POLITECNICO DI TORINO

Master's Degree in DATA SCIENCE AND ENGINEERING

Master's Degree Thesis

MARE-Graph

Multimodal Action Recognition in Egocentric video with Graph Neural Networks

Supervisors

Prof. GIUSEPPE BRUNO AVERTA

Dott. SIMONE ALBERTO PEIRONE

Prof.ssa FRANCESCA PISTILLI

Dott. ANTONIO ALLIEGRO

Candidate

DOMENICO MEREU

April 2024

Summary

In recent years, the growth of affordable wearable cameras, exemplified by devices like GoPro, has yielded a growing interest in first-person perspective, denoted as egocentric vision. The proximity of the camera to actions allows for a deep analysis of human behaviour and human-environment interaction. The benefit of egocentric vision finds exploitation in numerous applications, including augmented and mixed reality, human-robot interaction and behavior understanding.

Tasks related to video analysis demand a focus on the integration of diverse modalities due to their inherently multimodal nature. The inclusion of additional modalities provides complementary information, addressing limitations and enhancing the robustness and accuracy of egocentric action recognition systems. Nevertheless, the integration of diverse modalities introduces challenges arising from data heterogeneity, distinct preprocessing needs, and varying computational demands specific to each modality.

Recent studies in egocentric vision have explored graph-based approaches to build hierarchical representations of human activities, or extract topological maps of physical space. Additional research has showcased the adaptability of Graph Neural Networks (GNNs) in the domain of multimodal context. This thesis extends this exploration to leverage Graph Neural Networks (GNN) for action recognition, enhancing temporal reasoning over action sequences and supporting integration and cooperation between different modalities. We combine Graph Neural Networks (GNN) with a cross-modal attention mechanism, enabling reciprocal exploration of content between different modalities and enabling robust cooperation.

To further demonstrate the effectiveness of our approach in exploiting the synergies between the modes, we explore scenarios where a specific modality is not available during test time, attributed to factors like computational constraints or efficiency requirements. Our cross-modal interaction mechanism learns robust representations, showcasing robustness in the face of potential modality loss.

Experiments reveal a significant boost in accuracy compared to various baselines. This underscores the efficiency of GNN in handling multimodal contexts across diverse scenarios.

Acknowledgements

For this thesis, special thanks go to my supervisors who, with great humanity and professionalism, followed me every step of the way, especially Simone and his admirable humility. Thanks to my family for teaching me to live with respect, love and sacrifice. Thanks to my Mami who taught me that love is a revolt, and to my Papi who transmitted to me a passion for learning, a passion without which I would not have reached this little achievement. Thanks to my little brother, a pischello who gives me so much joy and pride. A hug to Giovanni, the best housemate in the world, who also has contributed to my survival. Finally, thanks to all my colleagues in the compañeros group who have made these years special. In particular, to Nico, a brother in breakfast and in life, thanks to the Chinese and his support, to Bulfi, too strong, and to the wonderful Chiaruzzas (blonde and brunette).

"Choose a card"

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent times, there has been a notable surge in the popularity of Egocentric Action Recognition (EAR) and, more broadly, first-person video analysis. This upsurge can be attributed to the widespread availability of lightweight and affordable wearable cameras, such as GoPro and similar devices. A first-person perspective provides a close-up view that allows for in-depth analysis of human behaviors, finding applications in areas where the first-person perspective appears to be useful or necessary, such as human-environment interaction, augmented and virtual reality, as well as in healthcare where egocentric vision systems serve as valuable tools for patient monitoring and surgical training.

Egocentric vision closely mimics the perceptual dynamics of a human observer. A critical aspect of human perception involves the interconnection of various sensory inputs. In contrast, much of the Initial research in the field of computer vision has been oriented towards unimodal investigations. The trend has undergone a shift with the development of multi-stream architectures [1], which have transcended the limitations of unimodal approaches. Two-stream architectures became popular, which employ both RGB and Optical Flow information as input, and a natural extension of them arose, including more branches and different input modalities. Each modality is assumed to be complementary to the rest and, thus, helpful to improve the classification of actions.

Moreover, contemporary research has ventured into the exploration of utilizing graphs for the purpose of action recognition. This encompasses the deployment of hierarchical graphs to represent activities, with each node encapsulating a unique action [2]. Furthermore, investigations have delved into the construction of topological graphs mapping physical space, wherein each node signifies a distinct zone within the video clip [3]. Recent developments in the field have also leveraged graph structures to effectively model interactions between hands and objects [4].

This emerging trend in action recognition research highlights a diverse spectrum of graph-based approaches, showing the versatility of graphical representations in

capturing intricate relationships and contextual nuances in video data.

Within the scope of this thesis, our aim is to show the power and versatility of graphs and Graphical Neural Networks (GNNs) in aggregating information from various modalities in the context of action recognition. By harnessing the capabilities of GNNs, we can effectively encapsulate the temporal dynamics present in video data, enabling a more nuanced understanding of the underlying temporal dependencies that characterise actions.

To highlight the effectiveness of our approach in leveraging modal synergies, we investigate scenarios where specific modalities are unavailable during testing due to factors such as computational constraints or efficiency requirements. Our cross-modal interaction mechanism, reinforced by strategic implementations, demonstrated resilience in learning representations, even in the face of potential modality loss. This step emphasizes the significance of exploring real-world scenarios that test the effectiveness and robustness of a solution.

Our final implementation, MARE-Graph, has yielded a notable performance improvement compared to various baselines, with an average increase of 7.20% in accuracy for action classification.

Chapter 2

Fundamental Topics

2.1 Convolutional Neural Network

The domain of computer vision has undergone a remarkable evolution driven by the assimilation of advanced deep-learning techniques and the progression of hardware technology. The advent of Convolutional Neural Networks (CNNs) has been a pivotal factor, contributing significantly to enhanced performance in models dedicated to image and video tasks. Initially conceived for the processing of 2-dimensional images, such as the versatile VGG [5] architecture, CNNs have proven to possess a versatile nature, extending their applicability to the analysis of diverse data forms, including audio spectrograms. Architectures like VGG employ a series of convolutional and pooling layers in a stacked configuration, while Resnet [6] introduced the concept of residual connections to boost performance. A noteworthy inclusion in this overview is the Inception Network [7], which has notably mitigated computational costs through the incorporation of inception layers. These architectonic frameworks remain at the forefront of ongoing research.

The typical structure of a CNN unfolds through successive layers, encompassing:

- **Fully Connected Layers:** These layers establish connections between every neuron, facilitating comprehensive feature learning across the entire input.
- **Convolutional Layers:** These layers perform the convolution operation, extracting features from the input data through filters.
- **Activation Layer:** Applied after convolutional and fully connected layers, the activation layer introduces non-linearity to the network, enabling it to learn complex patterns and relationships within the data.
- **Pooling Layers:** Employed for down-sampling, pooling layers reduce the spatial dimensions of the input, preserving the essential features.

2.1.1 Fully Connected Layer

Neural networks are a collection of interdependent non-linear functions, with each function embodied by a neuron, also known as a perceptron [8]. In fully connected layers, each neuron conducts a linear transformation on the input vector utilizing a weights matrix. Subsequently, a non-linear activation function f is applied to the resulting product.

Mathematically represented as:

$$f(X, \theta) = W \cdot X + b \tag{2.1}$$

X is a vector input of size D, W is the weight matrix with dimensions D $\times$ O (O being the output dimension), and b is the bias term added to the feature.

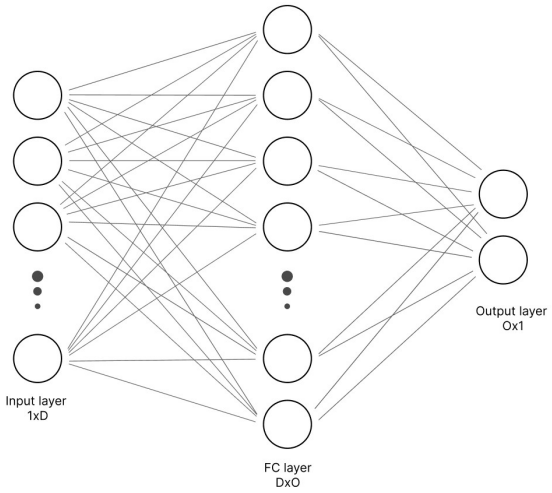


Figure 2.1: Structure of a Feed-Forward Neural Network: Information progresses from the initial layer, traversing various hidden layers, to the output layer.

In the context of classification-based CNNs, the final layers typically consist of one or more FC layers. The final layer's output undergoes a softmax operation to generate probabilities across the various classes. Figure 2.1 illustrates a single FC layer mapping an input to an output.

2.1.2 Convolutional Layer

A convolutional layer within a neural network operates similarly to a sliding dot product, wherein a kernel traverses the input matrix, computing dot products as vectors. The convolutional layer, characteristic of neural networks, exhibits non-uniform connectivity, whereby not all input nodes within a neuron are linked to output nodes, imparting increased adaptability for learning.

This sparse connectivity ensures that not all input nodes exert influence on each output node, thereby optimizing learning efficiency. Moreover, the constrained number of weights per layer proves advantageous for accommodating high-dimensional inputs, particularly in domains involving image data. This unique attribute of convolutional layers contributes to the efficacy of convolutional neural networks (CNNs) [9] in discerning intricate features, such as shapes and textures, within image data, underscoring their aptitude for advanced visual pattern recognition. The mathematical equation for a convolutional layer operation in a neural network can be expressed as follows:

$$Y(i,j) = (X \star K)(i,j) = \sum_m \sum_n X(m,n) \cdot K(i-m, j-n) \tag{2.2}$$

Let X be the input matrix, K be the convolutional kernel, and $*$ represent the convolution operation.

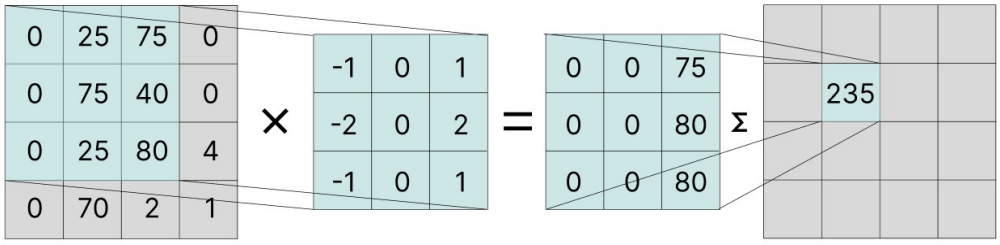


Figure 2.2: 2D Convolution: The elements within the matrix on the left undergo weighting based on the kernel values and are subsequently mapped to the output matrix.

Figure 2.2 illustrates convolutional layer operation.

2.1.3 Activation Layer

An activation layer is incorporated to introduce non-linearity into the produced feature maps. The prevalent activation function employed in CNNs is the Rectified Linear Unit (ReLU), which eliminates negative values in the features by setting them to 0.

Mathematically, ReLU can be expressed as:

$$f(x) = \max(0, x) \tag{2.3}$$

Here, x represents a value within the input feature map. This activation function enhances the network's ability to capture complex patterns and relationships within the data by introducing non-linear transformations.

2.1.4 Pooling Layer

Pooling involves reducing the dimensionality of features by aggregating information through methods like averaging or selecting the maximum value within localized regions. The prevalent technique is Max Pooling, where the maximum value is chosen over a small kernel window, systematically sliding across the feature map to generate a compact representation. Alternative pooling methods, such as averaging, exist, and they can operate over the entire feature map to create a global representation.

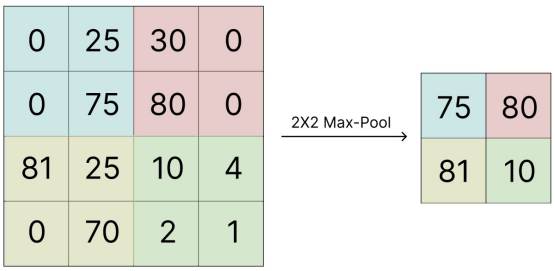


Figure 2.3: Max Pooling: each patch from the input is reduced to a single value using an aggregation function, specifically a max function in this instance.

Refer to Figure 2.3 for an illustration of a sample maxpooling operation on a 2-dimensional matrix using a 2×2 kernel with a slide length of 2.

2.2 Optimization

Optimization plays a crucial role in the landscape of deep learning, where the primary objective is the minimization of a designated loss function. Gradient descent is an iterative process that plays a crucial role in this effort, dynamically adjusting the neural network's weights to iteratively reduce the loss by navigating along the negative gradient. Through the backpropagation mechanism, gradients for each weight in every layer are computed by leveraging the chain rule of differential calculus.

The weights undergo updating in the following manner:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{N} \sum_{i=1}^{N} \nabla L(\theta_t, x_i) \tag{2.4}$$

Here, $L(\theta)$ denotes the loss function concerning the parameters θ, N is the size of the training set, t is the time step, η serves as the learning rate, dictating the convergence speed towards the minimum. This process is referred to as batch gradient descent, where the term "batch" denotes the entire dataset.To address the

challenges posed by extensive datasets, a practical approach involves grouping the data into mini-batches.

The mathematical formulation of is expressed as:

$$\theta_{t+1} = \theta_t - \eta \frac{1}{B} \sum_{i=1}^{B} \nabla L(\theta_t, x_i) \tag{2.5}$$

where B is the size of the mini-batches and θ is updated based on the average gradient over the mini-batches.

2.3 Attention: Self and Cross Attention

The attention mechanism [10], a pivotal algorithm in the field of deep learning, originally devised for enhancing feature representation in key sentence fragments within natural language processing, has found widespread applications in recent years. It has been extensively employed to address computer vision tasks, guiding deep neural networks (DNNs) in focusing on specific image features to enhance the understanding of semantic information in images. Beyond its semantic comprehension capabilities, the attention mechanism proves valuable in feature fusion, visual cue discovery, and temporal information selection-areas. Recently, attention has been extended to explore these areas.

We can distinguish between two types of attention mechanisms:

- **Self-Attention:**

  In the context of self-attention, the mechanism is employed to explore relationships and dependencies within the same set of data. Each element interacts with all other elements in the set, allowing for the assignment of different weights to each element based on its relationships with others. This mechanism is useful for enriching the representation of features belonging to the same modality based on their semantic similarity and temporal relationships.

- **Cross-Attention:**

  Conversely, cross-attention focuses on exploring relationships between two distinct sets. Each element of one set (usually referred to as the query) interacts with all elements of the other set (referred to as key and value). This is particularly valuable for feature fusion belonging to different modalities.

In summary, self-attention enhances relationships within the same set, enabling the assignment of varying weights based on semantic and temporal relationships. On the other hand, cross-attention is instrumental in investigating relationships between distinct sets, facilitating feature fusion across different modalities.

Consider an input sequence $X = [x_1, x_2, \ldots]$ residing in $\mathbb{R}^{N \times d}$, representing a sequence of N elements or tokens. Each input element undergoes embedding, resulting in Z. The embedding Z is subjected to three projection matrices: $W_Q \in \mathbb{R}^{d \times dq}$, $W_K \in \mathbb{R}^{d \times dk}$, and $W_V \in \mathbb{R}^{d \times dv}$, where $dq = dk$. These matrices generate three distinct embeddings, namely Q (Query), K (Key), and V (Value) The formula for calculating the attention output is obtained by multiplying the attention weights by the values:

$$\text{Output} = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.6}$$

Here, the attention operation returns a weighted combination of values V based on the normalized weights calculated using the softmax function, which depends on the relationships between queries Q and keys K.

2.4 Transformer

The Transformer model [10], a paradigm-shifting architecture in deep learning, introduces a groundbreaking approach to sequence-to-sequence tasks without relying on traditional recurrence or convolution mechanisms. At the heart of its innovation lies the concept of attention, a mechanism designed to dynamically weigh the importance of different elements within a sequence. Unlike conventional architectures, the Transformer leverages attention to capture long-range dependencies and relationships between elements, enabling it to excel in tasks such as machine translation and natural language processing. This introduction delves into the Transformer's unique design, emphasizing how attention mechanisms redefine the model's ability to process and contextualize information across sequences.

The Transformer model adopts an encoder-decoder architecture (Fig. 3.10), diverging from conventional methods by eliminating dependencies on recurrence and convolutions for output generation. The encoder, situated on the left side of the Transformer, maps input sequences to continuous representations, which are then passed to the decoder. In turn, the decoder, located on the right side, utilizes the encoder's output and its own preceding output to produce the final sequence. The model operates in an auto-regressive manner, where each step considers previously generated symbols as additional input for generating subsequent ones. The encoder comprises six identical layers, each containing a multi-head self-attention sublayer and a fully connected feed-forward sublayer with ReLU activation. Residual connections and normalization layers follow each sublayer. Notably, positional information is incorporated using positional encodings, as the Transformer lacks inherent awareness of word sequence positions. The decoder mirrors the encoder's structure but introduces a unidirectional nature, focusing only on preceding words

through masking in the multi-head attention mechanism. Positional encodings are also integrated into the decoder's input embeddings to provide positional context.

