó thực sự không thể áp dụng được cho các input clip có độ phân giải cao hoặc input clip dài.

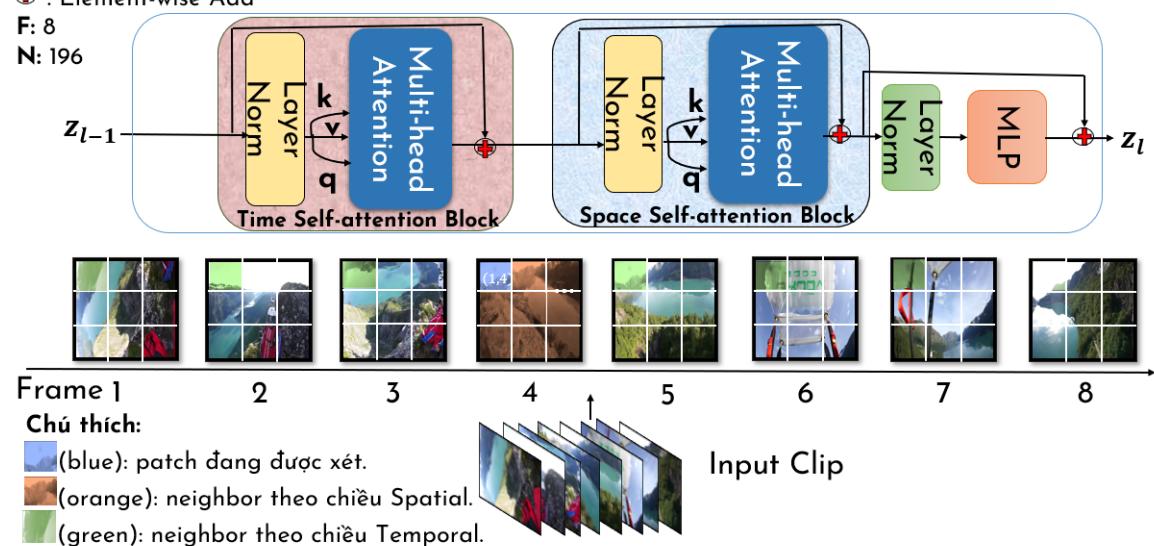
### 2.4.3 Divided Space-Time Attention (T+S)

#### Self-attention computation

Attention-type: "divided\_space\_time"

⊕ : Element-wise Add

$F: 8$   
 $N: 196$



Hình 2.12: Minh họa phương pháp tính self-attention Divided Space-Time Attention, kết hợp chú ý đến không gian trong mỗi frame (chiều Spatial) và các frame khác trong input clip (chiều Temporal), tuy nhiên, tách riêng việc tính toán ở hai chiều này.

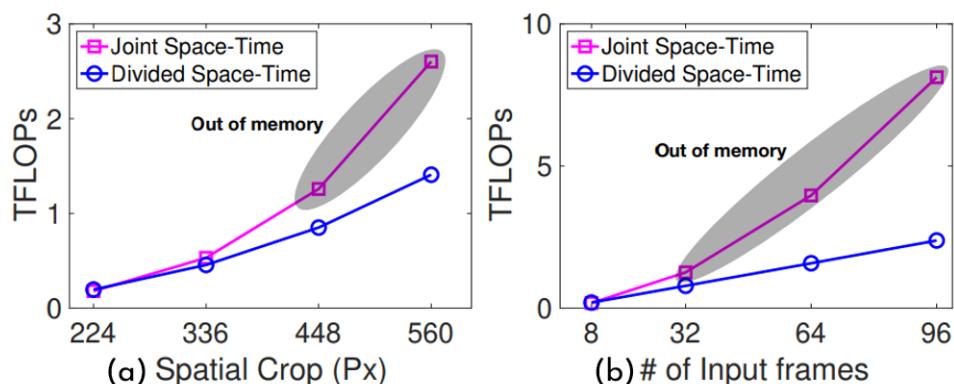
Thực hiện tính toán self-attention trên cả hai chiều tương tự như phương pháp ST. Tuy nhiên, nhìn vào hình 2.12, có thể thấy rằng, khác với phương pháp ST, phương pháp này học từ các ma trận  $q/k/v$  riêng biệt  $\{W_{Qtime}^{(l,a)}, W_{Ktime}^{(l,a)}, W_{Vtime}^{(l,a)}\}$  và  $\{W_{Qspace}^{(l,a)}, W_{Kspace}^{(l,a)}, W_{Vspace}^{(l,a)}\}$  theo hai chiều Temporal và Spatial, đồng thời, phương pháp này giảm bớt đáng kể việc tính toán ở chiều Temporal.

Cụ thể, trong mỗi Transformer Block **I**, trước tiên, thực hiện tính toán self-attention theo chiều Temporal, cụ thể là với  $\mathbf{q}$  của patch đang xét so sánh với tất cả  $\mathbf{k}$  của các patches ở cùng vị trí không gian trên các frame khác bao gồm cả  $\mathbf{k}$  của classification token theo như công thức 2.11.

$$\alpha_{(p,t)}^{(l,a)time} = \text{SM}\left(\frac{\mathbf{q}_{(p,t)}^{(l,a)T}}{\sqrt{D_h}} \cdot [k_{(0,0)}^{(l,a)} \{k_{(p,t')}^{(l,a)}\}_{t'=1,\dots,F}]\right) \quad (2.11)$$

Kết quả ở bước tính toán attention theo chiều Temporal  $z_{(p,t)}'^{(l)time}$  được sử dụng để tính toán attention theo chiều Spatial (công thức 2.9) thay vì chuyển đến hàm **MLP**. Cuối cùng, kết quả sau khi tính toán attention theo chiều Spatial  $z_{(p,t)}'^{(l)space}$  sẽ được đưa vào hàm **MLP** để tính toán giá trị encoding cuối cùng  $z_{(p,t)}^{(l)}$ .

Lưu ý, phương pháp này chỉ thực hiện  $N + F + 2$  ( $N$  patches + 1 classification token ở chiều Spatial và  $F$  frames + 1 ở classification token chiều Temporal) so sánh query-key cho mỗi patch so với  $N * F + 1$  của phương pháp ST.

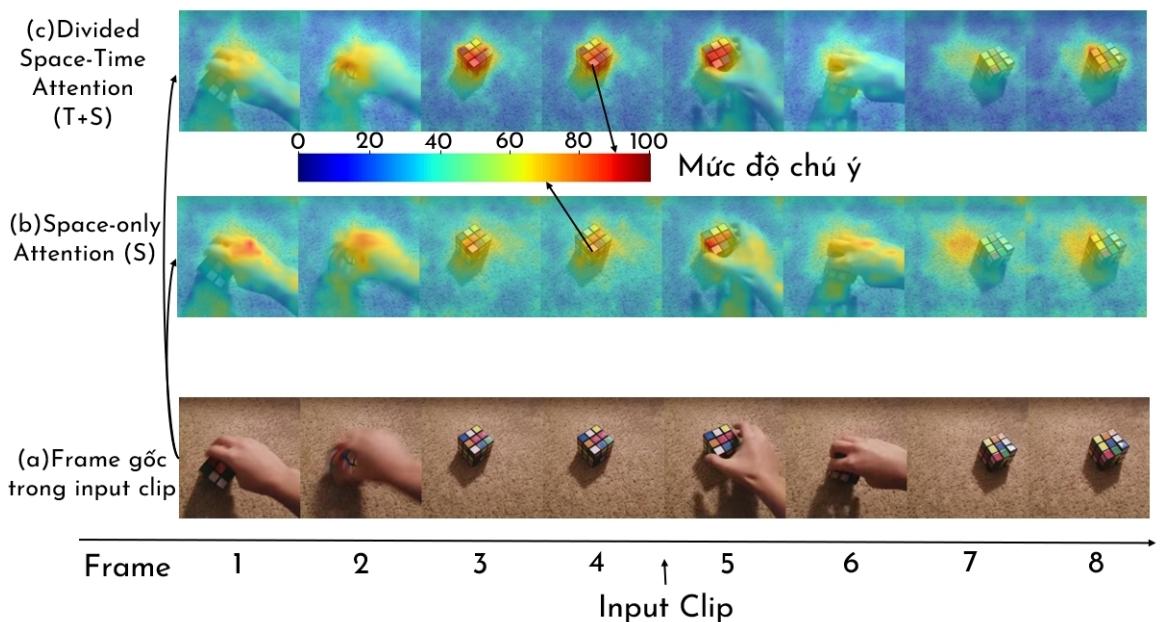


Hình 2.13: So sánh chi phí tính toán giữa phương pháp ST và T+S khi tăng dần độ phân giải ở (a) và số lượng frame đầu vào ở (b) bằng đơn vị TFLOPS (Teraflops - một thước đo quan trọng thể hiện sức mạnh tính toán của GPU).