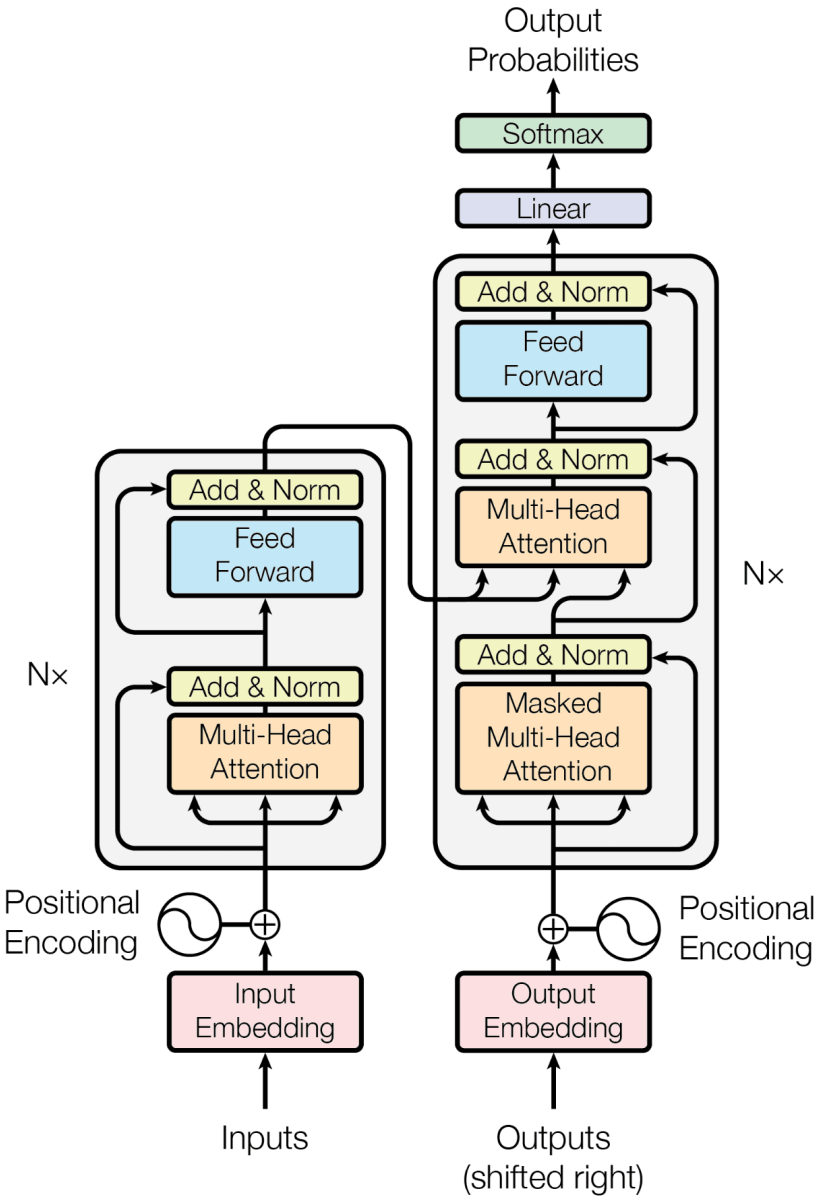


Figure 2.4: The Transformer - model architecture.

2.5 Vision Transformer

A Vision Transformer (ViT) [11] stands as a model designed for image classification and various computer vision tasks, adapting the foundational Transformer architecture initially developed for natural language processing (NLP). ViTs introduce pivotal modifications to the Transformer structure to optimize its performance for image processing. A noteworthy alteration is evident in ViTs' approach to image representation. While Transformer models in NLP represent text as sequential words, ViTs address the unique nature of images by representing them as a sequence of patches, small rectangular regions typically measuring 16x16 pixels. Following the partitioning of the image into patches, each patch is transformed into a vector, capturing its distinctive features. These features are typically extracted using a convolutional neural network (CNN), adept at discerning key attributes crucial for image classification. Subsequently, these vector representations of individual patches undergo processing through a Transformer encoder, a composite of self-attention layers. This design tweak facilitates the ViT's capacity to efficiently analyze image features and nuances, showcasing the model's versatility in handling visual data.

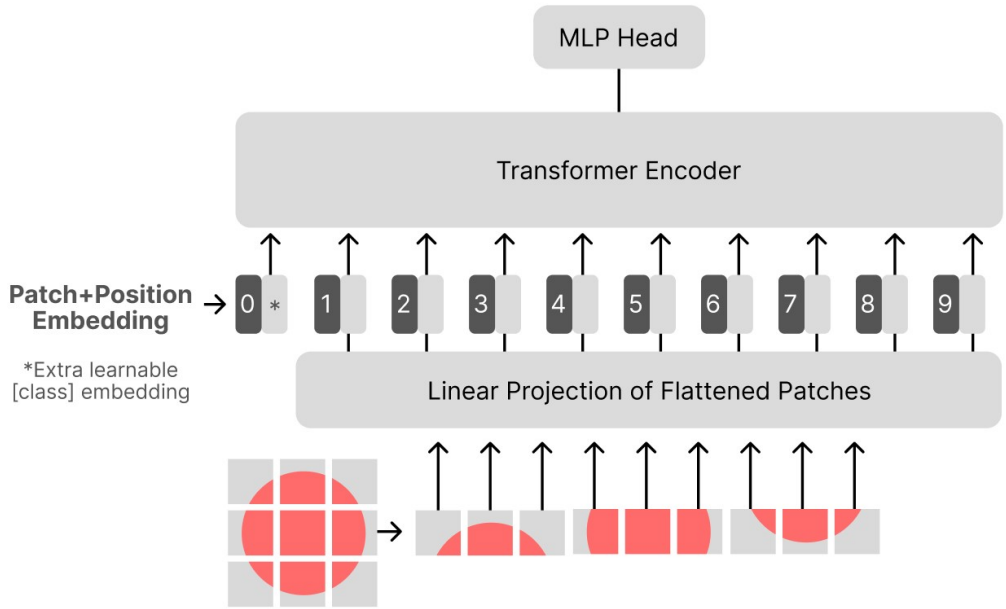


Figure 2.5: Overview of the model: The image is partitioned into patches of fixed size, each of which is linearly embedded, augmented with position embeddings, and then processed through a conventional Transformer encoder. For classification purposes, the standard method involves introducing an additional learnable "classification token" to the sequence

2.6 Graph Neural Networks

Convolutional Neural Networks (CNNs) have been extensively employed for processing data structured in a Euclidean grid, such as images and videos. However, many real-world scenarios involve data characterized by non-Euclidean structures, as seen in social networks or protein structures. Generalizing CNNs to graphs poses challenges, given that convolution and pooling operations are inherently defined on grid-like structures, risking the loss of topological information upon forced adaptation. Graph Neural Networks (GNNs) [12] emerge as deep learning models specifically tailored for processing data with intricate graph-based relationships. The versatility of GNNs and graph modeling finds applications across various domains, including medicine, physics, and social sciences, addressing tasks like node classification, link prediction, and graph classification. This versatility makes GNNs particularly valuable wherever topology plays a crucial role.

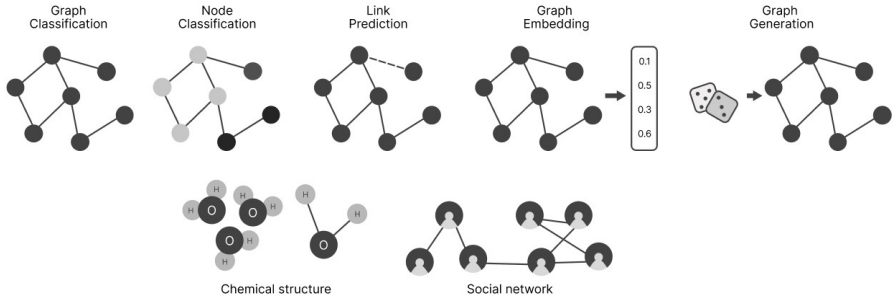


Figure 2.6: Illustration of the possible applications of Graph Neural Networks

In graph theory, a graph $G(V, E)$ is a data structure comprising a set of vertices (nodes) $i \in V$ and a set of edges $e_{ij} \in E$ connecting vertices i and j. The adjacency matrix A captures the connection information, where $e_{ij} = 1$ if nodes i and j are connected and $e_{ij} = 0$ otherwise. Graph Neural Networks (GNNs) are specialized models designed to operate on graphs, treating nodes as entities with distinct properties represented as features.

The GNN operates through a series of steps: Message Passing, Aggregation, and Update.

- **Message Passing:** GNNs learn structural information by considering nodes connected in a graph. The neighborhood N_i of a node i comprises nodes j connected to i by an edge. The GNN engages in Message Passing, where node features of neighbors are transformed and transmitted to the source node.
- **Aggregation:** Transformed messages are aggregated using functions like sum, mean, max, or min. The aggregated messages, denoted as $m_{N(u)}^{(k)}$, are computed based on the transformed node features of neighbors.

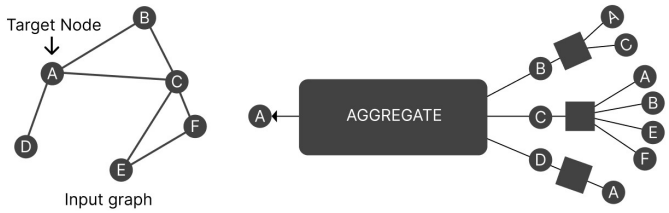


Figure 2.7: The input graph undergoes a sequence of neural network operations, represented by the squares. The new vector representations of neighbouring nodes are aggregated to obtain the new source node information

- **Update:** The GNN layer updates the source node's features using the aggregated messages. This step ensures that the node knows not only about itself but also about its neighbors. The update involves combining the source node's feature vector with the aggregated messages, typically performed through addition or concatenation operations.

In each iteration of message-passing within a Graph Neural Network (GNN), the embedding $h_u^{(k)}$ associated with each node $u \in V$ is updated by assimilating information from the graph neighborhood $N(u)$ of u (Figure 2.6). The message-passing update can be represented as:

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\left(h_u^{(k)}, \text{AGGREGATE}^{(k)}\left(\{h_v^{(k)}, \forall v \in N(u)\}\right)\right) \tag{4.2}$$

This can be further expressed as:

$$h_u^{(k+1)} = \text{UPDATE}^{(k)}\left(h_u^{(k)}, m_{N(u)}^{(k)}\right) \tag{4.3}$$

where $\text{UPDATE}^{(k)}$ and $\text{AGGREGATE}^{(k)}$ are arbitrary differentiable functions (i.e., neural networks), and $m_{N(u)}^{(k)}$ is the "message" aggregated from the graph neighborhood $N(u)$ of u. In each iteration, the AGGREGATE function takes the set of embeddings of nodes in the graph neighborhood $N(u)$ and generates a message $m_{N(u)}^{(k)}$ based on this aggregated neighborhood information. The UPDATE function then combines this message $m_{N(u)}^{(k)}$ with the previous embedding $h_u^{(k)}$ of node u to produce the updated embedding. The initial embeddings at $k = 0$ are set to the input features for all nodes, i.e., $h_u^{(0)} = x_u$, for all $u \in V$. After performing K iterations of GNN message passing, the output of the final layer is used to define the embeddings for each node:

$$z_u = h_u^{(K)}, \quad \text{for all } u \in V \tag{4.4}$$

It is noteworthy that GNNs defined in this manner are inherently permutation-equivariant due to the set-based nature of the AGGREGATE function.

2.6.1 Graph Convolutional Networks

One of the prevalent baseline models in the realm of graph neural networks is the Graph Convolutional Network (GCN)[13], which adopts the Kipf normalized aggregation along with the self-loop update methodology. The GCN model formulates the message-passing function as:

$$h_v^{(k)} = \sigma\left(W^{(k)} \sum_{v \in N(v) \cup \{v\}} \frac{h_v}{\sqrt{|N(u)||N(v)|}}\right)$$

In this formulation, the Kipf normalized aggregation function is utilized to gather information from the neighbors of a node, including the node itself (self-loop). The resulting embedding of node v at the k-th iteration, $h_v^{(k)}$, is thus determined by the weighted sum of the embeddings of its neighbors, with weights given by the weight matrix W and normalized. This iterative update process represents the message-passing mechanism across the graph, incorporating neighborhood information into the representation of the current node.

2.6.2 SAGEConv

GraphSAGE [14] constitutes a comprehensive framework designed for acquiring node embeddings through a systematic learning process. The model's efficiency in training is facilitated by an inductive approach that involves neighborhood sampling and aggregation. In this methodology, the sampling operation is responsible for the random selection of neighbors for each node within the graph. Subsequently, an aggregation step is executed, wherein the features of the sampled neighbors are harnessed to construct node embeddings. This process iterates from the top layer down to the final layer, ultimately producing the output embedding. This resultant embedding is then employed for updating the model's weights and can be applied to fulfill specific tasks within diverse applications.

The paper introduces three candidate aggregation functions, each designed to capture and consolidate information from neighboring vectors. The first is the mean aggregator, which computes the elementwise mean of neighboring vectors. The second, the LSTM aggregator, employs a more complex LSTM [15] architecture for information integration. Lastly, the pooling aggregator utilizes a fully-connected neural network followed by max-pooling to effectively aggregate and distill relevant information.

The expression for SAGEConv is as follows:

$$x_i' = W_1 x_i + W_2 \cdot \text{aggregation}_{j \in N(i)} x_j \tag{2.7}$$

Chapter 3

Related Works

3.1 Action Recognition from Videos

Human Activity Recognition (HAR) is a pivotal aspect of computer vision. The evolution of HAR can be traced from initial reliance on RGB data, with handcrafted feature-based approaches dominating the early stages. The advent of deep learning ushered in a new era, marked by neural network-based methods, such as two-stream Convolutional Neural Network (CNN) [1], Recurrent Neural Network (RNN) [16], 3D CNN [17], and Transformer-based approaches [18]. In recent years, the integration of diverse sensor modalities has garnered attention. The concept of multi-modality, leveraging complementary characteristics between different data modalities, has further improved action recognition accuracy. However, challenges arise in data acquisition, feature extraction, fusion, and temporal synchronization. The paradigm of three-dimensional networks has become a cornerstone in capturing intricate spatiotemporal relationships, exemplified by the Inflated 3-dimensional CNN (I3D) introduced by [17]. This innovative architecture strategically processes sequential frames, offering a nuanced understanding of video features across both temporal and spatial dimensions. To address computational efficiency, channel-separated convolutions have been introduced, providing an effective means to reduce computational costs. The SlowFast [19] approach further enhances video processing by adopting a two-stream methodology, leveraging both dense and sparse sampling strategies. This approach proves instrumental in capturing both temporal dynamics and spatial semantics. Embracing the complexity of multi-stream networks, TSN [20] pioneers temporal aggregation by segmenting action clips into multiple segments. Akin to TSN, the Temporal Binding Network (TBN) [21] employs a comparable approach, introducing asynchronous data sampling for a more comprehensive understanding of video dynamics.

3.2 Multi-modal Learning

Methods in multi-modal learning involve the integration of information from diverse sources, aiming to assess the viability of such fusion strategies. Despite demonstrating promise in enhancing model performance, these techniques, as evidenced by studies [22], are not without drawbacks. A thorough investigation into the comparison between single-modal and multi-modal training, as conducted by [23] in the context of classification tasks, reveals notable disparities in the overfitting dynamics across different modalities. Specifically, multi-modal networks exhibit a tendency to overfit at an accelerated pace due to their heightened complexity.

3.2.1 Problem definition

Multisensory perception entails the interaction of diverse modalities, such as audio and video. The fundamental objective is the mapping of features belonging to different modalities within a common categorical set Z. To achieve this goal, various unimodal networks can be employed, and the structure of the multimodal network can be formalized as $M = N_1 \bigoplus N_2$, where the symbol $\bigoplus$ denotes the fusion operation, and N_1 and N_2 represent two networks in the case of two modalities. The multimodal network M learns a shared representation by integrating information from different modalities. The process of inter and intra-modal learning involves representing objects from distinct perspectives, leveraging synergies among the involved modalities. Monomodal representation entails mapping individual input streams to a high-level semantic representation. Conversely, multimodal representation harnesses the correlational power of each modality through the aggregation of features. Multimodal fusion strategies play a crucial role in exploiting synergies within multimodal data, facilitating a more comprehensive understanding of the context. This approach overcomes limitations associated with individual modalities, promoting a deeper and integrated understanding of the examined phenomenon.

3.2.2 Conventional methods

Conventional methodologies encompass a diverse array of fusion techniques, ranging from early to hybrid schemes. In the Early Fusion paradigm, low-level features directly extracted from each modality undergo amalgamation before the classification stage. Late Fusion entails the independent classification of features derived from distinct modalities, which are subsequently fused. Hybrid Fusion integrates multimodal features derived from both early and late fusion stages in preparation for the conclusive decision-making step. Fig. 3.1 graphically illustrates the distinction among the three approaches.

- **Early Fusion (Feature-Level Fusion):** Early Fusion facilitates the amalgamation of low-level features extracted from various modalities, offering a unified and comprehensive information repository. The extracted features exhibit inherent heterogeneity attributed to the diverse nature of modalities and discrepancies in appearance. However, this fusion process, while providing a holistic representation, introduces the potential risk of yielding a singular, comprehensive representation that may lead to prediction errors.
- **Late Fusion Techniques:** Late Fusion techniques encompass methodologies such as majority voting and low-rank multimodal fusion. These methods are employed to aggregate final prediction scores derived independently from each modality. In this approach, each modality autonomously contributes to decision-making, potentially posing challenges to the overall integration performance.
- **Intermediate Fusion:** Intermediate Fusion involves the spatial amalgamation of representations obtained at varying scales and dimensions from diverse data streams. The resulting representations exhibit disparities, introducing complexities during the merging process and presenting challenges that need to be addressed for effective integration.

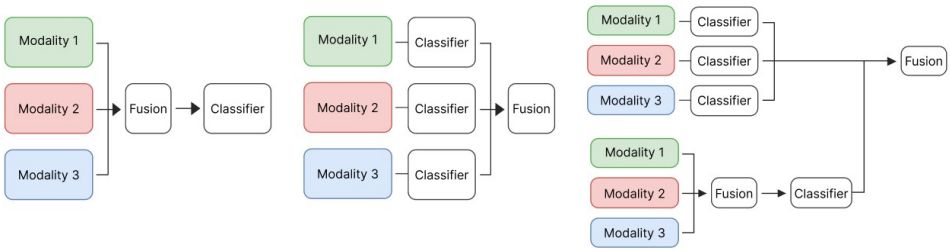


Figure 3.1: Conventional multimodal fusion methods. Left: Early Fusion (Feature-Level Fusion), center : Late Fusion, Right: Intermediate Fusion

3.2.3 Attention mechanism based

In accordance with the taxonomy provided by [24], methodologies grounded in attention mechanisms can be classified into distinct categories: early summation, early concatenation, hierarchical attention (multi-stream to one stream), hierarchical attention (one stream to multi-stream), cross-attention, and cross-attention to concatenation. Given X_A and X_B from two arbitrary modalities, $Z_{(A)}$ and $Z_{(B)}$ represent the respective embeddings of tokens. The resulting token embedding sequence, denoted as Z, is generated through multimodal interactions. The Transformer layers/block is denoted as $Tf(*)$. Each modality is elucidated as follows:

- **Early Summation**: This approach entails the weighted summation of token representations derived from disparate modalities prior to the classification process. Early summation involves the judiciously weighted amalgamation of initial features originating from diverse modalities, with manually specified weights. Formally:

$$Z = \mathrm{T}f(\alpha Z_{(A)} \oplus \beta Z_{(B)}) = \mathrm{MHSA}(Q_{(AB)}, K_{(AB)}, V_{(AB)})$$

  where $\bigoplus$ denotes element-wise sum, and α and β are weightings.

- **Early Concatenation**: Within this modality, sequences of token representations from various modalities are concatenated and subsequently input into Transformer layers.

$$Z = Tf([Z_{(A)}; Z_{(B)}])$$

  Early concatenation affords the consideration of all multimodal token positions as a unified sequence, thereby enhancing the contextual encoding of each modality.

- **Hierarchical Attention (Multi-stream to One Stream)**: This approach encompasses the utilization of independent Transformer layers to encode multimodal inputs, followed by the concatenation and fusion of their outputs through another Transformer.

$$Z = Tf_3([Tf_1(Z_{(A)}); Tf_2(Z_{(B)})])$$

  Hierarchical attention (multi-stream to one stream) serves as an instantiation of late interaction/fusion.

- **Hierarchical Attention (One Stream to Multi-stream)**: Exemplified by InterBERT [25], this hierarchical attention modality involves encoding concatenated multimodal inputs through a shared single-stream Transformer, succeeded by two distinct Transformer streams.

$$[Z_{(A)}; Z_{(B)}] = Tf_1([Z_{(A)}; Z_{(B)}]) \quad Z_{(A)} = Tf_2(Z_{(A)}) \quad Z_{(B)} = Tf_3(Z_{(B)})$$

  This method preserves cross-modal interactions while concurrently upholding the autonomy of unimodal representations.

- **Cross-Attention**: In instances where two-stream Transformers are employed, the embeddings of Query (Q) are exchanged in a cross-stream fashion to discern cross-modal interactions.

$$\begin{cases} Z_{(A)} = \text{MHSA}(Q_B, K_A, V_A) \\ Z_{(B)} = \text{MHSA}(Q_A, K_B, V_B) \end{cases}$$

  Cross-attention facilitates the discernment of cross-modal interactions, nonetheless, it may exhibit a limitation in considering the global cross-modal context.

- **Cross-Attention to Concatenation**: The concatenation of two cross-attention streams is processed by another Transformer to model the global context.

$$\begin{cases} Z_{(A)} = \text{MHSA}(Q_B, K_A, V_A), \\ Z_{(B)} = \text{MHSA}(Q_A, K_B, V_B), \\ Z = Tf([Z_{(A)}; Z_{(B)}]). \end{cases}$$

  The delineation of these methodologies, as expounded above, furnishes a comprehensive framework encapsulating diverse attention-based multimodal interaction modalities, each characterized by distinct attributes and challenges.

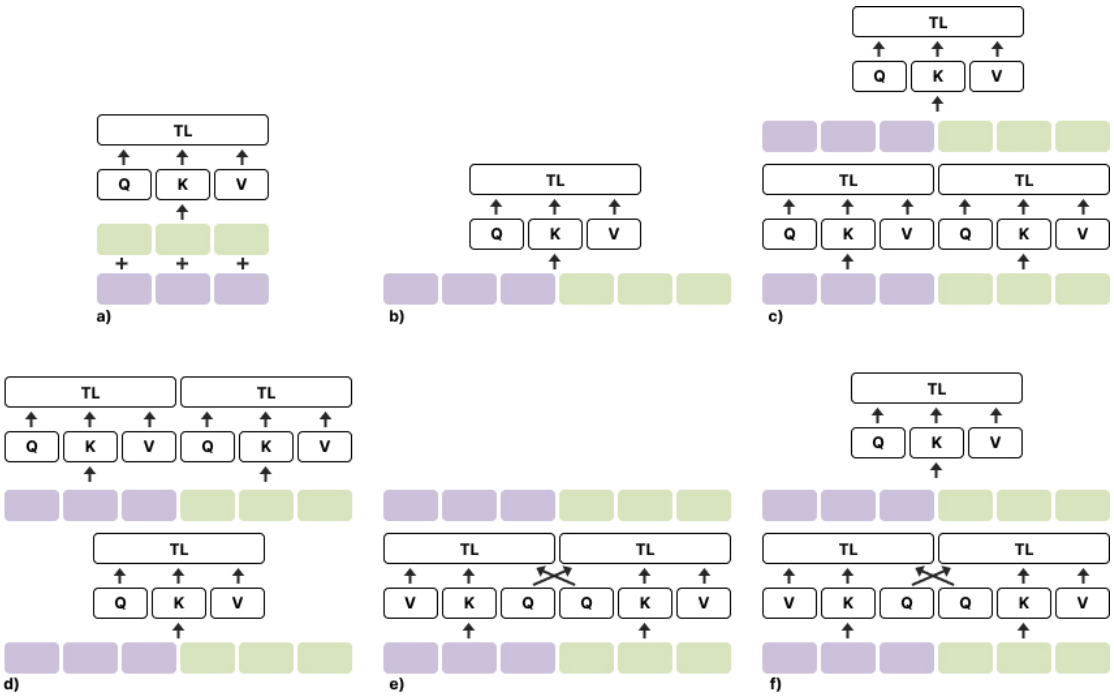


Figure 3.2: Attention mechanism based multimodal fusion methods: (a) Early Summation, (b) Early Concatenation, (c) Hierarchical Attention (from multi-stream to one-stream), (d) Hierarchical Attention (from one-stream to multi-stream), (e) Cross-Attention, and (f) Cross-Attention to Concatenation. In this context, "Q" represents the Query embedding, "K" denotes the Key embedding, and "V" stands for the Value embedding. The term "TL" refers to the Transformer Layer.

3.3 Graph-Based Action Recognition

Researchers have investigated the application of Graph Neural Networks (GNNs) in various video understanding tasks, including action recognition, temporal action localization, and video object segmentation. GNNs are employed to perform spatial and semantic relation reasoning over time among visual components, considering the critical role of spatio-temporal relations in video understanding. [26] provides a comprehensive overview on graph-based action recognition.

In the context of Video Action Recognition, Wang and Gupta [27] propose a graph-based approach for capturing long-range temporal contexts. They construct a space-time region graph, connecting humans and objects across frames, and utilize Graph Convolution Networks (GCNs) for action prediction. Ou et al. [28] enhance the spatial-temporal graph by introducing actor-centric object-level graphs and relation-level graphs. Zhang et al. [29] propose multi-scale reasoning over the temporal graph, employing Graph Attention Networks (GAT) and a learnable adjacency matrix. Gao et al. [30] extend GCN-based relation modeling to zero-shot action recognition, leveraging knowledge graphs. Zhao et al. [31] introduce graph-based higher-order relation modeling for long-term action recognition, utilizing GNNs to aggregate information from multiple graphs.

In Skeleton-Based Action Recognition, Yan et al. [32] propose an ST-GCN network that connects joints in a human skeleton based on natural connectivity. Shi et al. [33] introduce a fully-connected graph with learnable edge weights, while Li et al. [34] connect physically-apart skeleton joints. Zhao et al. [35] improve joint connectivity and use GCNs to capture relations between joints. Shi et al. [36] maintain edge features in addition to node features via directed graph convolution. Liu et al. [37] capture long-range spatial-temporal dependencies using dilated windows and multi-scale GCNs.

Various modifications to standard graph convolution layers have been explored. Si et al. [38] incorporate GCNs into LSTM units, while Cheng et al. [39] propose shift graph convolution networks inspired by shift CNNs. Cheng et al. [40] decouple aggregation in GCNs by splitting channels into multiple groups.

These advancements in GNN-based approaches contribute to improved performance in video understanding tasks, demonstrating the effectiveness of modeling spatio-temporal relationships through graph-based reasoning.

3.3.1 Skeleton Based Action Recognition

Skeletal information plays a crucial role in the recognition of human actions, as the spatio-temporal relationships among different parts of the human body provide insights into motion and action patterns. The approach of skeleton-based action recognition involves identifying human actions from a sequence of skeletal data extracted from a video. Recognizing the inherent graph structure of the human skeleton with natural joint connections, Yan et al. [32] propose an ST-GCN network. This network initially establishes connections between joints in the same frame based on natural connectivity and then extends these connections to the same joints in two consecutive frames to preserve temporal information. The application of Graph Convolution Networks (GCNs) on the joint graph facilitates the learning of both spatial and temporal action patterns, resulting in a significant improvement over previous methods. Shi et al. [33] enhance the approach by introducing a fully-connected graph with learnable edge weights between joints and a data-dependent graph derived from the input skeleton. GCNs are applied to all three graphs. In contrast to the conventional updating of only node features in GCNs, Shi et al. maintain edge features and concurrently learn both node and edge feature representations through directed graph convolution. Departing from previous approaches that focus on local spatial and temporal contexts in consecutive frames, Liu et al. [37] capture long-range spatial-temporal dependencies. This involves constructing multiple dilated windows over the temporal dimension, running separate GCNs on multiple graphs with varying scales within each window, and aggregating the results to capture multi-scale and long-range dependencies. Some studies have explored modifications to standard graph convolution layers for improved adaptation to action recognition. Shi et al. integrate GCNs into LSTM units, incorporating GCN-based updates for gates, hidden state, and cell memory. Inspired by shift CNNs [41], Cheng et al. [39] propose shift graph convolution networks, introducing multiple feature partitions at a joint and updating each partition using features from the corresponding neighboring joint.

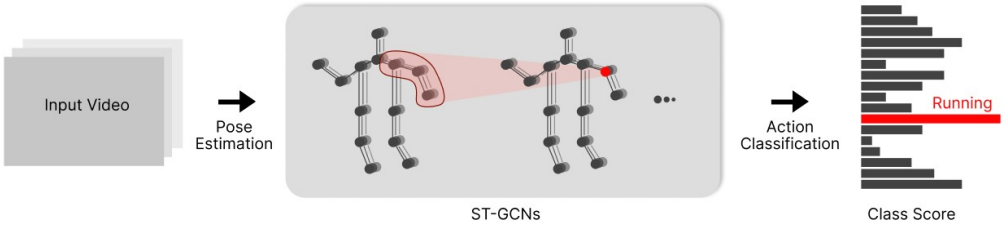


Figure 3.3: ST-GCN: Spatial-temporal graph is constructed based on skeleton sequences. Successive layers of spatial-temporal graph convolution (ST-GCN) are systematically applied.

3.3.2 Graph Based Temporal Reasoning

In the realm of video analysis, temporal reasoning entails the capability to identify and understand not only the temporal sequence of events but also the temporal dependencies between different actions. Traditional methodologies, grounded in temporal classifiers applied to low-level video features like I3D [17] features, have historically included sliding window approaches [42], segmental models [43], and recurrent networks [44]. However, these conventional approaches confront challenges in effectively capturing long-range action patterns and adapting to scenarios characterized by limited observations, as exemplified in a first-person video scenario extracted from the EPIC-Kitchens dataset [45]. To surmount these challenges, Graph-based Temporal Reasoning Module (GTRM) [46], leveraging the power of Graph Convolutional Networks (GCNs). The GTRM is strategically integrated with existing models to adeptly learn temporal relations among actions, facilitating the explicit modeling of interdependencies between adjacent actions and refining the outcomes. Evaluations conducted on demanding datasets, such as EGTEA [47] and EPIC-Kitchens [45], underscore the enhancement brought about by the GTRM, particularly in scenarios characterized by limited observations and protracted video durations, where backbone models, especially those employing recurrent networks, witness performance improvements.

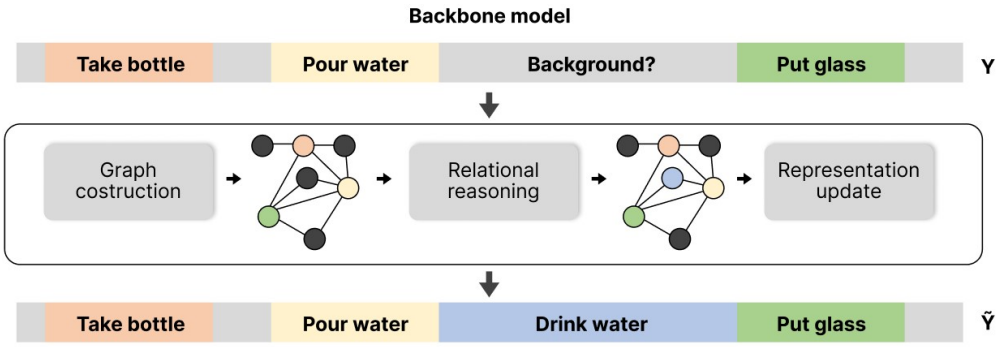


Figure 3.4: The backbone model identifies the poured water segment as background. With GTRM addition, it successfully detects this as "drink water" by learning temporal relations between actions and adjusting segment boundaries based on multiple actions.

3.3.3 Object-Relation Reasoning Graph

Current methodologies primarily utilize either object-level graphs or scene graphs to depict the dynamics of objects and their relationships in videos. However, they often neglect the direct modeling of intricate transitions in relationships. Ou et al. [28] proposes an innovative framework known as the Object-Relation Reasoning Graph (OR^2G) to address the intricacies of action reasoning in video sequences. By amalgamating both an object-level graph (Fig. 3.5), and a relation-level graph (Fig. 3.6), the OR^2G adeptly captures transitions in object attributes while concurrently reasoning about the dynamic shifts in relationships between objects. Additionally, the study introduces a Graph Aggregating Module (GAM), incorporating a multi-head edge-to-node message passing operation. The GAM plays a crucial role in channeling information from relation nodes back to object nodes, thereby augmenting the synergy between the object-level graph and the relation-level graph.

- **Object-level Graph Reasoning:** To enhance the understanding of objects, two distinct high-level features are utilized. Firstly, the visual feature v_i^O is extracted through ResNet [6]. Simultaneously, the semantic feature s_i^O is derived by embedding the object category into a semantic feature space. Given that actions hinge on the action subject, an actor-centric object-level graph is devised, positioning the person node as its central element.

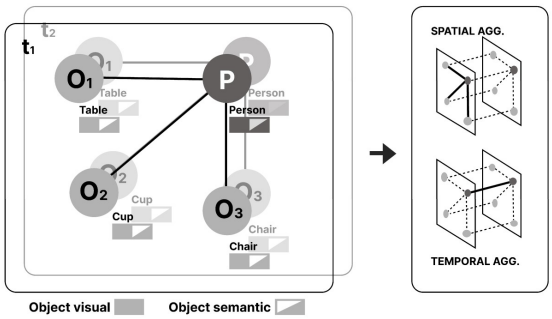


Figure 3.5: Object-level Graph: visual and semantic features are extracted from the object locations to construct the nodes in the graph. The node features of the graph are refined at the object-level with spatial and temporal aggregations

- **Relation-level Graph Reasoning:** The nodes within the relation-level graph play the role of capturing the intricate connections between the subject and objects, denoted as relation nodes. Firstly, the spatial feature characterizes the spatial relationships between the subject and objects through relative spatial position descriptors. The second feature is the semantic feature derived

from embedding the visual relationship category between the subject and objects into the semantic feature space.

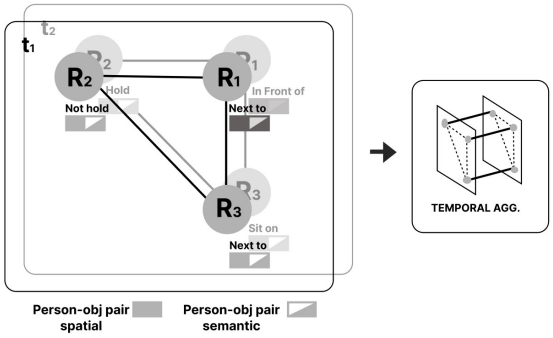


Figure 3.6: Relation-level Graph: spatial relationships between the subject and objects and semantic feature construct the nodes in the graph The node features of the graph are refined with temporal aggregation

- **Graph Aggregating Module:** The object-level graph is responsible for transferring and processing information related to the object nodes, whereas the relation-level graph handles the information of the relation nodes. To enhance their coupling, a graph aggregating module featuring a multi-head attention edge-to-node message passing operation has been introduced. This mechanism facilitates the feedback of information from the relation-level graph to the object-level graph, specifically in the spatial dimension. The formulation for updating the person node in each frame is expressed as:

$$G_p^A = \frac{1}{d_p} \sum_{i=1}^{h} \text{head}_i(G_p^R, G_p^R, G_p^R) W_A$$

where G_p^A represents the updated person node, G_p^R is the feature matrix of all the edges connected to node p, d_p is a normalization factor, $head_i$ denotes the attention mechanism. The central node, along with the updated object nodes and the resultant output from the object-level graph, are combined and processed to yield the comprehensive output of the graph aggregating module.