Trong hình 2.13, tác giả tiến hành so sánh chi phí tính toán của hai phương pháp **ST** và **T+S** trong trường hợp các input clip có độ phân giải cao dần ở hình (a) và dài dần hơn ở hình (b). Các thực nghiệm của tác giả cho thấy rằng phương

pháp **ST** dẫn đến chi phí tính toán cao hơn đáng kể, thậm chí gây ra tình trạng tràn bộ nhớ GPU. Ngược lại, phương pháp **T+S**, mặc dù có số lượng tham số lớn hơn, vẫn hoạt động hiệu quả hơn ở các input clip có độ phân giải cao và dài hơn, không chỉ làm giảm chi phí tính toán mà còn giúp cải thiện độ chính xác của việc phân loại. Vì vậy, tác giả rất khuyến khích sử dụng phương pháp **T+S** cho mô hình Timesformer.

Đặc biệt, hình 2.14 dưới đây là ví dụ chứng minh rằng mô hình có thể học cách chú ý đến các vùng có liên quan trong input clip để thực hiện việc lý luận không gian, thời gian phức tạp. Ngoài ra, hình còn thể hiện cách chú ý khác nhau giữa hai phương pháp **S** và **T+S**.



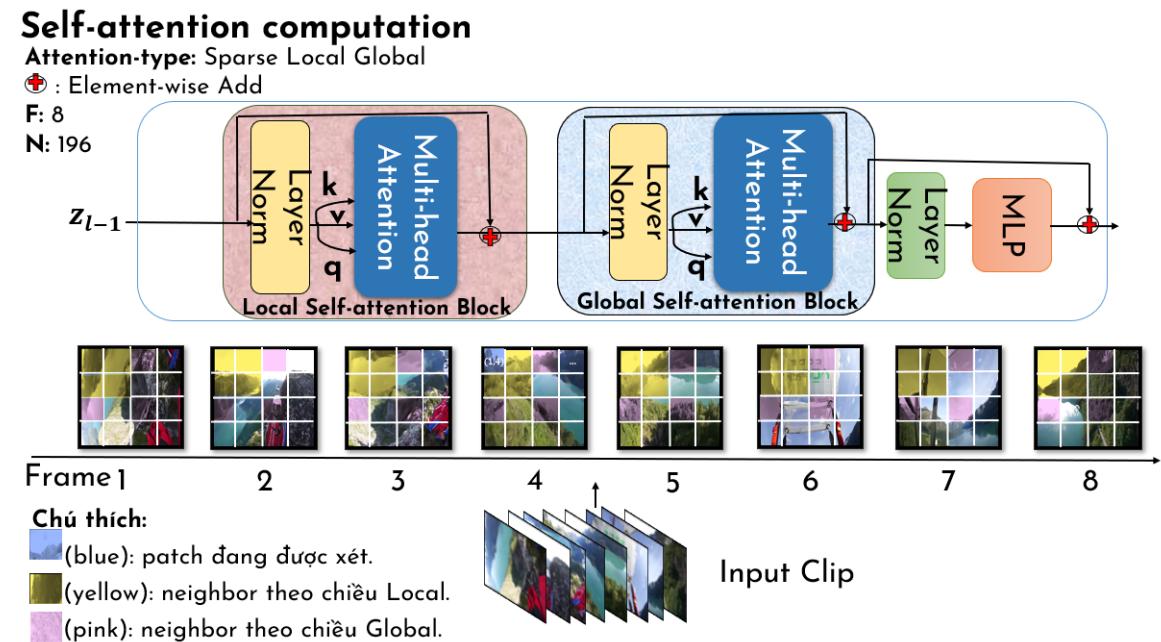
Hình 2.14: Minh họa các vùng chú ý bởi mô hình (theo **Mức độ chú ý**) với hai phương pháp **S** và **T+S** trên cùng một input clip. Trong đó, (a) biểu diễn các frame gốc của input clip, (b) và (c) lần lượt biểu diễn heatmap trên từng frame được tính từ kết quả đầu ra của lớp Encoding cuối cùng theo hai phương pháp **S** và **T+S**.

Từ hình 2.14, có thể thấy được mô hình có thể học cách chú ý đến các vùng có liên quan như khối rubik, bàn tay. Tuy nhiên, sự khác nhau về cơ chế tính toán self-attention giữa hai phương pháp **S** và **T+S** đã dẫn tới vùng chú ý của hai phương pháp này trên một input clip là khác nhau, chúng ta có thể nhìn thấy rõ được sự khác nhau ở (b) và (c). Cụ thể, ở (b), khi có bàn tay xuất hiện trong khung hình (Frame 1, 2, 5, 6), mô hình đã chú ý đến cấu hình hầu hết toàn bộ

bàn tay, trong khi đó, ở (c), mô hình chỉ chú ý đến cấu hình phần các ngón tay của bàn tay, chú ý rằng ở cùng vị trí không gian này ở các frame khác (chiều Temporal) trong (c) chính là vị trí của khối rubik. Từ đó, (c) chứng minh được cơ chế tính toán self-attention của phương pháp **T+S** có sự liên quan giữa các frame trong input clip và có thể hiện tốt hơn trong việc chú ý đến vật thể chính là khối rubik. Ngược lại, (b) cho thấy sự chênh lệch vùng chú ý ở các frame khác nhau khá lớn, chứng minh phương pháp **S** chỉ học trong vùng không gian của frame đang xét, không có sự liên quan đến các frame khác trong input clip.

#### 2.4.4 Sparse Local Global Attention (L+G)

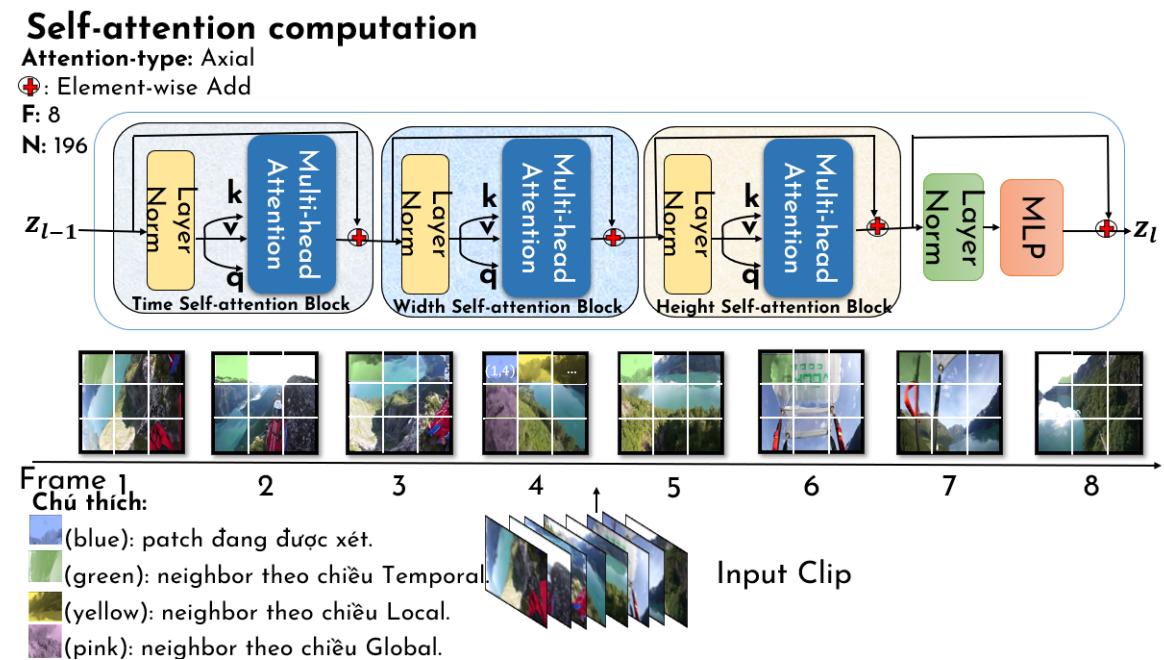
Đầu tiên thực hiện việc tính toán self-attention locally bằng cách thực hiện trên các patch ở liền kề, sao cho kích thước bao quát của local neighbor sẽ bằng  $H/2 \times W/2$  và trải dài tất cả các frame. Sau đó, tính toán globally bằng cách sử dụng stride bằng 2 ở cả hai chiều là chiều rộng và chiều cao, xuyên suốt toàn bộ input clip (hình 2.15).