3.4 Multi-stream Architectures

3.4.1 Two-Stream Inflated 3D ConvNet (I3D)

A pivotal advancement in the evolution of deep learning involves the ability to extend 2D Convolutional Neural Networks (ConvNets) into 3D, leveraging knowledge gained from pretraining on extensive datasets such as ImageNet. This process entails taking a 2D architecture and transforming all filters and pooling kernels from their typical square form to cubic, where N × N filters become N × N × N. To achieve this transformation, the authors of [17] propose a method known as inflating 2D ConvNets into 3D. Leveraging the linearity property, the weights of the 2D filters are duplicated N times along the time dimension and then rescaled by dividing them by N. By implementing this approach, the outputs of pointwise non-linearity layers, as well as average and max-pooling layers, remain consistent with the 2D case. This technique effectively allows for the bootstrapping of 3D filters from their 2D counterparts, enabling the seamless integration of temporal information into the neural network architecture. To capture detailed information pertaining to motion, the authors propose a methodological approach involving the independent training of an I3D network dedicated to RGB inputs and another specialized for optical flow inputs. During the testing phase, predictions generated by these networks are subjected to averaging.

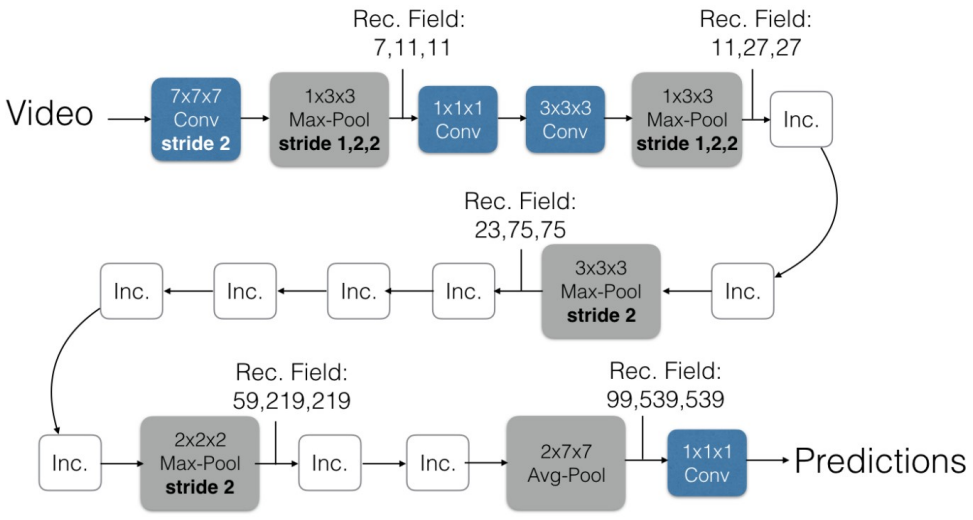


Figure 3.7: The Inflated Inception-V1 architecture

3.4.2 Temporal Segment Network (TSN)

Within the context of video analysis, the Temporal Segment Network (TSN) [20] emerges as a innovative solution designed to address a critical limitation present in two-stream Convolutional Neural Networks (ConvNets) [1]. TSN operates within a distinct video-level framework, specifically developed to overcome the challenges associated with modeling prolonged temporal structures, a task for which traditional ConvNets exhibit shortcomings. In response to this limitation, TSN introduces a novel approach to effective temporal modeling. This video-level framework operates on a sequence of sparsely sampled short snippets extracted from the entirety of the video under analysis. What sets TSN apart is its incorporation of both spatial stream ConvNets and temporal stream ConvNets to process these short snippets. Each snippet independently generates a preliminary prediction for action classes, and a consensus is subsequently derived among these predictions to formulate a comprehensive video-level prediction. In the mathematical formulation of the Temporal Segment Network (TSN), consider a video V partitioned into K segments of equal duration, denoted as $S_1, S_2, ..., S_K$. The TSN function is defined as:

$$TSN(T_1, T_2, ..., T_K) = H\left(G\left(F(T_1; W), F(T_2; W), ..., F(T_K; W)\right)\right)$$

Here, $(T_1, T_2, ..., T_K)$ represents the sequence of snippets each randomly sampled from its corresponding segment S_k, and $F(T_k; W)$ is the function representing a ConvNet with parameters W operating on the short snippet Tk, producing class scores for all classes. The segmental consensus function G combines outputs from various short snippets to derive a consensus on class hypotheses among them. Based on this consensus, the prediction function H estimates the probability distribution for each action class across the entire video.

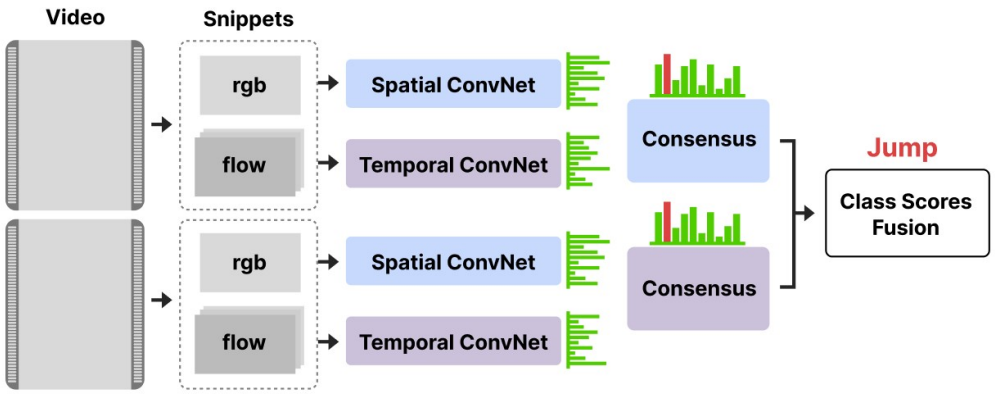


Figure 3.8: Temporal Segment Network, spatial stream ConvNets and temporal stream ConvNets process short sparsely sampled snippets

3.4.3 Temporal Binding Network (TBN)

The Temporal Binding Network (TBN) [21] introduces a novel approach to multimodal fusion, focusing on the integration of modalities within specified Temporal Binding Windows (TBWs). In this paradigm, modalities (m_1, m_2) are fused with diverse temporal offsets, all confined within a predetermined time window of width ±b. This process is formalized as

$$y = h\left(G\left(f_{tbw}(m_{1j}, m_{2k})\right)\right) \quad k \in \left[\left\lceil \frac{jr_2}{r_1} - b \right\rceil, \left\lceil \frac{jr_2}{r_1} + b \right\rceil\right]$$

f_{tbw} denotes a multimodal feature extractor operating within the TBW, r_i is the modality's framerate and j the time step. Now, contrasting TBN with an extended version of the Temporal Segment Network (TSN) [20], two key distinctions become evident. Firstly, TSN relies on independent temporal aggregation of each modality across segments, with the modalities only combined through late fusion. Unlike TSN, TBN enables the advantageous combination of modalities within a segment, enhancing its temporal integration capabilities. Secondly, in TSN, modalities are trained independently, and their predictions are combined during inference. Conversely, TBN adopts a simultaneous training approach, where all modalities are trained collectively, and their combination is learned jointly.

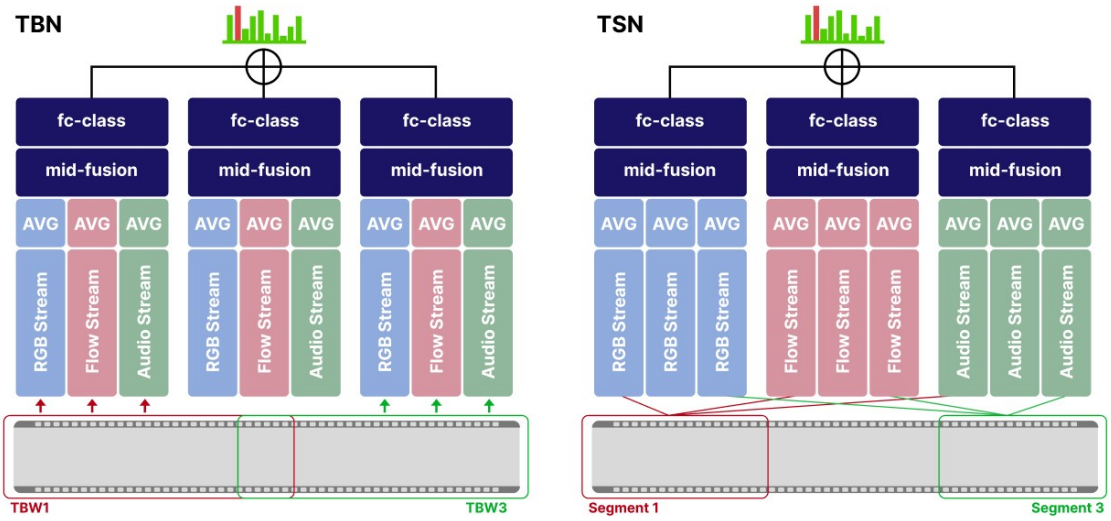


Figure 3.9: Temporal Binding Network (left): all modalities are trained collectively, and their combination is learned jointly within specified Temporal Binding Windows (TBWs).

3.4.4 Temporal Relational Reasoning (TRN)

Temporal Relation Reasoning Network (TRN) [48] employ a similar approach to Temporal Segment Networks (TSN) [20] by passing video snippets through a 2D Convolutional Neural Network (CNN) until reaching the pre-classification layer. However, in contrast to directly obtaining class confidence scores, TRN focuses on generating features at the segment level. To facilitate modeling temporal relationships between segments, a modified relational module, particularly sensitive to item ordering, processes these segment-level features. Once the relational features are computed, they undergo summation and are then input to a classification layer for the final prediction. This approach enhances the network's ability to capture inter-segment temporal dependencies and offers flexibility through the multi-scale variant. Drawing inspiration from the relational reasoning module employed in visual question answering, the pairwise temporal relation is introduced as a composite function denoted by $T_2(V)$:

$$T_2(V) = h_\phi \left(\sum_{i<j} g_\theta(f_i, f_j) \right)$$

Here, the input comprises the video V with n selected ordered frames, represented as $V = \{f_1, f_2, ..., f_n\}$, where f_i is the representation of the i-th frame, for instance, the output activation from a standard CNN. The functions h_ϕ and g_θ are employed to fuse features from different ordered frames, implemented through multilayer perceptrons (MLP) with parameters ϕ and θ respectively. Expanding on the concept of 2-frame temporal relations, higher frame relations are defined, including the 3-frame relation function denoted by $T_3(V)$:

$$T3(V) = h'_\phi \left(\sum_{i<j<k} g'_\theta(f_i, f_j, f_k) \right)$$

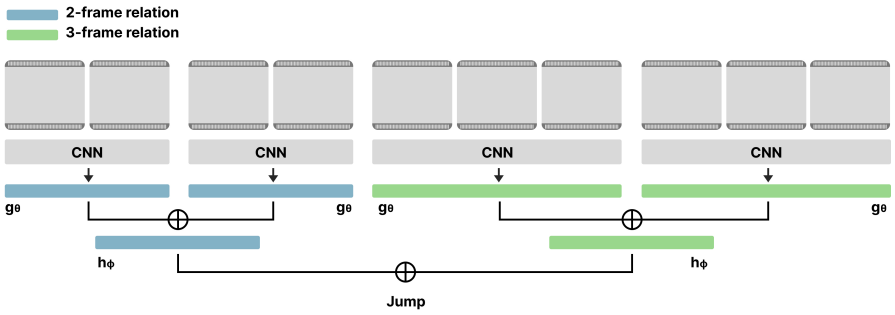


Figure 3.10: TRN: Frames are sampled and fed into different relation modules

3.4.5 SlowFast

The SlowFast [19] network architecture is designed as a unified framework operating at dual framerates. This architecture incorporates distinct pathways, namely the Slow pathway and the Fast pathway, fused through lateral connections to form the comprehensive SlowFast network.

In the Slow pathway, a convolutional model processes a spatiotemporal volume representing a video clip. Notably, the key feature of the Slow pathway is a substantial temporal stride (τ) on input frames, meaning it processes only one out of τ frames. This temporal downsampling results in a slower refreshing speed, and a typical value studied is $\tau = 16$, corresponding to roughly 2 frames sampled per second for 30-fps videos. The Slow pathway thus samples a subset of frames denoted as $T \times \tau$, where T represents the number of frames sampled.

Simultaneously, the Fast pathway operates in parallel, acting as another convolutional model with distinct characteristics. The Fast pathway aims for a high frame rate, working with a smaller temporal stride (τ/α), where $\alpha > 1$ signifies the frame rate ratio between the Fast and Slow pathways. This pathway samples αT frames, α times denser than the Slow pathway, providing a fine representation along the temporal dimension. Importantly, the Fast pathway maintains high temporal resolution features throughout the network hierarchy, avoiding temporal downsampling layers until the global pooling layer before classification. It also operates with significantly lower channel capacity (β) compared to the Slow pathway, where $\beta < 1$ (typical value $\beta = 1/8$).

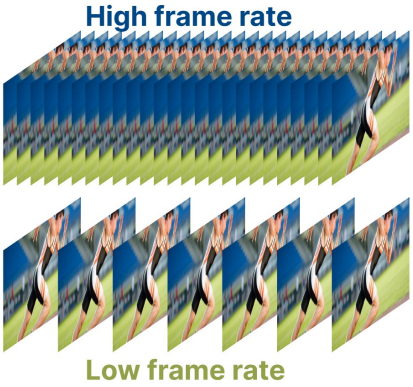


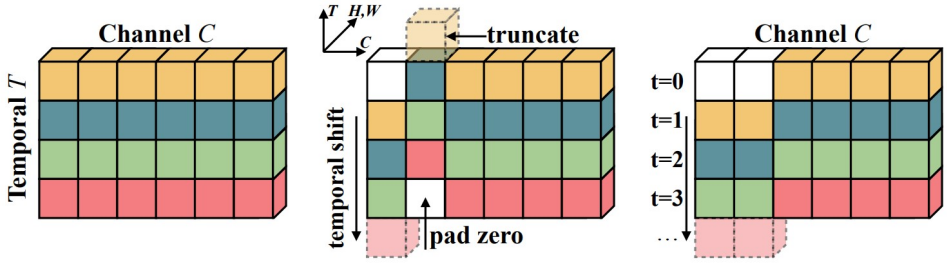
Figure 3.11: SlowFast: Slow pathway and the Fast pathway

In summary, the SlowFast architecture synergizes the temporal fidelity of the Slow pathway with the fine temporal resolution of the Fast pathway. The Fast pathway, although lightweight, effectively captures high-temporal-resolution features while making a desired tradeoff by weakening its spatial modeling ability. This innovative

approach enhances computational efficiency, making the SlowFast model a powerful solution for video analysis.

3.4.6 Temporal Shift Module (TSM)

The Temporal Shift Module (TSM) [49] represents a sophisticated mechanism for achieving efficient temporal modeling within the context of video recognition architectures. Its primary functionality revolves around the adept manipulation of feature maps along the temporal dimension, facilitating robust temporal modeling without incurring significant computational overhead. TSM seamlessly integrates with 2D convolutional operations, introducing an additional layer of temporal adaptability to enhance the model's understanding of dynamic video sequences. One of the notable advantages of TSM lies in its versatility, offering support for



(a) The original tensor without shift. **(b)** Offline temporal shift (bi-direction). **(c)** Online temporal shift (uni-direction).

Figure 3.12: Temporal Shift Module (TSM): A tensor comprises C channels and T frames. Features at distinct time stamps are represented by varying colors in each row. Within the temporal dimension, a subset of channels shifts by -1, another by +1, while the remainder remains unaltered (refer to Figure 1b). An online version of TSM is also presented for online video recognition (refer to Figure 1c). In this setting, access to future frames is restricted, limiting shifts only from past frames to future frames in a uni-directional manner.

both offline and online video recognition scenarios. In offline video recognition, where high throughput is a priority, TSM incorporates a bi-directional approach. This method intelligently amalgamates information from both past and future frames with the current frame, harnessing a comprehensive temporal context to bolster recognition accuracy. Conversely, in scenarios demanding low-latency online video recognition, TSM employs a uni-directional strategy. Here, it selectively incorporates information solely from the past frame, ensuring that the temporal modeling process is streamlined to meet the demands of real-time video analysis.

3.5 Missing Modality

We can refer to the missing modality as a problem where, at test time, there is the possibility that one or more modalities may be absent for a model that has been trained with all modalities. This issue may stem from extraction errors, low computational cost requirements, or limited hardware resources that prevent the extraction or processing of computationally demanding modalities. In this highly realistic multimodal context, the challenge lies in designing a model that can maximize synergy between modalities and manage the potential absence of a subset of them. Following the taxonomy proposed by [50], we can summarize the possible methodologies as follows:

- **Synthesis models:** A viable solution to address this challenge involves the use of synthesis models, which aim to create or estimate missing modalities to maintain a complete and coherent data flow during model testing.

  For instance, a synthesis model may leverage techniques such as neural networks or restricted Boltzmann machines [51] to generate missing data based on information available in other modalities. This synthesis process aims to enhance the model's ability to cope with data absence during testing, enabling more accurate and consistent predictions even in the case of missing modalities.

- **Shared latent space:** Common Latent Space allows for the creation of a unified or shared representation for the involved modalities. This is crucial because it enables the model to learn common features among different modalities, facilitating a cohesive interpretation of the data even in the absence of specific modalities. Hetero-Modal Image Segmentation (HeMIS) [52] is an example that utilizes shared latent space (Fig. 3.13). The HeMIS architecture applies a series of three connected blocks: a Back-end block to encode each modality into a latent space, an Abstraction block to extract statistical features (first and second-order moments) and finally a Front-end block to generate the prediction.

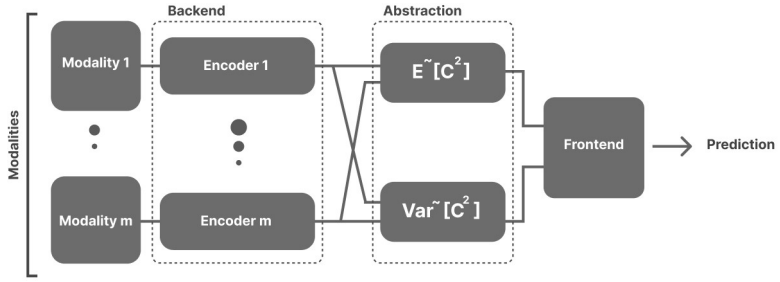


Figure 3.13: Hetero-Modal Image Segmentation (HeMIS) [52]

- **Knowledge distillation networks:** The methodology of knowledge distillation [53] in the context of missing modality involves training a larger, more complex model, known as the teacher network, with access to all available modalities. The knowledge gained from this comprehensive model is then transferred to a smaller model, known as student, with the objective of obtaining a network that can effectively operate with a subset of modalities. This approach aims to distill the valuable information and insights obtained by the teacher network into a more compact student network.

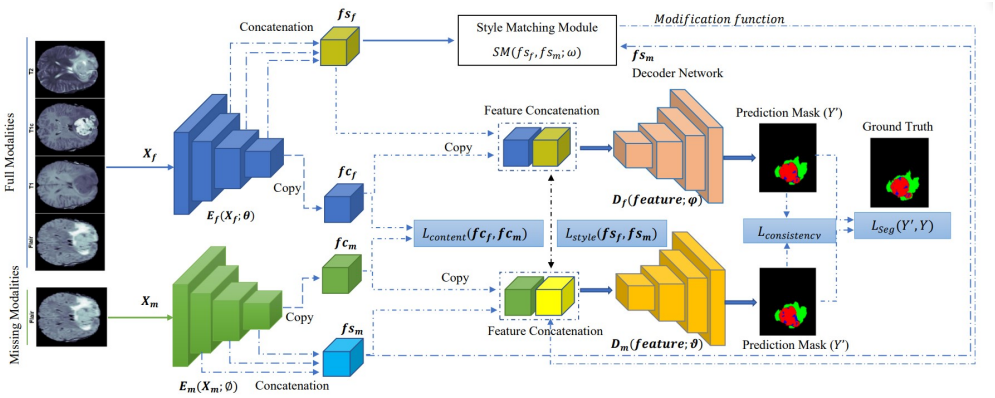


Figure 3.14: The technique for style matching, as introduced in [54], employs a knowledge distillation strategy to align the style and content representations between the complete modality and the absent modality pathways.

One notable implementation of knowledge distillation is [54], a style matching U-Net provides a solution for missing modalities by decomposing the feature representation into style and content 3.14. This approach conducts style and content matching at different levels, effectively distilling informative features from the full-modality path into a dedicated network designed for handling missing modalities.

- **Mutual information maximization:** the mutual information maximization [55] strategy involves optimizing similarity metrics between available modalities during training, aiming to minimize information loss. By maximizing mutual information, the model aims to enhance the relationships and dependencies between modalities, ensuring that the information shared among them is effectively captured and utilized. This strategy contributes to a more robust and comprehensive representation of the data, even when specific modalities are absent, thereby improving the model's ability to handle missing modalities during testing.

- **Generative Adversarial Networks** In the context of addressing missing modalities, Generative Adversarial Networks (GANs), first presented by [56], constitute a machine learning paradigm comprising a generator (G) focused on generating the missing modality and a discriminator (D) assigned to differentiate between generated samples and the original training data. Simultaneous training of both networks improves the generator's capability to reconstruct samples that include the absent information. An exemplary

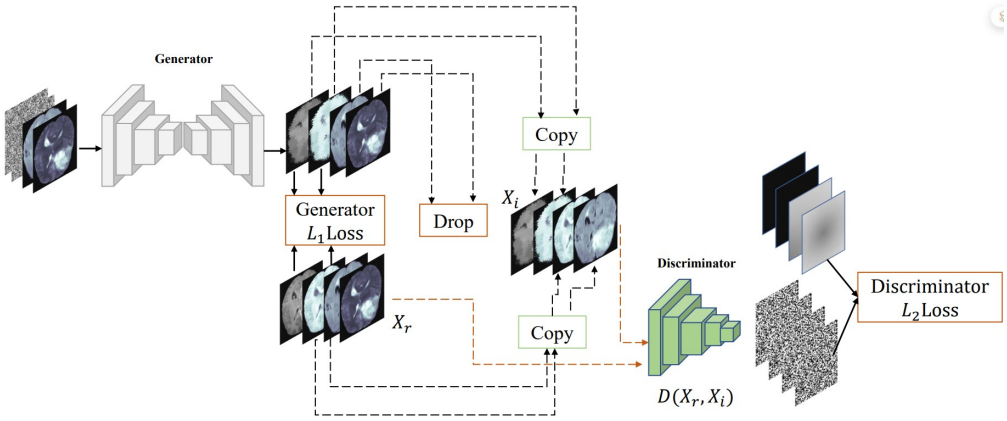


Figure 3.15: The Multi-Modal Generative Adversarial Network (MM-GAN) [57]

method employing this technique is the multi-modal generative adversarial network (MM-GAN)[57]. This approach adeptly synthesizes the missing modality in a single forward pass, benefiting from the enhancement provided by implicit conditioning (IC). The process involves the generator (U-Net) imputing the missing modality, computing the L1 loss for the generated scans, and employing a discriminator (PatchGAN) [58] with modality-selective L2 loss computation.

Chapter 4

Data

4.1 Epic Kitchens Dataset

This section introduces EPIC-KITCHENS-100 [59], an extensive dataset in the domain of egocentric vision. Building upon the foundation of EPIC-KITCHENS, this enhanced dataset, comprising 100 hours of footage across 700 variable-length videos, captures unscripted activities in 45 distinct environments. The videos are recorded using head-mounted cameras, offering a unique perspective for understanding long-term, daily-life actions.

EPIC-KITCHENS-100 boasts notable improvements over its predecessor, EPIC-KITCHENS [60], in terms of annotation density and completeness. Employing a novel annotation pipeline, it achieves 54% more actions per minute and a substantial increase (+128%) in fine-grained action segment annotations. The dataset introduces new challenges, including action detection and assessing model generalization over time. Specifically, it examines whether models trained on data collected in 2018 can adapt to new footage acquired two years later.

Aligned with six challenges, EPIC-KITCHENS-100 supports action recognition (full and weak supervision), action detection, action anticipation, cross-modal retrieval (from captions), and unsupervised domain adaptation for action recognition. Each challenge is accompanied by defined tasks, baseline models, and evaluation metrics. The dataset encompasses 89,977 segments of finely annotated actions extracted from 700 videos, totaling 100 hours in length. Table 1 presents comprehensive statistics, distinguishing between videos collected previously and those newly acquired.

Comparing to the previous dataset, EPIC-KITCHENS-100 exhibits a significant expansion, nearly doubling in duration with 1.8 times more hours and featuring 2.3 times more action segments. In Fig. 4.1, the frequency distribution of verb (97) and noun (300) classes reveals a clear long-tail distribution grouped into 13 verb

and 21 noun categories. The dataset splits into Train/Val/Test, with 75/10/15 ratios, with the Test split exclusively comprising newly-collected videos.

Additionally, the Val/Test splits include subsets such as Unseen Participants, featuring individuals not present in the training set, and Tail Classes, representing smaller classes accounting for 20% of total instances in training. These subsets enhance evaluations of model generalizability and performance on challenging classes, respectively.

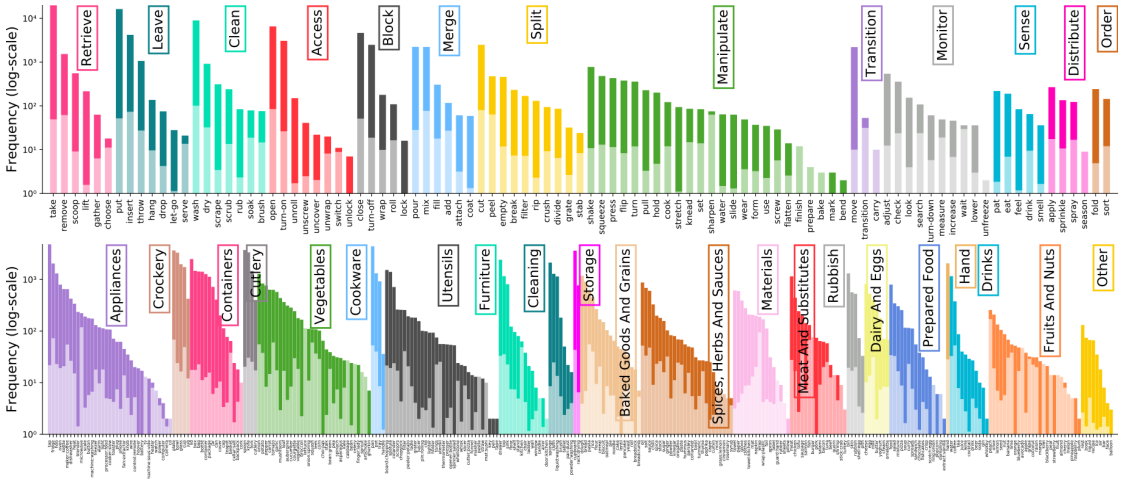


Figure 4.1: Epic-Kitchens-100 verbs and nouns distributions

Figure 4.2: Sample of Epic-Kitchens-100

Chapter 5

Methodology

5.1 Problem Statement

The ultimate goal of the model is to identify an action within a time window. In the context of the EPIC-KITCHENS-100 dataset [59], an action is defined by a verb and an object, as specified in the previous section. In order to accurately classify these two constituent elements of an action, we opted to use two distinct modalities: Optical Flow and RGB. Optical flow is a modality with information inherent to the relative motion of the elements in a video, for this reason it appears to be a good choice for verb classification. At the same time, RGB provides the information needed to identify the active object in the scene. The multimodal choice offers advantages associated with the distinctive information that each modality provides, but poses challenges related to their heterogeneity due to their different nature and the dissimilar preprocessing of each. Some modalities may have a high computational cost of predelaboration. In settings where limited hardware resources are available or processing time is prioritized, modalities such as optical flow may be unaffordable or unusable. In order to address this realistic issue, we adapted the model to the missing modality setting with several different implementations, including the introduction of a reconstruction module that reconstructs the absent modality from the one available. In addition, following the footsteps of previous studies such as SlowFast [19] and TRN [48] that have shown how multi-level temporal aggregation leads to the extraction of distinct and complementary information, we decided to raise the aggregation plan to a level that works on sequences of actions. To do this we implemented a module that takes as input an entire scene and allows the visual information related to each action to communicate with each other resulting in clip-level aggregation. Detailed explanations of the implemented solutions are provided in the subsequent paragraphs.

5.2 Model Architecture

Leveraging a strategic fusion of innovative components, our core model architecture 5.1 is designed to excel in video action recognition task. At its foundation, we employ pre-trained Convolutional Neural Networks (CNNs) as feature extraction modules, setting the stage for the subsequent integration of a Temporal Reasoning Graph Module and a Cross-Modal Interaction Module.

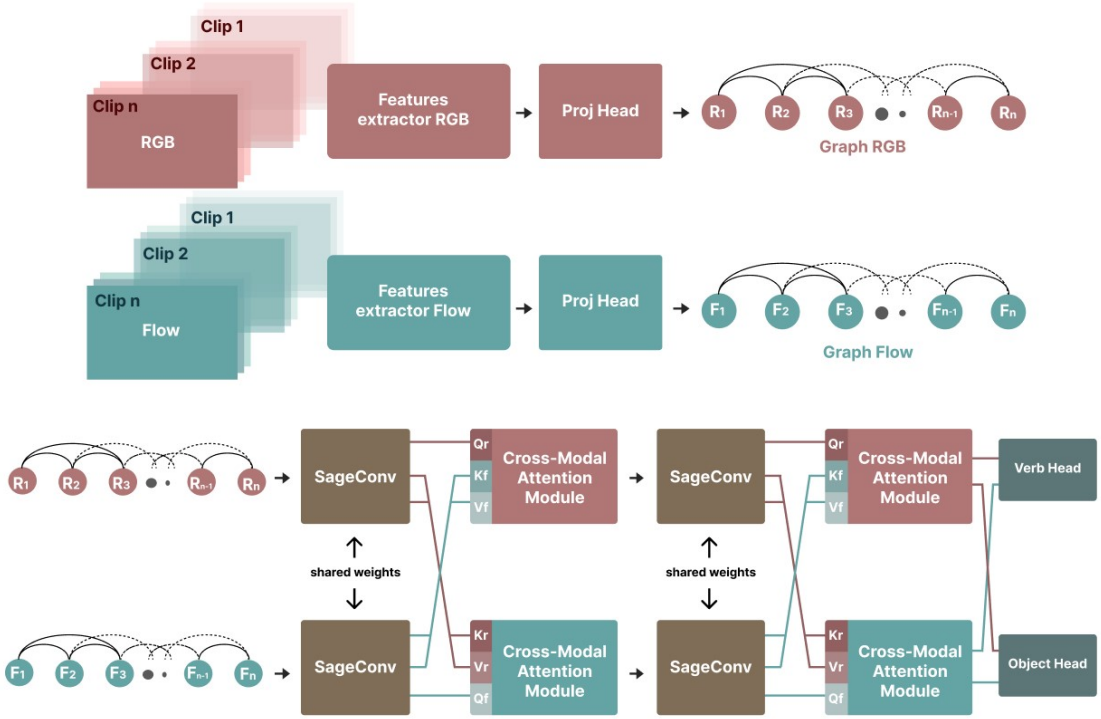


Figure 5.1: MARE-Graph architecture: Utilizing pre-trained CNNs for feature extraction, SAGEConv layer for temporal aggregation, and a Cross-Modal Interaction Module for nuanced cross-modal reasoning

The primary motivation behind utilizing pre-trained CNNs lies in their proven efficacy as robust feature extractors across diverse visual datasets. These modules act as the bedrock for our model, providing a rich representation of visual information.

The temporal reasoning graph module is introduced to address the inherent inefficiencies of pre-trained CNNs in capturing high-level temporal context.

The video is modeled as a regular graph, wherein each video clip serves as a node, and connections exist between each clip and its preceding and subsequent clips, forming an observation window. This graph representation is then input into a SAGEConv convolutional layer. In this context, the SAGEConv convolutional layer

proves advantageous due to its ability to aggregate information from neighboring nodes within the defined observation window. This aggregation mechanism allows the model to capture temporal dependencies and relationships effectively.

The regular graph structure, with clips as nodes and connections delineating temporal associations, aligns with the sequential nature of video data.This adaptive aggregation enables the model to discern nuanced temporal patterns and dependencies, contributing to robust temporal modeling.

Complementing the temporal reasoning graph module, our architecture incorporates the Cross-Modal Interaction Module. This component is motivated by the necessity to enhance the model's cross-modal reasoning capabilities. Through a flexible cross-attention mechanism, the module enables individual modalities to dynamically exchange information, fostering a holistic understanding of the input data. This dynamic exchange not only enhances the interpretability of the model but also ensures robust performance across diverse scenarios.

The strategic combination of these components results in a versatile and powerful model architecture, adept at addressing the intricacies of video analysis tasks. The pre-trained CNNs provide a strong foundation for feature extraction, the temporal reasoning graph module efficiently captures temporal dependencies, and the cross-modal interaction module facilitates nuanced cross-modal reasoning. Together, these components synergize to empower our model leading to superior performance.

5.2.1 Modalities and Backbones

Large-scale datasets like Epic-Kitchens [59] and Ego4d [61] offer multiple modalities, fostering the exploration of novel techniques that leverage the unique characteristics of each modality.

Audio data, for instance, serves as a valuable complement to the visual stream, transcending the constraints of the camera's field of view. Typically, fixed-size audio segments undergo conversion to spectrograms, followed by processing through a CNN backbone, treating them akin to 2D images. Consequently, audio-based learning presents a computationally lightweight task compared to other data sources incurring higher computational costs.

Optical flow, in contrast, identifies the direction of motion between consecutive frames, providing valuable insights into the areas involved in an action. By focusing solely on motion information, optical flow reduces the influence of visual characteristics such as color or text, which can sometimes hinder generalization.