Hình 2.15: Minh họa phương pháp tính self-attention Sparse Local Global.

## 2.4.5 Axial Attention (T+W+H)

Cuối cùng, phương pháp “Axial Attention” tính toán self-attention theo ba bước riêng biệt: thời gian, chiều rộng, chiều cao. Có thể thấy rằng, phương pháp này tương tự với phương pháp T+S, tuy nhiên giảm bớt ở chiều Spatial còn là cột (trở thành chiều Global) và hàng (trở thành chiều Local) ứng với patch đang xét thay vì là toàn bộ patch của frame (hình 2.16).



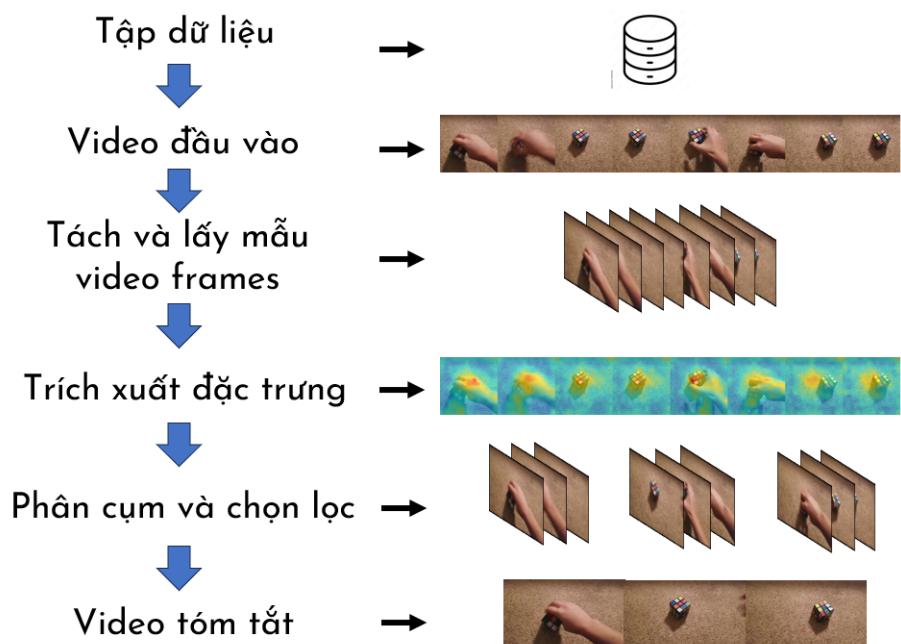
Hình 2.16: Minh họa phương pháp tính self-attention Axial Attention.

## Chương 3

# THÍ NGHIỆM VÀ ĐÁNH GIÁ

### 3.1 Thiết lập thí nghiệm

#### 3.1.1 Tổng quan các bước thí nghiệm và thiết lập tham số



Hình 3.1: Minh họa tổng quan các bước thí nghiệm.

Quá trình thí nghiệm được biểu diễn theo hình 3.1, chi tiết như sau:

- **Đầu tiên**, để thực hiện thí nghiệm, ta cần tìm nguồn dữ liệu là các tập dữ liệu video công khai trên mạng Internet. Bảng 3.1 sau đây biểu diễn một số tập dữ liệu tiềm năng cho bài toán tóm tắt video.

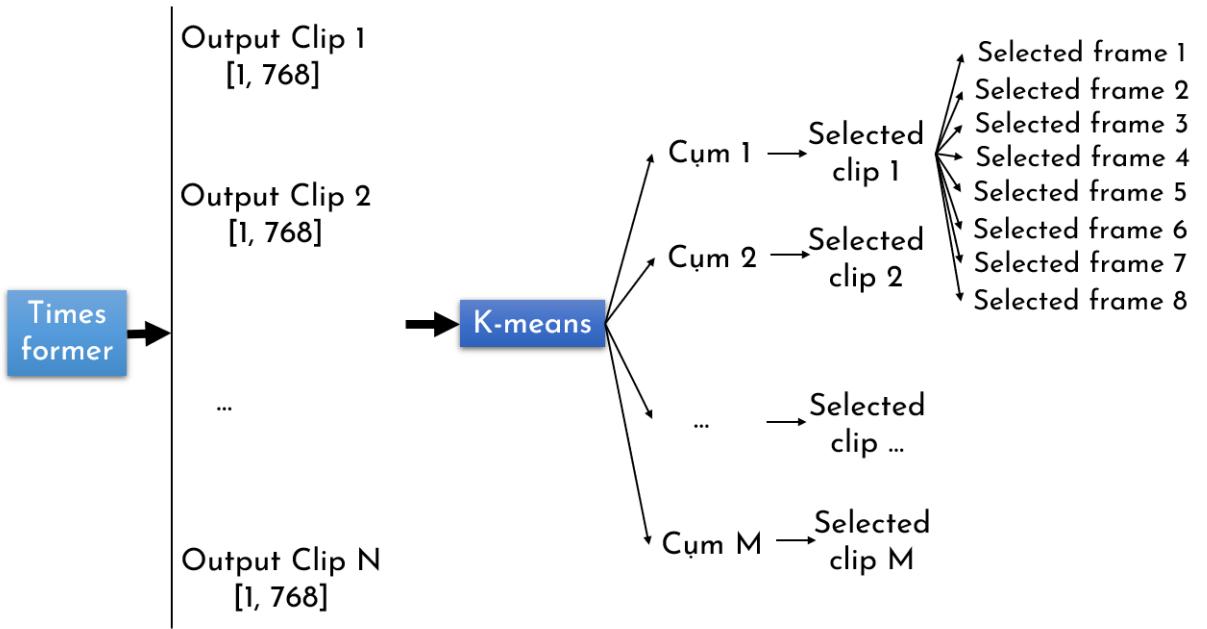
Bảng 3.1: Bảng tổng hợp thông tin một số tập dữ liệu cho bài toán tóm tắt video.

STT	Tên	Số lượng videos	Thời gian	Lĩnh vực
1	SumMe [9]	25	1-6 phút	Ngày lễ, sự kiện và thể thao
2	MED [10]	160	1-5 phút	15 lĩnh vực khác nhau
3	TVSum [15]	50	2-10 phút	Phim tài liệu và video do người dùng tạo
4	UTEgo [11]	4	3-5 giờ	Nấu ăn, lái xe và sinh hoạt

- **Bước 2**, từ tập dữ liệu, lấy ra các video gốc để chuẩn bị cho việc tóm tắt.
- **Bước 3**, tiến hành tách video ra thành danh sách các frame, sau đó, tiến hành quá trình lấy mẫu frame theo chỉ số  $F/F_s$ , gọi là các key frame. Tiếp theo, thực hiện lược bỏ các frame có vị trí lớn hơn số bé hơn gần nhất chia hết cho số frame đầu vào của mô hình (chỉ số  $F$ ). Sở dĩ cần thực hiện điều này là vì mô hình Timesformer sẽ báo lỗi nếu như số lượng frame của input clip không đủ. Ví dụ, với  $F = 8$ ,  $F/F_s = 1/1$ , tổng số frame video gốc là 4492, thì ta có được 4492 key frames, sau đó, cần lược bỏ 4 frames phía sau cùng để đảm bảo đầu vào mô hình.

- **Bước 4**, lần lượt  $F$  key frames sẽ tạo thành input clip và truyền vào mô hình **Timesformer** để trích xuất đặc trưng hình ảnh.
- **Tiếp theo**, sau khi có được đầu ra của mô hình, tôi tiến hành thí nghiệm phân cụm và chọn lọc theo ba thuật toán như sau:

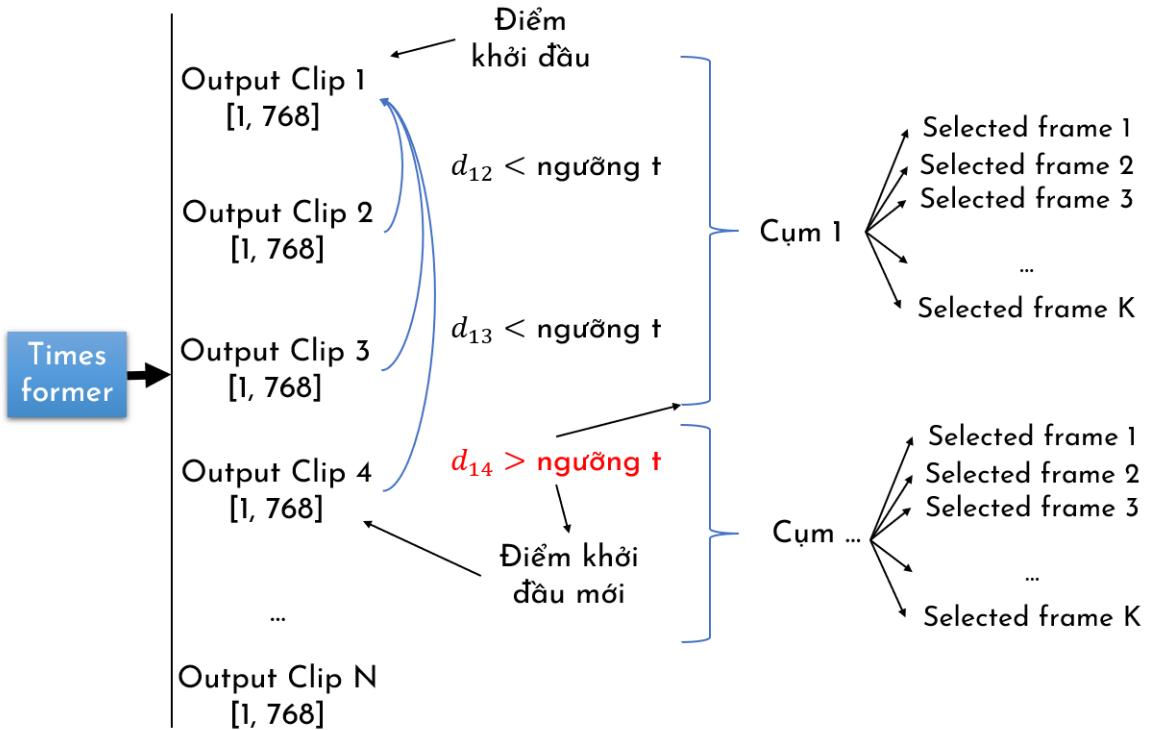
**Thuật toán (1):** Với đầu ra của mô hình là  $[1, 768]$  tương ứng với mỗi clip, sử dụng phương pháp phân cụm **K-means** với  $K$  là số lượng cụm bằng 15 phần trăm tổng số clip được chia. Sau khi thực hiện phân cụm, ta sẽ có được  $K$  cụm với mỗi cụm sẽ bao gồm các clip tương đồng với nhau. Tiếp theo, thực hiện tính khoảng cách giữa mỗi tâm cụm và danh sách clip, tìm ra clip có khoảng cách gần nhất với mỗi tâm cụm. Kết quả ta có được danh sách  $K$  clip được chọn, gọi là selected clip. Cuối cùng, chuyển đổi danh sách selected clip được chọn thành danh sách selected frames. Thuật toán được biểu diễn như hình 3.2 dưới đây:



Hình 3.2: Minh họa thuật toán phân cụm và chọn lọc (1).

**Trong đó:**  $N = \text{Tổng số key frames} / F$ ;  $M = N * 0.15$ .

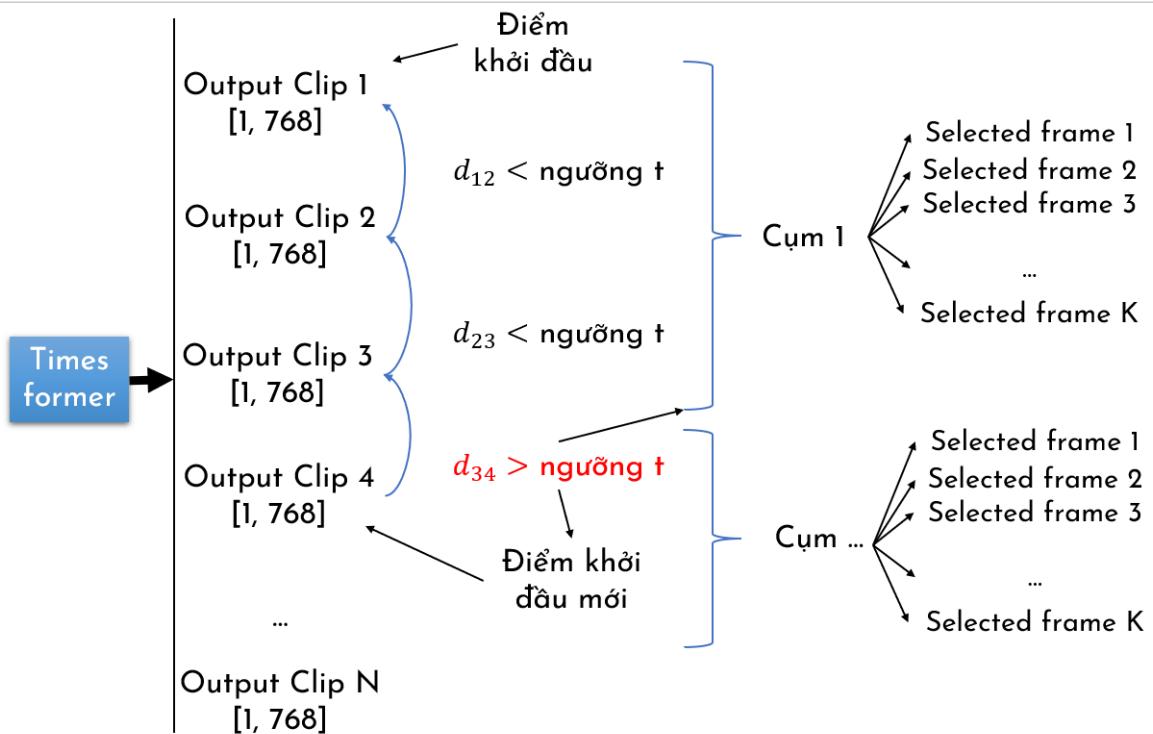
**Thuật toán (2):** Thiết lập clip đầu tiên là điểm khởi đầu, bắt đầu duyệt từ clip thứ 2, thực hiện so sánh sự khác nhau giữa features extracted giữa clip hiện tại và với clip khởi đầu. Nếu như độ khác nhau vượt qua ngưỡng cho phép: tiến hành lưu lại các clip ở điểm khởi đầu tới clip kế trước của clip hiện tại tạo thành một cụm các clip tương đồng và liên tục, sau đó, lấy ngẫu nhiên  $K$  frame id nằm trong mỗi cụm (Với  $K$  là 15 phần trăm tổng số frames của cụm) tạo thành danh sách selected frames, cuối cùng, lưu lại danh sách selected frames và thiết lập điểm khởi đầu mới là clip hiện tại, lặp lại thuật toán cho tới khi vượt qua clip cuối cùng. Thuật toán được biểu diễn như hình 3.3 dưới đây:



Hình 3.3: Minh họa thuật toán phân cụm và chọn lọc (2).