The decision to incorporate both RGB and optical flow modalities is grounded in their distinct functional attributes, each addressing specific aspects of the video analysis task. RGB, chosen for its effectiveness in capturing visual information, is well-suited for tasks involving object classification. The inherent characteristics of RGB, encompassing color and spatial features, align with the requirements for

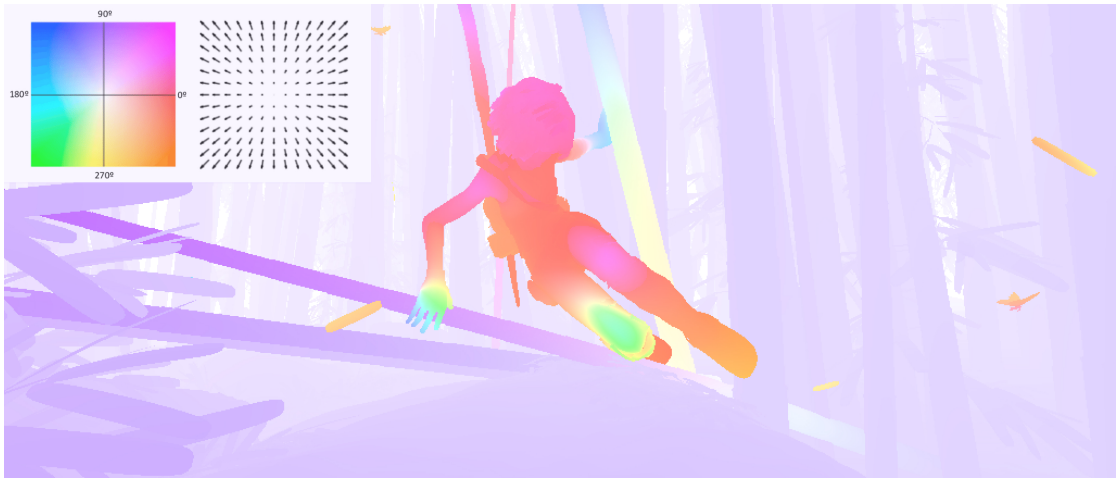


Figure 5.2: A frame derived from the Sintel dataset showcasing optical flow and the associated color coding scheme for optical flow representation

recognizing and categorizing objects within video frames.

Conversely, optical flow emerges as the optimal selection for verb classification, capitalizing on its efficacy in delineating temporal dynamics across consecutive frames.

The amalgamation of RGB and optical flow modalities, tailored to their respective strengths, synergistically enhances the holistic video analysis pipeline, providing a nuanced understanding of both object and verb components.

To showcase the versatility of our implementation, we systematically assessed the efficacy of diverse backbone architectures through comprehensive experiments. We specifically examined the extracted features from Temporal Segment Networks (TSN) [20], Temporal Shift Modules (TSM) [49], SlowFast [19], and combinations of these. This rigorous evaluation aimed to elucidate the contributions of each backbone configuration and their collective impact on the overall performance of the model.

5.2.2 Temporal Graph Modeling

The features originating from the initial frame-level representations provided by the backbones are systematically aggregated to form clip-level representations for both optical flow and RGB streams performed through two distinct projection heads. These representations serve as nodes within our constructed graph. Our architectural selection encompasses the establishment of a regular graph, wherein each node systematically observes 'n' preceding and 'n' succeeding nodes. The parameter 'n' intricately defines a temporal window, delimiting the scope within which each node assimilates contextual information from its temporal neighbours.

Formally, Let $G_{\text{RGB}} = (V_{\text{RGB}}, E_{\text{RGB}})$ and $G_{\text{OF}} = (V_{\text{OF}}, E_{\text{OF}})$ denote the constructed graphs for the RGB and Optical Flow modalities, respectively, where V_{RGB} and V_{OF} represent the set of nodes and E_{RGB} and E_{OF} represent the set of edges. Each node v_i in the graph corresponds to a clip-level representation obtained from the initial frame-level representations provided by the backbones. Let $B_{\text{RGB}} = \{f_{\text{RGB}}^{(1)}, f_{\text{RGB}}^{(2)}, ..., f_{\text{RGB}}^{(n)}\}$ denote the features obtained from the RGB backbone, and $B_{\text{OF}} = \{f_{\text{OF}}^{(1)}, f_{\text{OF}}^{(2)}, ..., f_{\text{OF}}^{(m)}\}$ denote the features obtained from the optical flow backbone. These features are concatenated to form the input to two distinct multi-layer perceptrons (MLPs), denoted as MLP_{RGB} and MLP_{OF}. The resulting clip-level representations C_{RGB} and C_{OF} are obtained as follows:

$$C_{\text{RGB}} = \text{MLP}_{\text{RGB}}(B_{\text{RGB}})$$

$$C_{\text{OF}} = \text{MLP}_{\text{OF}}(B_{\text{OF}})$$

These clip-level representations C_{RGB} and C_{OF} serve as the nodes within the respective graphs. The set of edges E can be defined as:

$$E = \{(v_i, v_j) \mid v_i, v_j \in V, \text{ and } |i - j| \leq n\}$$

Here, $|i - j|$ represents the temporal distance between nodes v_i and v_j, and n defines the size of the temporal window.

This decision is motivated by the intention to capture contextual information from a localized temporal window, enhancing the model's ability to discern temporal dependencies and patterns within the video data.

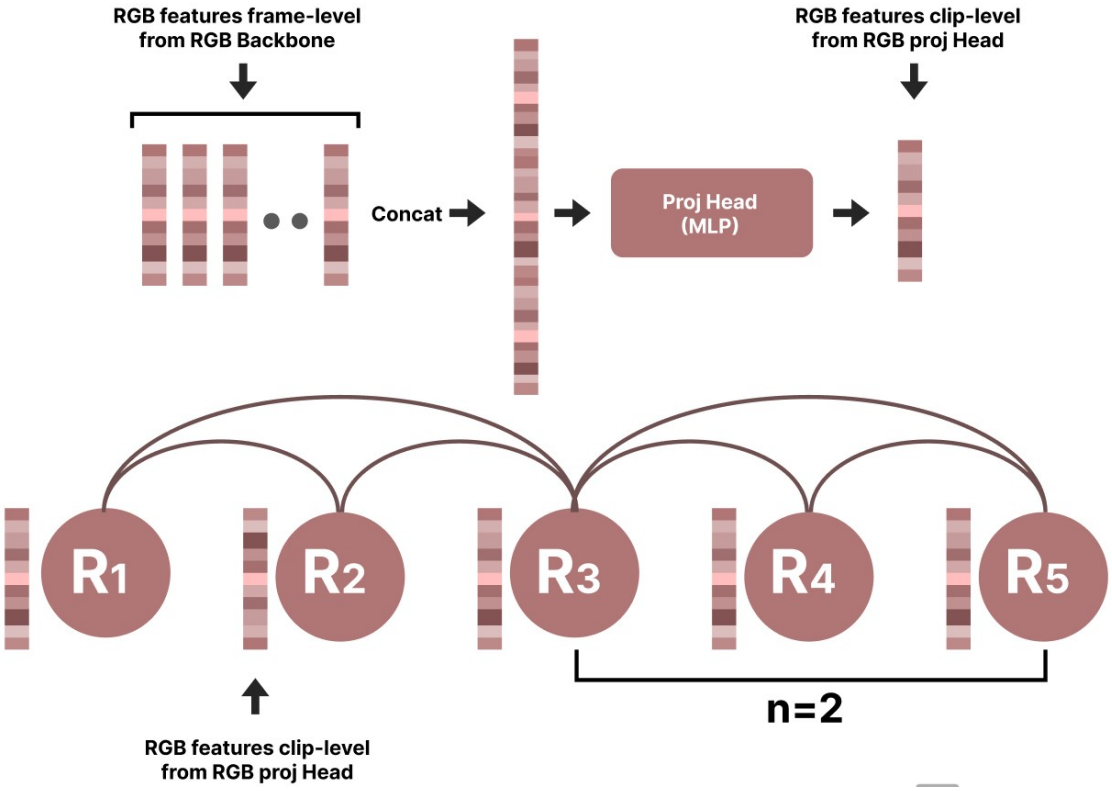


Figure 5.3: Temporal Graph Modeling: Features from frame-level representations aggregate into clip-level nodes within a regular graph. Nodes observe 'n' preceding and 'n' succeeding nodes, defining a temporal window.

5.2.3 Temporal Reasoning Module

The Temporal Reasoning Module, illustrated in the accompanying figure 5.4, is integral to our architecture. In order to bolster the model's capacity for generalization and extract intricate interactions, we have incorporated both a Dropout layer and a RELU activation function.

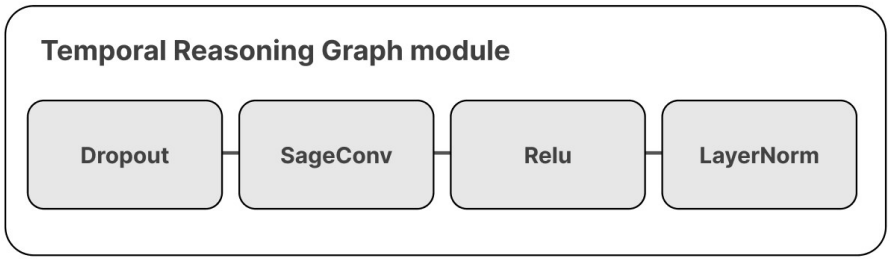


Figure 5.4: Temporal Reasoning Module

The primary submodule dedicated to the actualization of temporal processing within this module is the SAGEConv. As elucidated by the equation 5.1, this submodule adeptly aggregates information from neighboring nodes, thereby enriching and updating the original information associated with the observing node through the assimilation of contextual information.

$$x_i' = W_1 x_i + W_2 \cdot \max_{j \in N(i)} x_j \tag{5.1}$$

Furthermore, a crucial objective of employing this submodule is to facilitate the subsequent communication between modalities. This is achieved through the intricacies of intra-modal aggregation, overseen by the Cross-Modal Interaction Module. The Cross-Modal Interaction Module is specifically designed to manage and optimize the exchange of information across diverse modalities, ensuring a synergistic collaboration that enhances the overall efficacy and comprehensiveness of our model.

5.2.4 Cross-Modal Interaction Module

The Cross-Modal Interaction Module incorporates a cross-attention transformer encoder, serving the dual purpose of preserving distinctive intra-modal aggregations and establishing effective inter-modal observations.

Each vector representation undergoes an additive process involving relative positional encoding. This encoding serves a dual purpose: first, it facilitates temporal synchronization across modalities, ensuring cohesive temporal alignment; second, it enables the module to extract temporal patterns that necessitate sequence order information. This approach of introducing relative positional encoding contributes significantly to the module's capacity to discern intricate temporal patterns effectively while maintaining temporal alignment across modalities.

To optimize the module's temporal reasoning capabilities, a sliding attention window is incorporated into its design. This design choice allows the module to focus selectively on a limited contextual window, prioritizing the relevance of closer temporal relationships. The rationale behind this lies in the understanding that temporal dependence tends to diminish as one progresses temporally away from the query action

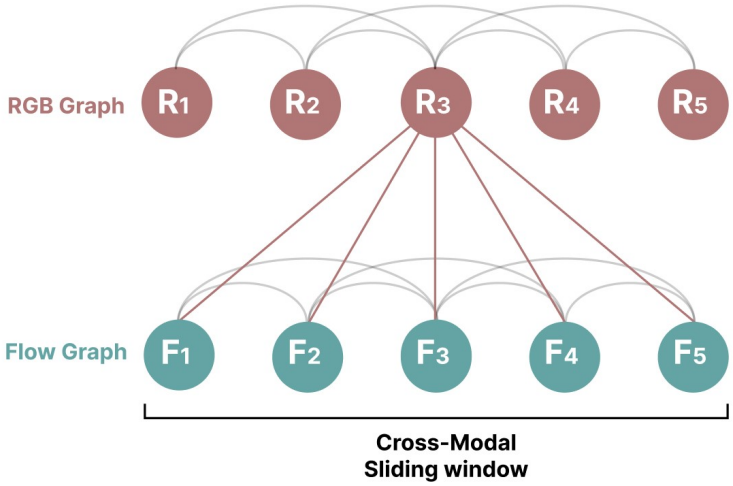


Figure 5.5: The Cross-Modal Interaction Module integrates a cross-attention transformer encoder, ensuring intra-modal aggregations are preserved while establishing effective inter-modal observations. The sliding attention window optimizes the module's temporal reasoning by selectively focusing on a limited contextual window, prioritizing closer temporal relationships

Let $\mathbf{Q}_t^{(1)}$ represent the query vector from modality m_1 at time t, and $\mathbf{K}_{t'}^{(2)}$ and $\mathbf{V}_{t'}^{(2)}$ denote the key-value pairs from modality m_2 within an attention window $[t-n, t+n]$. Relative position embeddings are learned to capture the pairwise distances between query and key positions within this window. These embeddings, indexed by distance, interact with queries to produce a logits matrix $\mathbf{S}_{\text{rel}}$, which modulates attention probabilities. Formally, Relative Attention is defined as:

$$\text{RelativeAttention} = \text{Softmax}\left(\frac{Q_t^{(1)} K_{t'}^{T(2)} + S_{\text{rel}}}{\sqrt{D_h}}\right) V_{t'}^{(2)},$$

where $S_{\text{rel}} = QR^\top$ represents the calculation of the logits matrix. Here, R is a matrix of relative position embeddings, where each entry R_{ij} corresponds to the relative distance between the i-th query and the j-th key within the attention window.

Since the calculation of R is a memory bottleneck, as the matrix requires $O(L^2 d)$ extra space, the skewing mechanism introduced in [62] was implemented by calculating S_{rel} without explicitly calculating R (Fig. 5.6). The algorithm originally designed for causal attention was adapted for the entire sliding window.

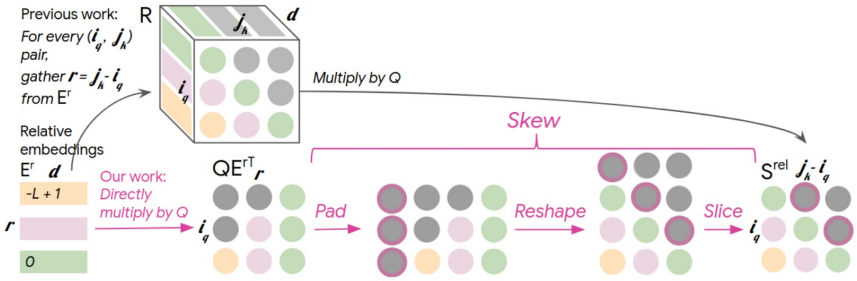


Figure 5.6: The bottom row illustrates the memory-efficient "skewing" algorithm, which eliminates the need for instantiating R (top row, which has a complexity of $O(L^2 d)$. Gray shading indicates masked or padded positions, while each color represents a distinct relative distance.

In summary, the Cross-Modal Interaction Module is designed with these elements to strike a delicate balance between capturing essential temporal dependencies and mitigating computational complexity. This holistic approach ensures that the module stands as a pivotal component in the extraction and processing of temporal patterns, elevating the system's performance in multimodal action recognition.

5.3 Missing modality setting

In order to assess the effectiveness of the model in capitalizing on synergies among modalities, a thorough exploration is undertaken, focusing on scenarios where a particular modality might be unavailable during testing. This examination extends beyond conventional testing scenarios, allowing for a more nuanced understanding of the model's behavior and its potential applications in real-world, resource-constrained environments. Within this context, two distinct implementations have been developed: DropGraph and the Reconstruction module.

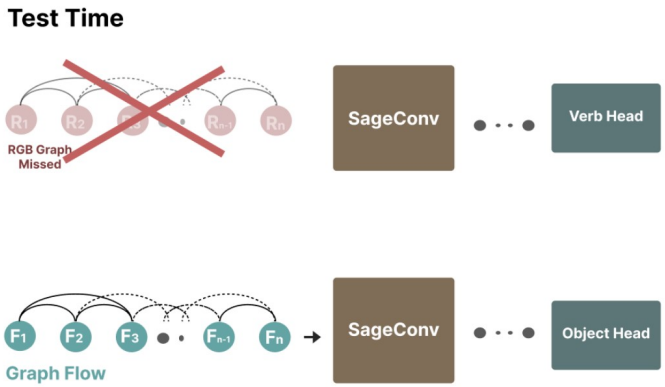


Figure 5.7: Missing modality: a particular modality might be unavailable during testing

- **DropGraph:** In the DropGraph implementation, the training process involves setting with a probability p_1 , the vector representations related to a video clip to zero, and with a probability p_2 , the entire graph is dropped. This approach is grounded in the belief that introducing stochasticity during training enables the model to adapt to scenarios where certain modalities might be missing during the testing phase. The advantages of this method lie in its simplicity and the ability to mimic real-world scenarios where data might be incomplete.
- **Reconstruction module:** On the other hand, the Reconstruction module tackles the missing modality challenge by incorporating a reconstruction mechanism. In this setup, one of the modalities is taken as input, and the module seeks to reconstruct the missing one. The reconstruction process employs a Multilayer Perceptron (MLP), and the Mean Squared Error (MSE) loss is computed between the reconstructed modality and the original one. This approach leverages the power of reconstruction models to predict missing modalities, fostering adaptability and robustness in scenarios where certain modalities may be absent.

5.3.1 DropGraph

The DropGraph implementation (Fig. 5.8) introduces a training strategy that incorporates a form of dropout, specifically applied to both vector representations of video clips and the entire graph. This dropout mechanism is governed by probabilities p_1 and p_2 , determining the likelihood of zeroing out vector representations and dropping the entire graph, respectively. During the training process, with a probability p_1 , the vector representations associated with a video clip are stochastically set to zero. Simultaneously, with a probability p_2 , the entire graph is dropped, introducing a level of randomness to the model's learning process. This stochasticity serves as a form of regularization, preventing the model from becoming overly reliant on specific modalities or structures.

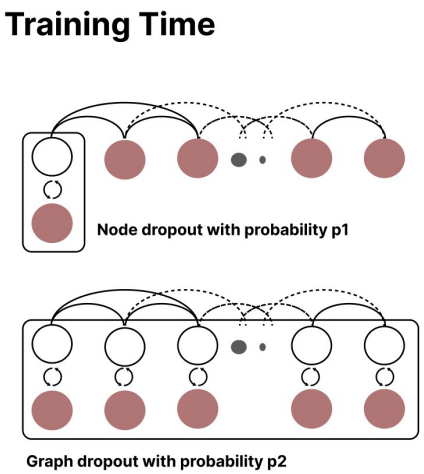


Figure 5.8: DropGraph: introduces dropout during training, stochastically zeroing out video clip vector representations (with a probability p_1) and dropping the entire graph (with a probability p_2). This adds regularization, preventing over-reliance on specific modalities and ensuring alignment between training and test conditions.

Advantages of the DropGraph method include its simplicity in implementation and the capacity to emulate real-world scenarios where data may be incomplete or certain modalities might be missing. The stochastic dropout nature helps the model generalize better to unforeseen situations and promotes adaptability. Furthermore, the simplicity of the approach facilitates seamless integration into various architectures. In this context, cross attention plays a crucial role by enabling the potential reconstruction of the missing modality through the observation of the available one. However, it is essential to note potential drawbacks. Excessive dropout rates (p_1 and p_2) might lead to loss of valuable information and hinder the model's learning capacity.

5.3.2 Reconstruction module

The Reconstruction module (Fig. 5.9) addresses the issue of missing modalities by introducing a reconstruction mechanism into the system. In this configuration, one of the modalities serves as the input, and the module endeavors to reconstruct the missing modality. The reconstruction is carried out through the utilization of a Multilayer Perceptron (MLP), and the Mean Squared Error (MSE) loss is calculated by comparing the reconstructed modality with the original one. This methodology harnesses the capabilities of reconstruction models to predict absent modalities, thereby enhancing adaptability and robustness in situations where specific modalities might be lacking.

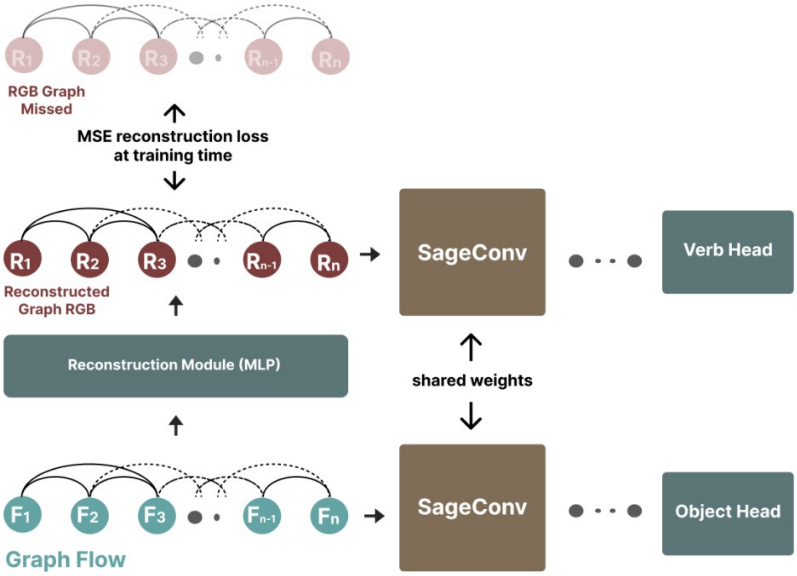


Figure 5.9: The Reconstruction module predicts missing modalities through an MLP-driven reconstruction mechanism. This approach fosters adaptability in scenarios with incomplete data promoting robustness. The rest of the network remains unaltered

Advantages of the Reconstruction module include its ability to predict missing modalities, which contributes to a more comprehensive and complete representation of the data. The use of an MLP allows for non-linear mappings and complex relationships to be captured during the reconstruction process. This approach promotes adaptability in scenarios with incomplete data, as the module learns to fill in the gaps based on the information available from the observed modality.

Chapter 6

Experiments and Results

6.1 Implementation

6.1.1 Backbones implementation

Publicly available PyTorch [63] model definitions for TSN [20], TSM [49], and SlowFast [19] are utilized in this study. The ResNet-50 [6] backbone is employed across all models, with ImageNet [64] weights for TSN and Kinetics weights for TSM and SlowFast. Two instances of each model undergo training: one with 8 RGB frames and the other with 8 stacks of 5 (u, v) flow fields computed using TV-L1 [65].

- **TSN and TSM models:**
  A two-way output in the last layer predicts verbs and nouns with an average verb/noun loss. Training spans 80 epochs with SGD, momentum 0.9, and a learning rate of 0.01, decayed at epochs 20 and 40 by a factor of 10. TSN and TSM models train on 8 GPUs with a batch size of 128, while SlowFast uses a batch size of 32 on 8 GPUs. Weight decay of 0.0005, dropout with p = 0.7, and gradient clipping above 20 are applied. Center-crop evaluation is used, and RGB and optical flow models are trained individually, with predictions averaged pre-softmax during inference.

- **SlowFast model:**
  For SlowFast, the publicly available PyTorch model [19] is employed. The model is adjusted to feature a two-way output for verbs and nouns, trained with the average verb-noun loss. SlowFast 8x8 with ResNet-50 backbone, initialized from Kinetics pretrained weights, is utilized. Training encompasses 30 epochs with SGD, momentum 0.9, and a learning rate of 0.01, decayed at epochs 20 and 25 by a factor of 10. The model is trained on 8 GPUs with a batch size of 32, incorporating a weight decay of 0.0001 and dropout with p =

0.5. Batch-normalization layers' parameters and statistics are frozen during training. During testing, predictions are obtained by uniformly sampling 10 clips (1s each) from each video, and averaging the results from a single center crop per clip.

Pretrained models on Epic-Kitchens were employed, utilizing weights provided by [59].

6.1.2 MARE-Graph implementation

The entire architecture was trained for 100 epochs, freezing the backbone weights, using Adam optimizer with a weight decay of 5e-4. The learning rate was initially set to 1e-4 and decayed at epochs 50 and 75 by a factor of 10. The model was trained on a single GPU with a batch size of 4 graphs. Backbone features are aggregated through concatenation and projected into a 1024-dimensional space.

The final configurations include a radius graph with a distance of 4. Edges are created based on node positions to all points within a specified distance. The Temporal Reasoning Graph Module consists of a dropout set at 0.5, a SAGEconv, and a layer norm. The SAGEconv is configured with the max function as the aggregation method, linear projection, and subsequent RELU activation function before aggregation. The final output is L2 normalized.

The sliding attention window width of the cross-module is set to 10. A relative positional encoding is adopted, initializing learnable vectors for each possible relative distance during training. The positional encoding is added to the vector representation of the clip before each cross-module.

For regularization, DropPath [66] of 0.1 and a dropout of 0.5 for the feed-forward of the Transformer encoder are applied. The two final prediction heads are two MLPs with a dropout of 0.5, receiving the information from the final cross-attention and predicting verb and object, respectively. The training loss is the average cross-entropy for both verb and object predictions.

Computational resources were provided by VANDAL - VISUAL AND MULTIMODAL APPLIED LEARNING LABORATORY and HPC@POLITO.

6.1.3 Missing modality implementation

- **DrophGraph implementation:**
  In the context of the DrophGraph implementation, the original architecture remains unchanged. However, two dropout mechanisms were introduced. Firstly, a node dropout with a probability of 0.5 was added and a graph dropout with a probability of 0.1. The graph dropout randomly sets to zero either the entire graph of one modality or the other, with equal probability. Other settings and configurations remain consistent with the original implementation.

- **Reconstruction module implementation:**
  In the context of missing modality with a reconstruction module, we implemented an MLP with a dropout rate of 0.5. This MLP takes the available modality as input and aims to reconstruct the missing modality. The reconstruction mechanism is activated during training with a probability of 0.1. We employed a Mean Squared Error (MSE) loss between the original and reconstructed information to guide the reconstruction module. The overall architecture remains unchanged, along with other configurations and settings.

Normal settings	
Epochs	100
Optimizer	Adam
Classification loss	Cross-Entropy
Initial Learning rate	1e-4
Scheduler	MultiStepLR
Milestones	[50,75]
Weight decay	5e-4
Graph window	4
Cross attention window	10
hidden dimension	1024
CL Head Dropout	0.5
Transformer Feed-Forward Dropout	0.5
DropPath	0.1
Missing modality settings	
Reconstruction loss	MSE
Node Dropout	0.5
Graph Dropout	0.1

Table 6.1: Normal technical settings and settings relating to the missing modality

6.2 Sequence-based action recognition

In 6.2, the comparative baseline results on the Epic-Kitchens validation set are presented, revealing the Vision Transformer as the top-performing model. Notably, the SlowFast model with a single modality (RGB) achieves comparable performance to TSM with two modalities.

Top-1 accuracy (%) Baselines			
	Verb	Noun	Action
TSN	60.18	46.03	33.19
TRN	65.88	45.43	35.34
TBN	66.00	47.23	36.72
TSM	67.86	49.01	37.39
SlowFast	65.56	50.02	36.81
ViVit-L	66.4	56.8	44.0

Table 6.2: Baselines Action recognition in EPIC-KITCHENS-100 validation set. The results are those reported by the corresponding papers

Top-1 accuracy (%)			
RGB	Verb	Noun	Action
w=0	49.58	44.05	27.10
w=2	58.13	46.10	33.60
w=4	**58.26**	**46.21**	**33.70**
w=8	57.57	46.04	33.29
Flow	Verb	Noun	Action
w=0	55.97	31.52	23.14
w=2	60.80	36.05	28.24
w=4	**61.25**	**36.68**	**28.26**
w=8	60.59	36.07	27.70

Table 6.3: Impact of window size with respect to single modalities using TSN as backbone

Turning attention to 6.3, an ablation study on the graph window size dimension is detailed. The investigation seeks to discern the impact of varying window sizes on performance. Notably, if the window size exceeds a threshold, performance diminishes. This phenomenon can be attributed to the diminishing relevance of context information from more distant clips, introducing disturbances to the central action and undermining the fidelity of the original action representation. Empirically, a window size of 4 emerges as the optimal choice.

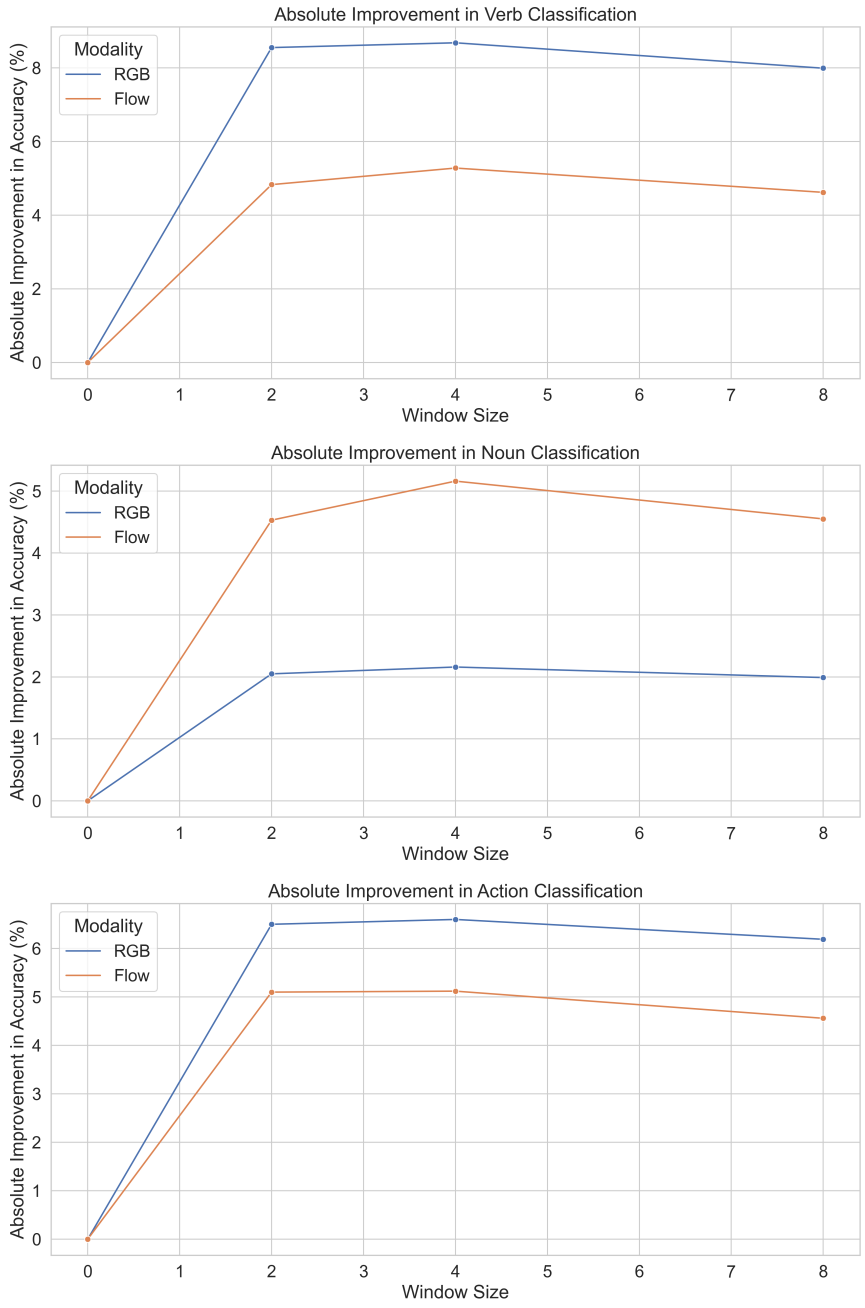


Figure 6.1: Plot of the absolute improvement in accuracy (%) with respect to the window size

6.3 Multi-modal action recognition

The presented results in 6.4 showcase an in-depth analysis of multi-modal action recognition using TSN as the backbone for both modalities. Notably, the addition of the temporal reasoning module and the cross module contributes significantly to the overall performance. Experimental exploration involved introducing a self-attention module before classification heads, aiming to enhance intra-modal aggregation. Self attention gives a minimal contribution when using TSN as a backbone, in the final implementation we opted not to use self attention since we noticed no contribution with other backbones, indicating that the SAGEconv module adeptly fulfills its role in intra-modal information aggregation.

			Top-1 accuracy (%)		
TR	Cross	Self	Verb	Noun	Action
×	×	×	60.18	46.03	33.19
✓	×	×	65.24	48.80	37.93
×	✓	×	64.14	47.23	36.30
×	×	✓	64.27	47.24	36.62
✓	✓	×	65.78	49.68	39.27
✓	×	✓	65.78	49.15	38.53
✓	✓	✓	66.27	50.32	39.97

Table 6.4: Impact of Temporal Reasoning (TR), Cross Attention, and Self Attention on Top-1 accuracy (%) for Verb, Noun, and Action classes. The table showcases the performance variations with different combinations of these modules

The observed improvement resulting from the cross-attention module highlights its pivotal role in facilitating multi-level inter-modal communication within the network, thereby enhancing the final classification accuracy. This enhancement is particularly noteworthy as it suggests that the cross-attention module enables effective information exchange at various levels, contributing to improved understanding and representation of complex multi-modal data. Moving on to 6.5, where the backbones employed include SlowFast for RGB features and TSM for Optical Flow features, various configurations were meticulously tested, with a specific focus on feature projection. The experimentation involved evaluating the impact of employing a single projection head versus two distinct projection heads. Notably, the configuration with two separate projection heads exhibited superior performance, underscoring the significance of this design choice in potentially mitigating challenges associated with heterogeneous data representations. Furthermore, the optimal configuration incorporated temporal reasoning modules with shared weights, providing additional insight into the architectural decisions affecting model performance.

			Top-1 accuracy (%)		
			Verb	Noun	Action
	TSM		67.86	49.01	37.39
	SlowFast		65.56	50.02	36.81
Head	Cross	TR	Verb	Noun	Action
2	Different	Different	69.91	55.03	44.12
1	Different	Different	69.83	54.74	44.00
2	Same	Different	69.89	55.01	44.21
2	Same	Different	69.99*	55.02*	44.31*
2	Different	Same	**69.96**	**55.26**	**44.44**

Table 6.5: Evaluation of different configurations with SlowFast for RGB features and TSM for Optical Flow features. The table presents the results with various combinations of cross attention and temporal reasoning modules (TR). Results with an asterisk are obtained by modality embedding before cross attention.

	Top-1 accuracy (%)		
	Verb	Noun	Action
TRN	65.88	45.43	35.34
TBN	66.00	47.23	36.72
SlowFast	65.56	50.02	36.81
ViVit-L	66.4	**56.8**	44.0
TSN	60.18	46.03	33.19
TSN+TR	65.24	48.80	37.93
MARE-Graph(TSN)	65.78	49.68	39.27
TSM	67.86	49.01	37.39
TSM+TR	68.30	52.1	41.65
MARE-Graph(TSM)	68.96	52.93	42.72
MARE-Graph(TSM-SlowFast)	**69.96**	55.26	**44.44**

Table 6.6: Comparative performance of different backbone architectures. The table shows the improvment of the temporal reasoning module (TR) and the utilisation of the entire MARE-Graph architecture with respect to different baselines and backbones.