**Trong đó:** ngưỡng  $t = 400$ ;  $K = Số frames cụm tương ứng * 0.15$ ;  
 $N = Tổng số key frames / F$ ;  $d_{ij}$  là khoảng cách **Sum of Squared Difference** (Tổng bình phương chênh lệch) giữa clip i và clip j.

**Thuật toán (3):** Tương tự với thuật toán (2), tuy nhiên, thay vì so sánh clip hiện tại với clip khởi đầu, thuật toán này so sánh clip hiện tại với clip kế trước nó. Thuật toán được biểu diễn như hình 3.4 dưới đây:



Hình 3.4: Minh họa thuật toán phân cụm và chọn lọc (3).

**Trong đó:** ngưỡng  $t = 400$ ;  $K = Sô frames cụm tương ứng * 0.15$ ;  
 $N = Tông số key frames / F$ ;  $d_{ij}$  là khoảng cách **Sum of Squared Difference** (Tổng bình phương chênh lệch) giữa clip i và clip j.

- **Cuối cùng**, thực hiện sắp xếp các selected frames theo thứ tự để đảm bảo tính thời gian của video, sau đó, ghép lại tạo thành bản video tóm tắt.

Tham số trong quá trình thí nghiệm được thiết lập như bảng 3.2 sau đây:

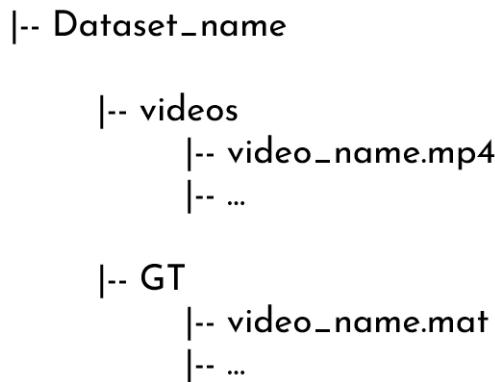
Bảng 3.2: Bảng tổng hợp thiết lập tham số thí nghiệm.

STT	Kí hiệu	Giải thích	Giá trị
1	F/Fs	Tần suất lấy mẫu frames từ video gốc	1/1
2	F	Số lượng frame trong input clip của mô hình	8
3	K	Số cụm K-means	15% tổng số key frames
4	H, W	Chiều cao, chiều rộng khung hình	224
5	P	Chiều cao, chiều rộng của từng patch	16
6	FPS	Tốc độ khung hình	FPS của video gốc
7	D	Chiều của embedding vectors	768
8	L	Số lớp Transformer Encoder	12
9	Attention-type	Phương pháp self-attention	Divided Space-time Attention

### 3.1.2 Dữ liệu thí nghiệm

Mặc dù, nguồn dữ liệu video trên mạng Internet là khổng lồ, việc tìm kiếm một tập dữ liệu phù hợp để có thể thực hiện việc thí nghiệm và đánh giá vẫn rất đang rất khó khăn.

Chỉ có một vài tập dữ liệu có thể sử dụng được việc thí nghiệm và đánh giá, trong đó có tập dữ liệu **SumMe** [9]: bao gồm 25 video có thời lượng tối đa 6 phút, với 25 chủ đề khác nhau, bao gồm thể thao, sự kiện và ngày lễ. Thêm nữa, SumMe có các tập chú thích (annotation) được thu thập từ 15 đến 18 người khác nhau cho mỗi video. Mỗi bản tóm tắt có độ dài từ 5 đến 15 phần trăm thời lượng của video gốc. Cấu trúc dữ liệu của SumMe được thể hiện như hình 3.5.



Hình 3.5: Minh họa cấu trúc tập dữ liệu thí nghiệm.

Trong đó:

- Thư mục **videos** chứa 25 videos gốc (dạng .mp4) thuộc nhiều lĩnh vực khác nhau.
- Thư mục **GT (Ground truth)** chứa 25 files (dạng .mat) chú thích bởi con người của mỗi video. Vì máy tính cá nhân không đủ yêu cầu để tải xuống Matlab, tôi tiến hành chuyển qua các file .mat thành dạng dictionary và có được các thông tin như trong hình 3.6.

```

{
    '__header__': b'MATLAB 5.0 MAT-file',
    'Platform': 'GLNXA64',
    'Created on': 'Mon Mar 3 12:53:15 2014',
    '__version__': '1.0',
    'all_userIDs': array([[array(['ME'], dtype='<U2'),
                           ... ],
                           array([[array([[ 15.46178976, 19.41713132],
                                         [ 56.9928762 , 62.678596 ],
                                         [ 76.05043102, 82.70259638],
                                         [157.403824 , 167.20307529]]),
                           ... ],
                           array([[4494]], dtype=uint16),
                           array([[179.772]]),
                           array([[25]], dtype=uint8),
                           array([[0., 0., 0., ..., 0.]]),
                           array([[0, 0, 0, ..., 0, 0, 0],
                                  [0, 0, 0, ..., 0, 0, 0],
                                  [0, 0, 0, ..., 0, 0, 0],
                                  ..., dtype = uint8)
}

```

Hình 3.6: Minh họa cấu trúc thông tin của file Air\_Force\_One.mat ở dạng dictionary.

### 3.1.3 Cấu hình phần cứng thiết bị thí nghiệm

Vì hạn chế về phần cứng của máy tính cá nhân, tôi tiến hành thí nghiệm hệ thống tóm tắt video trên kernel miễn phí được cung cấp bởi **Kaggle**, chi tiết về thiết bị như sau:

Bảng 3.3: Thông tin thiết bị thí nghiệm.

Tên thiết bị	Kaggle Kernel
Số lượng CPU	4
Bộ xử lí	Intel(R) Xeon(R) CPU @ 2.00GHz
Bộ nhớ	29GB
GPU	2 x Tesla T4
Bộ nhớ lưu trữ	19.5GB

## 3.2 Kết quả thí nghiệm

### 3.2.1 Phương pháp đánh giá

Trong đồ án này, tôi tiến hành việc đánh giá dựa trên thước đo **F1** giữa các bảm tóm tắt được tạo ra từ hệ thống tóm tắt video của tôi với phần chú thích (**annotation**) bởi con người có sẵn trong tập dữ liệu SumMe, được xem là **ground truth**.

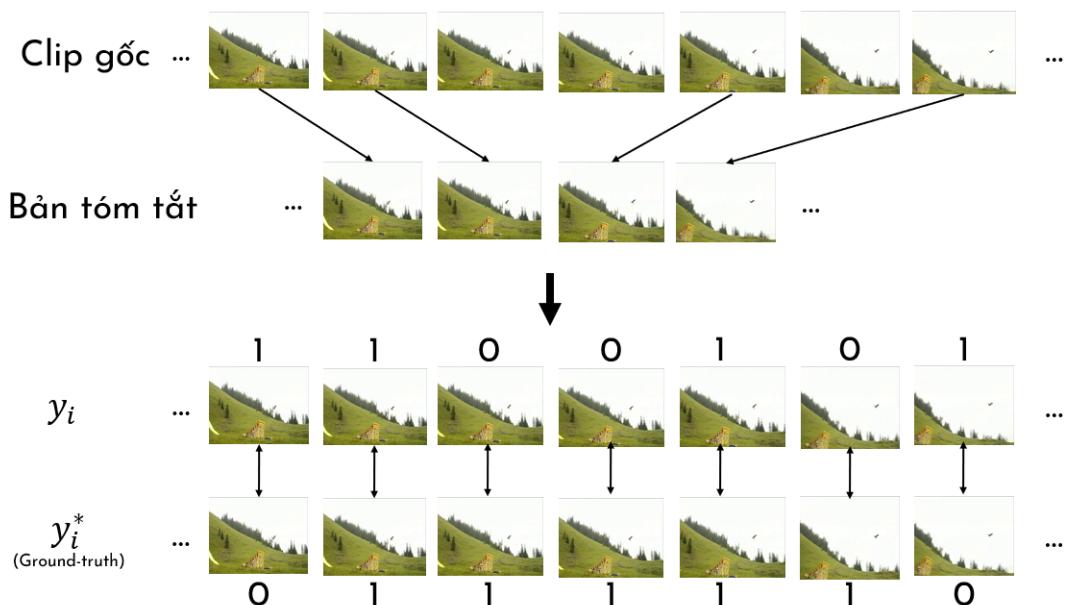
Cụ thể, tôi sử dụng  $y_i \in \{0, 1\}$  thể hiện cho việc khung hình nào từ video gốc được chọn vào bản tóm tắt từ hệ thống tóm tắt video của tôi (tức  $y_i = 1$  nếu khung hình thứ  $i$  được chọn và ngược lại là 0). Ngược lại, tôi sử dụng  $y_i^* \in \{0, 1\}$  với bản tóm tắt được chú thích sẵn trong SumMe (hình 3.7). Như vậy, điểm **F1** được tính toán theo công thức 3.1 sau đây:

$$F1 = \frac{2PRE \cdot REC}{PRE + REC} \quad (3.1)$$

Trong đó, PRE và REC lần lượt là điểm precision và recall, với công thức tính 3.2 và 3.3 (với  $N$  là tổng số lượng frames trong video gốc).

$$PRE = \frac{\sum_{i=1}^N y_i \cdot y_i^*}{\sum_{i=1}^N y_i} \quad (3.2)$$

$$REC = \frac{\sum_{i=1}^N y_i \cdot y_i^*}{\sum_{i=1}^N y_i^*} \quad (3.3)$$



Hình 3.7: Minh họa cách đánh giá điểm F1 trên tập dữ liệu SumMe.

### 3.2.2 Kết quả thí nghiệm

Trong thí nghiệm, điểm **F1** sẽ được tính riêng cho từng bản tóm tắt tham chiếu và điểm **F1** của tập dữ liệu nhận được bằng cách tính trung bình hoặc chọn mức tối đa cho mỗi video, ta được kết quả thí nghiệm theo thuật toán phân cụm và chọn lọc (1) ở bảng 3.4 sau đây:

Bảng 3.4: Bảng thống số kết quả thí nghiệm Timesformer trên tập dữ liệu SumMe theo thuật toán phân cụm và chọn lọc (1).

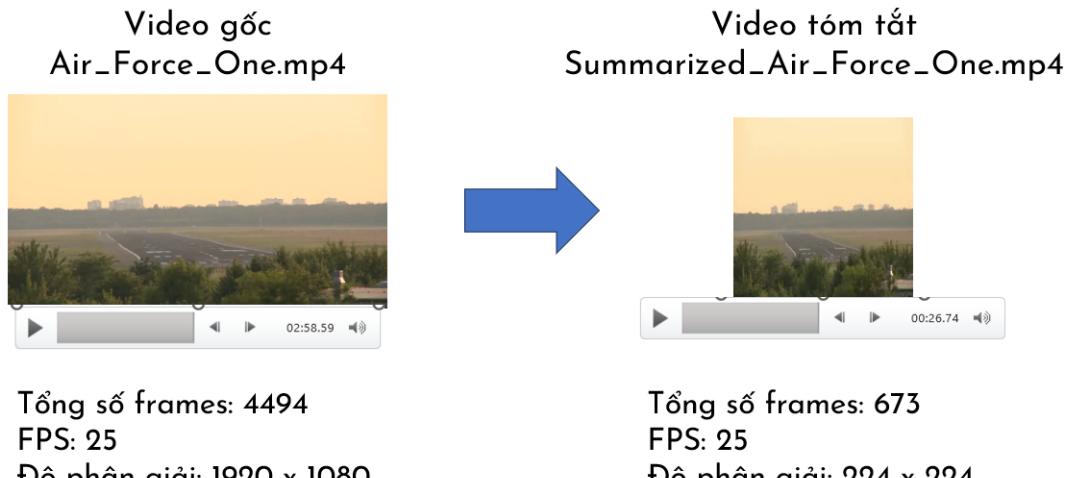
STT	Tên video (.mp4)	FPS	Số lượng frame	Thời gian tóm tắt	Số lượng frame sau tóm tắt	F1-score
1	Air_Force_One	25	4494	2m 32s	673	21.08
2	Base jumping	29.97	4729	2m 37s	709	21.25
3	Bearpark_climbing	25	3341	2m 01s	500	18.92
4	Bike Polo	30	3064	1m 59s	459	23.91
5	Bus_in_Rock_Tunnel	30	5133	2m 57s	769	17.21
6	car_over_camera	29.97	4382	2m 25s	656	29.22
7	Car_railcrossing	29.97	5075	2m 54s	760	15.99
8	Cockpit_Landing	29.97	9046	5m 32s	1356	16.07
9	Cooking	15	1287	1m 06s	192	29.27
10	Eiffel Tower	25	4971	2m 44s	745	20.07
11	Excavators river crossing	25	9721	5m 46s	1458	16.78
12	Fire Domino	30	1612	1m 22s	241	28.51
13	Jumps	25	950	0m 48s	141	17.39
14	Kids_playing_in_leaves	29.97	3187	1m 55s	477	18.00
15	Notre_Dame	24	4608	2m 40s	691	17.45
16	Paintball	23.97	6096	3m 24s	914	18.26
17	paluma_jump	29.97	2574	1m 45s	385	23.37
18	playing_ball	30	3119	1m 52s	466	22.37
19	Playing_on_water_slide	29.97	3065	1m 50s	459	21.57
20	Saving dolphins	29.97	6683	3m 45s	1002	17.23
21	Scuba	30	2221	1m 34s	332	23.73
22	St Maarten Landing	25	1751	1m 12s	261	29.83
23	Statue of Liberty	25	3863	2m 21s	578	22.40
24	Uncut_Evening_Flight	29.97	9672	5m 56s	1450	15.98
25	Valparaiso_Downhill	29.97	5178	3m 01s	776	20.67
26	Trung bình					21.06

Tương tự với cách phân cụm và chọn lọc (1), tôi tiến hành thí nghiệm theo cách phân cụm và chọn lọc (2) và (3) và đạt được kết quả như bảng 3.5.

Bảng 3.5: Bảng thống số kết quả thí nghiệm Timesformer trên tập dữ liệu SumMe theo thuật toán phân cụm và chọn lọc (2) và (3).

STT	Tên video (.mp4)	FPS	Số lượng frame	Thời gian tóm tắt	Số lượng frame sau tóm tắt	F1-score (Thuật toán (2))	F1-score (Thuật toán (3))
1	Air_Force_One	25	4494	1m 45s	673	16.79	16.94
2	Base jumping	29.97	4729	1m 55s	709	17.00	15.75
3	Bearpark_climbing	25	3341	1m 15s	500	17.45	17.02
4	Bike Polo	30	3064	1m 15s	459	15.60	15.94
5	Bus_in_Rock_Tunnel	30	5133	2m 13s	769	16.67	17.28
6	car_over_camera	29.97	4382	1m 40s	656	15.13	14.96
7	Car_railcrossing	29.97	5075	2m 25s	760	16.42	17.02
8	Cockpit_Landing	29.97	9046	4m 50s	1356	15.64	16.22
9	Cooking	15	1287	0m 25s	192	16.62	17.23
10	Eiffel Tower	25	4971	2m 01s	745	15.57	15.88
11	Excavators river crossing	25	9721	5m 01s	1458	15.08	15.68
12	Fire Domino	30	1612	0m 44s	241	17.66	17.87
13	Jumps	25	950	0m 20s	141	18.69	18.06
14	Kids_playing_in_leaves	29.97	3187	1m 15s	477	15.37	17.96
15	Notre_Dame	24	4608	2m 01s	691	15.66	17.38
16	Paintball	23.97	6096	2m 32s	914	15.51	15.57
17	paluma_jump	29.97	2574	1m 01s	385	15.61	16.43
18	playing_ball	30	3119	1m 16s	466	15.73	16.22
19	Playing_on_water_slide	29.97	3065	1m 05s	459	15.42	18.48
20	Saving dolphins	29.97	6683	2m 20s	1002	16.49	16.17
21	Scuba	30	2221	0m 45s	332	18.32	16.59
22	St Maarten Landing	25	1751	0m 41s	261	17.49	16.99
23	Statue of Liberty	25	3863	1m 20s	578	15.99	17.44
24	Uncut_Evening_Flight	29.97	9672	5m 03s	1450	15.37	15.55
25	Valparaiso_Downhill	29.97	5178	2m 12s	776	14.94	15.72
26			Trung bình			16.25	16.66

Hình 3.8 dưới đây minh họa kết quả bản video tóm tắt có được từ một video gốc trong tập **SumMe**.



Hình 3.8: Minh họa kết quả thí nghiệm của video Air\_Force\_One.mp4 trong tập dữ liệu SumMe.

Cuối cùng, tôi so sánh kết quả của ba phương pháp của mình với một số nghiên cứu trước đây để có được cái nhìn tổng quan hơn, kết quả thể hiện ở bảng 3.6:

Bảng 3.6: Bảng so sánh kết quả thí nghiệm với các nghiên cứu trước đây trên tập dữ liệu SumMe.

<b>Phương pháp</b>	<b>F-avg</b>	<b>F-max</b>
CSUV [8]	23.1	-
VS-LMM [7]	-	40.3
dppLSTM [18]	-	43.2
VS-DSF [12]	18.3	-
Summary Transfer [17]	-	41.2
DR-DSN [20]	-	41.3
re-seq2seq [19]	-	45.1
SASUM [16]	-	45.3
Timesformer (Thuật toán (1))	21.06	29.83
Timesformer (Thuật toán (2))	16.25	18.69
Timesformer (Thuật toán (3))	16.66	18.48

### 3.3 Phân tích và thảo luận

**Đầu tiên**, trong thí nghiệm, tôi đã thực hiện việc lấy mẫu 1/1 frame (**F/Fs** bằng 1/1), có thể hiểu là giữ toàn bộ videos gốc để thực hiện việc tóm tắt video. Như vậy, việc đánh giá sẽ đảm bảo được tính chính xác vì số lượng frames video gốc và video tóm tắt là bằng nhau. Tuy nhiên, vấn đề đã nảy sinh khi tôi tiến hành tóm tắt với một số video có độ dài hơn 8 phút (khoảng 13000 frames), phần cứng đã không đáp ứng được các video này. Để giải quyết, tôi tiến hành tăng **F/Fs** lên thành 1/T với  $T = \text{Tổng số frames} / 13000$  để giảm bớt số key frames về trong khoảng dưới 13000. Việc làm này có ưu điểm sẽ giảm gánh nặng bộ nhớ, tuy nhiên, lại mang tới nhiều nhược điểm to lớn, việc lược bỏ một số lượng lớn frame trong video gốc sẽ làm cho việc đánh giá trở nên kém đi, vì có thể một số frame quan trọng sẽ bị lược bỏ. Điều này chứng minh rằng, yêu cầu phần cứng là một điều kiện rất quan trọng trong quá trình triển khai và ứng dụng bài toán vào thực tiễn.

Thêm nữa, từ hình 3.8, ta thấy được độ phân giải video tóm tắt đầu ra trong thí nghiệm của tôi đã giảm đi một cách đáng kể, lí do một lần nữa lại là vì yêu cầu phần ứng không đủ để đáp ứng. Trên thực tế, chúng ta có thể tạo ra các video tóm tắt có độ phân giải giống như video gốc bằng cách lưu danh sách vị trí các frame được chọn, sau đó, ta lấy ra từ video gốc các frame nằm trong danh sách đó. Tuy nhiên, bộ nhớ **Kaggle** kernel đã không đáp ứng được, nếu như thực hiện trên máy tính cá nhân với ổ đĩa lớn thì sẽ khả thi.

**Thứ hai**, về phương pháp phân cụm **K-means**, tôi phát hiện rằng sử dụng hàm KMeans từ thư viện **faiss** sẽ giảm đáng kể thời gian tính toán so với sử dụng hàm KMeans từ thư viện **scikit-learn**. Lí do chủ yếu nhờ vào các tối ưu hóa về phần cứng, cấu trúc dữ liệu, thuật toán, song song hóa và quản lý bộ nhớ của thư viện **faiss**. Bên cạnh đó, tôi thiết lập số lượng cụm được phân ra là 15 phần trăm so với tổng số frame được chọn, lí do là vì trong tập dữ liệu SumMe đã đề cập đến việc đảm bảo tổng số lượng frame của video tóm tắt gần bằng nhau sẽ tạo ra sự nhất quán trong việc đánh giá.

Ngoài ra, kết quả hệ thống tóm tắt video dựa trên **Timesformer** có kết quả chưa tốt so với các nghiên cứu trước đó. Lí do đầu tiên, ba thuật toán tôi thí nghiệm là những phương pháp phân cụm đơn giản, dễ triển khai nhưng mang nhiều hạn chế như nhạy cảm với dữ liệu đầu vào với các outliers (ngoại lệ), .... Lí do thứ hai đến từ việc phân đoạn video, trong thí nghiệm, tôi đã thực hiện việc phân đoạn video đồng đều 8 frames, tuy nhiên, việc phân đoạn như vậy sẽ có hạn chế khi tạo ra các video bao gồm các frame từ 2 cảnh khác nhau, làm cho mô hình Timesformer không thực hiện tốt cấu trúc học thời gian - không gian của mình.

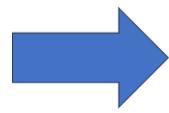
**Thứ ba**, nhìn vào bảng 3.4 và 3.5, ta thấy rõ được thuật toán **(2)** và **(3)** có thời gian xử lí nhanh hơn đáng kể, lí do là việc thay đổi thuật toán phân cụm và chọn lọc đã giúp hệ thống xử lí nhanh hơn các cụm video tương đồng so với phương pháp K-means truyền thống. Tuy nhiên, việc nhanh hơn cũng mang lại sự hạn chế, điểm F1 của thuật toán **(2)** và **(3)** đang thấp so với mặt bằng chung. Lí do lớn nhất cho điều này là việc thực hiện lấy ngẫu nhiên các selected frames, làm cho bài toán đánh giá không quá tốt. Đây cũng sẽ là một hướng phát triển trong tương lai cần chú ý nếu muốn phát triển tiếp đồ án này.

**Thứ tư**, khi so sánh ba thuật toán thí nghiệm, ta cũng thấy được rõ cách tiếp cận khác biệt với bài tóm tắt video của mỗi thuật toán. Cụ thể, đối với thuật toán **(1)**, khi sử dụng K-means, ta cần nắm rõ số lượng cụm chính xác là bao nhiêu, để có được điều này, đương nhiên phải có tổng số frames của video, như vậy, K-means tiếp cận với bài toán tóm tắt video theo hướng tóm tắt các video đã được lưu trữ, có sẵn. Thêm nữa, với độ phức tạp tính toán của K-means, khiến cho việc tóm tắt diễn ra không được quá nhanh, sẽ không phù hợp với hướng tiếp cận còn lại - hướng tiếp cận triển khai trực tiếp (online inference). Đối với hướng tiếp cận online, thuật toán **(2)** và **(3)** cho thấy khả năng tốt hơn, phù hợp hơn. Cụ thể, khi triển khai online, các clip sẽ đến liên tục, hai thuật toán sẽ tiến hành xét các clip mới vào này với clip trước nó xem thử liệu clip mới vào có tương đồng với clip trước không? Nếu có, thì sẽ không lưu trữ clip mới này, ngược lại, tiến hành lưu vào. Như vậy, thuật toán **(2)** và **(3)** sẽ bỏ qua

việc lưu trữ các clip tương đồng với clip trước đó, giúp giảm bớt gánh nặng lưu trữ đáng kể.

**Cuối cùng**, để nhìn nhận khách quan hơn về tính khả thi và ứng dụng của hệ thống tóm tắt video của tôi, tôi thực hiện thử nghiệm trên video mẫu được cung cấp bởi thầy giáo Lê Quang Chiến và đạt được kết quả như hình 3.9 dưới đây. Tuy nhiên, việc minh họa một video tóm tắt bằng hình ảnh là không hiệu quả, xin vui lòng truy cập vào Github đồ án của tôi để thấy rõ hơn.

**Video gốc**  
nhab2\_LAP9.2\_main\_2  
0240502100904.dav



**Video tóm tắt**  
summarized\_lab.mp4



Tổng số frames: 9990  
FPS: 25  
Độ phân giải: 1920 x 1080

Tổng số frames: 1184  
FPS: 25  
Độ phân giải: 224 x 224

Hình 3.9: Minh họa kết quả thí nghiệm trên video mẫu cung cấp bởi thầy Lê Quang Chiến, trích xuất từ camera giám sát phòng học LAP9, dãy nhà B, tầng 2, trường Đại học Khoa Hoc, Đại học Huế.

## Chương 4

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 4.1 Kết luận

Trong quá trình nghiên cứu đề tài "Tìm hiểu Timesformer và ứng dụng cho bài toán tóm tắt video", tôi đã thu được nhiều kết quả quan trọng, mở ra những kiến thức ứng dụng sâu rộng trong lĩnh vực xử lý video. Timesformer, với khả năng vượt trội trong việc phân tích các đặc trưng không gian và thời gian, kết hợp cùng thuật toán K-means, đã chứng minh tính hiệu quả trong việc tạo ra các bản tóm tắt video chất lượng khá tốt. Kết quả cho thấy rằng hệ thống đã phần nào đáp ứng tốt nhu cầu thực tiễn, giúp người dùng tiết kiệm thời gian và nâng cao hiệu suất làm việc.

Việc nghiên cứu và ứng dụng Timesformer cho phép tôi tận dụng tìm hiểu sâu vào cơ chế self-attention mạnh mẽ, mô hình đã học được các mối quan hệ phức tạp trong video một cách tự nhiên. Điều này đặc biệt hữu ích khi xử lý các video có nội dung phong phú và đa dạng, từ đó tạo ra các bản tóm tắt mang tính chính xác và bao quát cao. Vì vậy, Timesformer cũng chứng tỏ được tiềm năng to lớn của mình trong lĩnh vực Video Understanding, mang lại nhiều triển vọng cho lĩnh vực Thị giác máy tính trong tương lai.

Ngoài ra, kết quả thí nghiệm cho thấy rằng việc ứng dụng Timesformer vào bài toán tóm tắt video có những ưu, nhược điểm riêng. Với yêu cầu lượng dữ liệu và tính toán lớn, hệ thống vẫn còn gặp nhiều hạn chế khi áp dụng vào các video thực tế. Tuy nhiên, không thể tranh cãi với lợi ích của bài toán tóm

tắt video mang lại, tôi tin rằng những tiến bộ, cải tiến sẽ sớm được thực hiện và đẩy mạnh, các mô hình sẽ ngày càng đa dạng, hiệu quả, giúp cho bài toán ngày càng sớm đạt được sự tín nhiệm của người dùng.

## 4.2 Hướng phát triển

Tôi tin rằng mô hình Timesformer và bài toán tóm tắt video sẽ ngày càng hoàn thiện và chứng tỏ được công dụng của mình. Vì vậy, có một số hướng phát triển tiếp theo của đồ án như sau:

- **Cải tiến và tối ưu hóa Timesformer:** Timesformer đã chứng minh được hiệu quả vượt trội, tuy nhiên, việc tối ưu hóa thêm nữa có thể mang lại nhiều lợi ích. Các nghiên cứu tiếp theo có thể tập trung vào việc cải tiến kiến trúc mô hình để giảm thiểu tài nguyên tính toán mà vẫn giữ được hiệu suất cao. Việc áp dụng các kỹ thuật như pruning và quantization có thể giúp mô hình trở nên nhẹ nhàng hơn, phù hợp với các hệ thống có tài nguyên hạn chế.
- **Phát triển chất lượng đánh giá bản tóm tắt:** Tìm hiểu thêm các tiêu chí đánh giá tự động có thể giúp đo lường chất lượng của các bản tóm tắt video một cách khách quan và nhanh chóng. Ngoài ra, tìm kiếm và phát triển thêm các tập dữ liệu quy mô lớn, uy tín cao để phục vụ cho việc đánh giá và thí nghiệm.
- **Phát triển phương pháp phân cụm:** Tìm hiểu và thử nghiệm với một số phương pháp phân cụm khác như Gaussian Clustering, DBScan, ....
- **Tìm hiểu thêm các phương pháp phân đoạn video (video segmentation):** Trong quá trình nghiên cứu, tôi đã đọc được bài báo tiềm năng, đáng tìm hiểu có tựa đề "**Rethinking the Evaluation of Video Summaries**" [13], bài báo này có đề cập tới tầm quan trọng của việc chọn phương pháp video segmentation đối với bài toán tóm tắt video, một số phương pháp được nhắc đến như Uniform, One-peak, KTS, ... đang có những kết quả khá tốt.

Tóm lại, tương lai của công nghệ tóm tắt video hứa hẹn sẽ còn nhiều phát triển mạnh mẽ, góp phần nâng cao hiệu suất làm việc và trải nghiệm người dùng trong nhiều lĩnh vực khác nhau. Đề án này, với những kết quả và hướng phát triển đã nêu, hy vọng sẽ đóng góp một phần nhỏ vào sự tiến bộ đó, mở ra nhiều cơ hội nghiên cứu và ứng dụng mới trong tương lai.

# Tài liệu tham khảo

- [1] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? 2021.
- [2] João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. 2017.
- [3] J. Deng, W. Dong, R. Socher, L. J. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. 2009.
- [4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.
- [5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2020.
- [6] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzyńska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something"video database for learning and evaluating visual common sense. 2017.
- [7] M. Gygli, H. Grabner, , and L. Van Gool. Video summarization by learning submodular mixtures of objectives. 2015.

- [8] M. Gygli, H. Grabner, H. Riemenschneider, , and L. van Gool. Creating summaries from user videos. 2014.
- [9] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc van Gool. Creating summaries from user videos. 2014.
- [10] Koji Mineshima Hitomi Yanaka, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. Can neural networks understand monotonicity reasoning? 2019.
- [11] Y. J. Lee, J. Ghosh, , and K. Grauman. Discovering important people and objects for egocentric video summarization. 2012.
- [12] M. Otani, Y. Nakashima, E. Rahtu, J. Heikkila, and `` N. Yokoya. Video summarization using deep semantic features. 2016.
- [13] Mayu Otani, Yuta Nakashima, Esa Rahtu, and Janne Heikkila. Video summarization via semantic attended networks. 2019.
- [14] Laura Sevilla-Lara, Shengxin Zha, Zhicheng Yan, Vedanuj Goswami, Matt Feiszli, and Lorenzo Torresani. Only time can tell: Discovering temporal data for temporal modeling. 2021.
- [15] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. Tvsun: Summarizing web videos using titles. 2015.
- [16] H. Wei, B. Ni, Y. Yan, H. Yu, X. Yang, , and C. Yao. Video summarization via semantic attended networks. 2018.
- [17] K. Zhang, W.-L. Chao, F. Sha, , and K. Grauman. Summary transfer: Exemplar-based subset selection for video summarization. 2016.
- [18] K. Zhang, W.-L. Chao, F. Sha, , and K. Grauman. Video summarization with long short-term memory. 2016.
- [19] K. Zhang, K. Grauman, , and F. Sha. Retrospective encoders for video summarization. 2018.
- [20] K. Zhou, Y. Qiao, , and T. Xiang. Deep reinforcement learning for un-

supervised video summarization with diversityrepresentativeness reward.  
2018.