Maintaining a nuanced perspective on 6.5, the distinct weights assigned to the cross-attention modules proved to be crucial. This design choice was observed to enable the extraction of unique and meaningful inter-modal communications based on the viewpoint, emphasizing the importance of preserving modality-specific information during the multi-modal fusion process. In the final 6.6, the adaptability of MARE-graph to various backbones is explored. The architecture consistently delivers substantial contributions, effectively handling features from diverse combinations. Specifically, the use of SlowFast for RGB-related features and TSM for flow-related features exemplifies the architecture's capability to manage and integrate information from disparate sources, showcasing its robustness and versatility in handling complex multi-modal features.

Below are presented the plots of the attention matrices for the first and second cross-module of the network, with respect to a sample from the dataset.

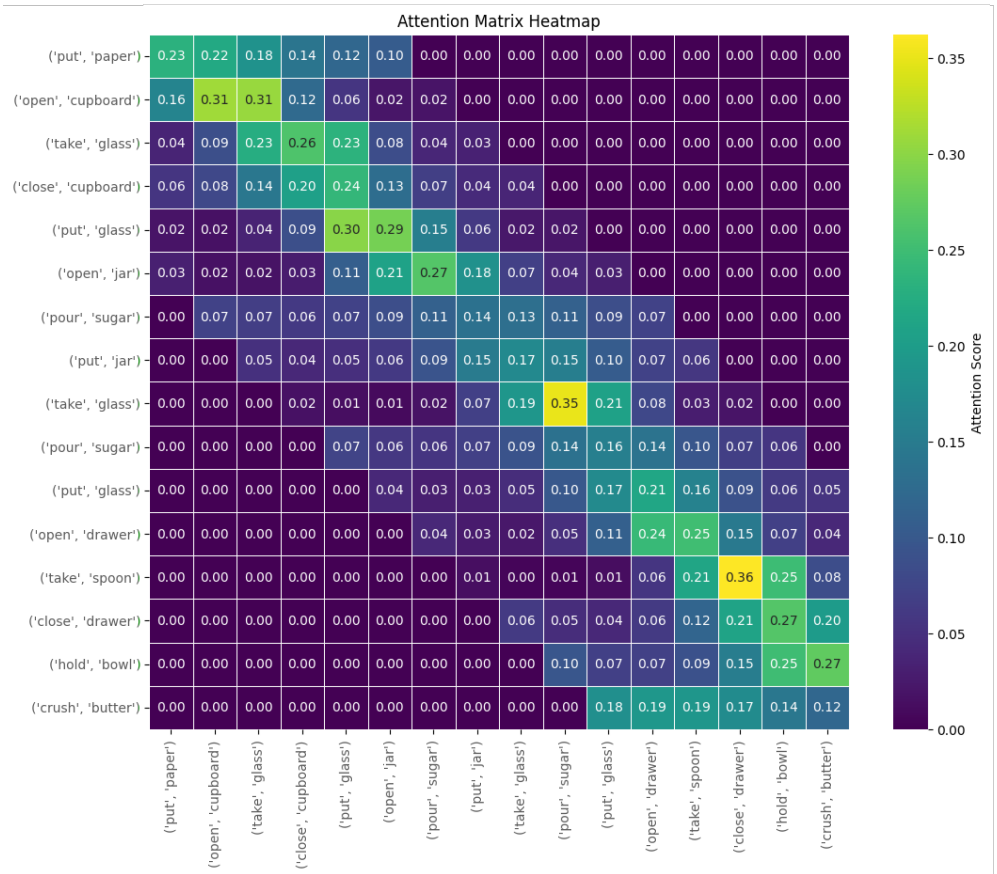


Figure 6.2: RGB Query in First Cross Attention: Attention matrix visualization representing the interaction between RGB features as the query and Optical Flow features as keys/values in the first cross-attention module.

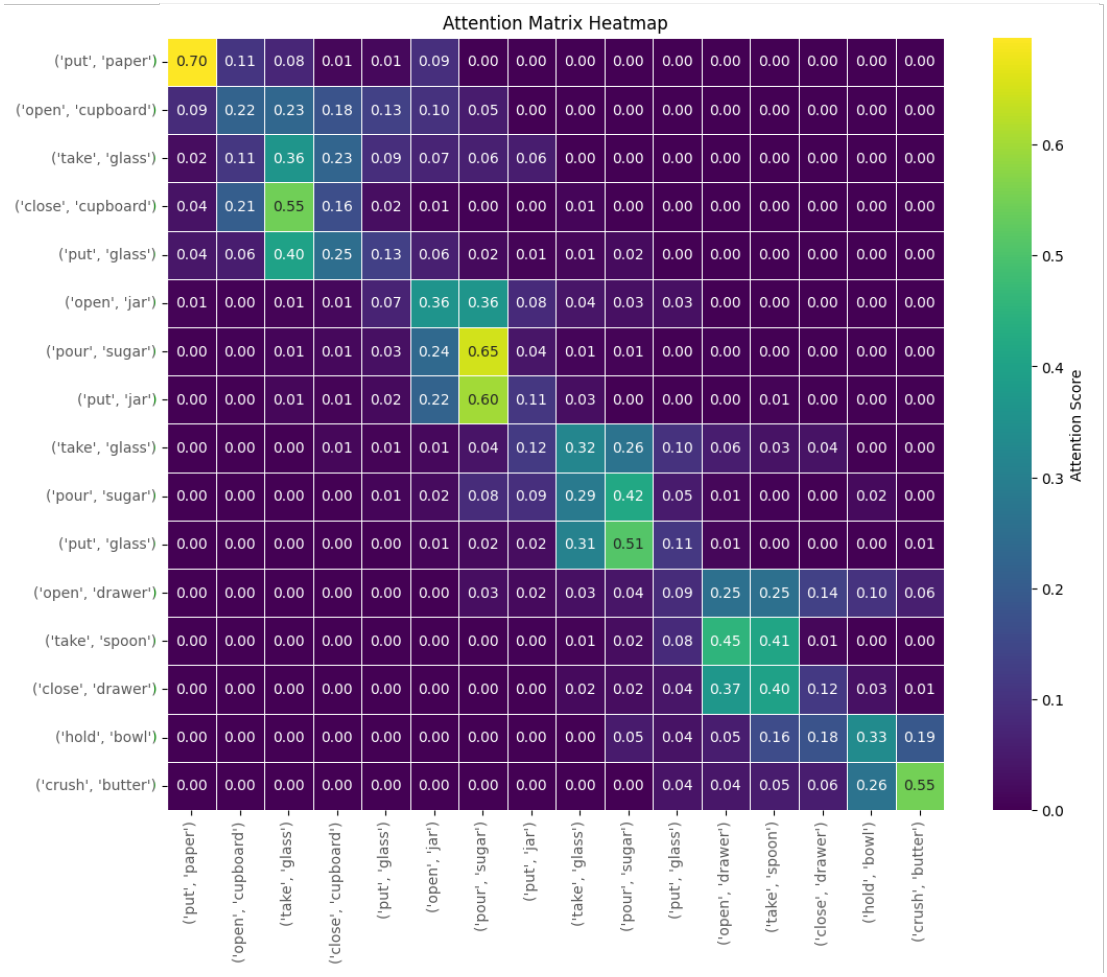


Figure 6.3: Optical Flow Query in First Cross Attention: Visualization of the attention matrix with Optical Flow features as the query in the first cross-attention module.

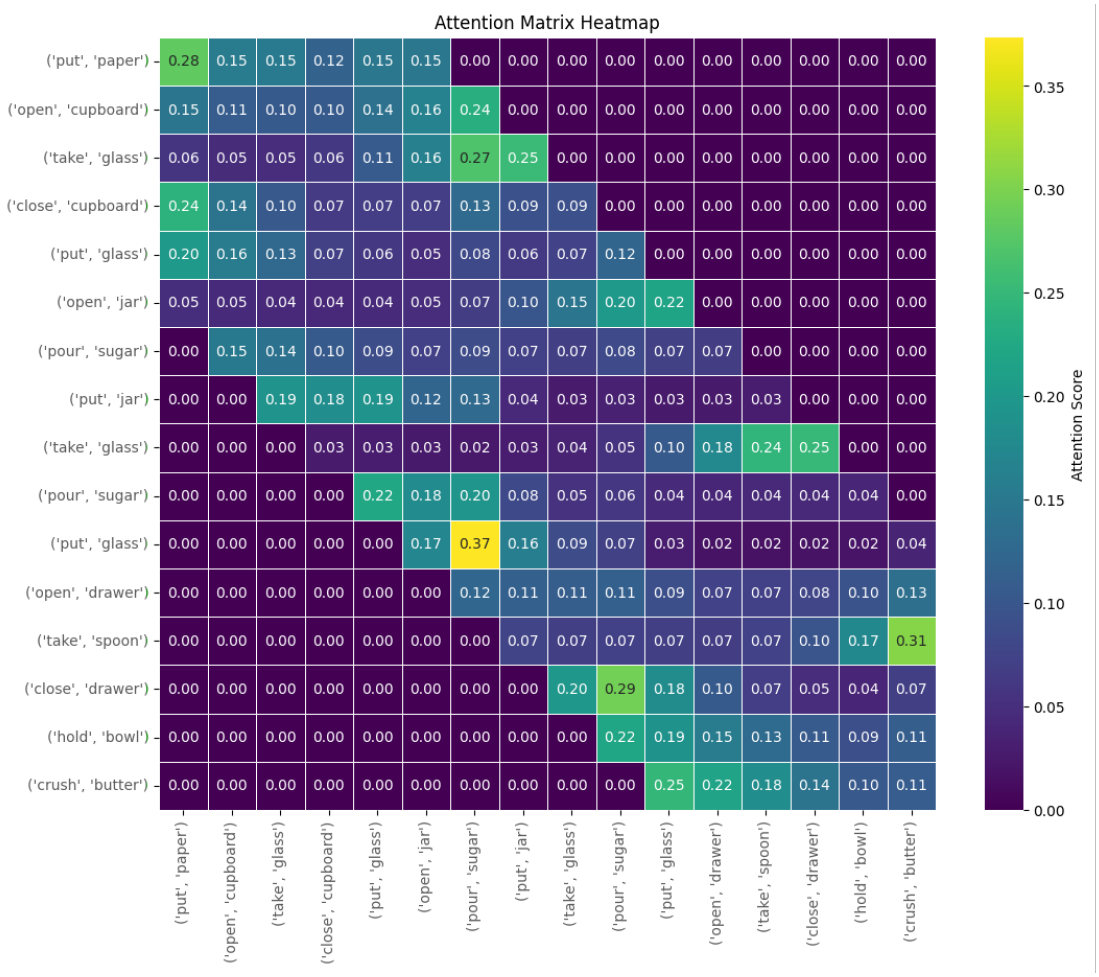


Figure 6.4: RGB Query in Second Cross Attention: Attention matrix representation depicting the interaction between RGB features as the query and Optical Flow features as keys/values in the second cross-attention module. The plot provides insights into the interplay between modalities.

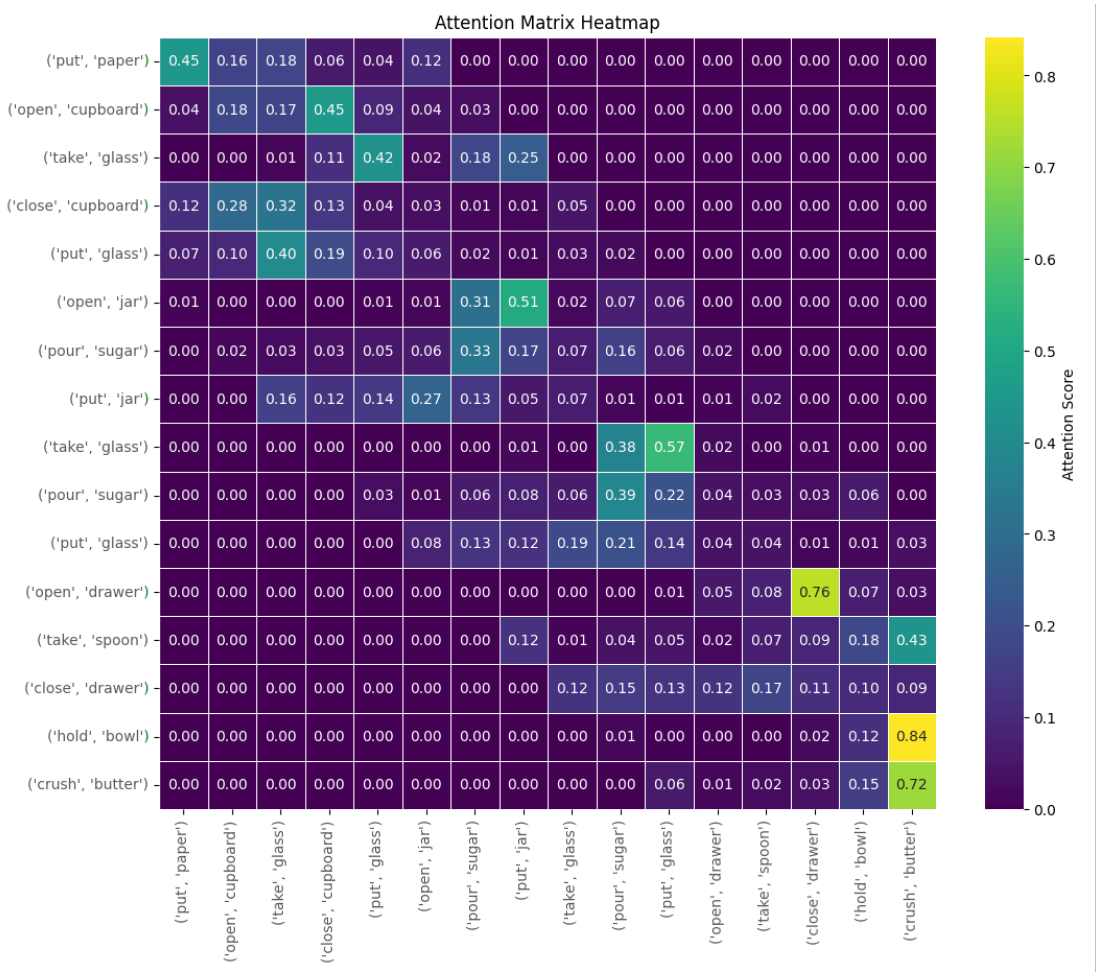


Figure 6.5: Optical Flow Query in Second Cross Attention: Visualization of the attention matrix with Optical Flow features as the query in the second cross-attention module.

6.4 Missing modality

In the context of missing modality, Table 6.7 provides a comprehensive view of the outcomes obtained by dropping RGB and Optical Flow during test time. The results notably showcase a substantial decline in performance, particularly when the information related to RGB is absent.

	Test Modalities		**Top-1 Accuracy**		
	RGB	Optical Flow	Verb	Noun	Action
MARE-Graph(TSM-Slowfast)	✓	✓	69.96	55.26	44.44
MARE-Graph(TSM-Slowfast)	✓	×	66.96	52.72	41.02
MARE-Graph(TSM-Slowfast)	×	✓	61.30	29.99	23.61

Table 6.7: Consequences of missing modalities on model performance. The underlying architecture remains consistent with the previously described setup, utilizing TSM as the backbone for optical flow and SlowFast for RGB.

To ensure a consistent training and testing conditions, strategic implementations such as DropGraph and the reconstruction module, as explained in the preceding paragraph, were incorporated. The results in Table 6.8 highlights the efficacy of these adjustments in mitigating the adverse effects of missing modalities.

Further insights are gleaned through ablation studies, incorporating two additional experiments aimed at unraveling the intricacies of the model's response to missing modalities. In one experiment, the missing information is replaced with a learnable miss token, introduced during training to assist the cross-module in the reconstruction process. The results of this experiment provide valuable insights into the model's ability to adapt and leverage learned tokens to compensate for absent information.

In a parallel ablation experiment within the DropGraph setting, the cross-module is intentionally removed to assess its impact on performance. The results of this experiment underscore the pivotal role played by the cross-attention mechanism in the missing modality scenario.

Among the various implementations, the reconstruction module emerges as the most effective, showcasing its pivotal role in compensating for missing modalities. The model, with these refined adjustments, demonstrates robustness in handling the absence of modalities during test time.

Table 6.9 provides a nuanced examination of the model's behavior under different percentages of modality dropout, offering a comprehensive understanding of its adaptability in varying conditions.

	Test Modalities		Top-1 Accuracy		
	RGB	Optical Flow	Verb	Noun	Action
Upper bound	✓	✓	**69.96**	**55.26**	**44.44**
No Implementation	✓	×	66.96	52.72	41.02
MissingToken	✓	×	67.21	53.05	41.62
DropGraph	✓	×	67.27	53.43	41.89
wo. Cross	✓	×	66.51	52.20	40.77
Rec. Module	✓	×	**67.70**	**54.06**	**42.60**
No Implementation	×	✓	61.30	29.99	23.61
MissingToken	×	✓	64.04	38.47	30.51
DropGraph	×	✓	63.81	39.65	31.19
wo. Cross	×	✓	63.57	38.48	30.19
Rec. Module	×	✓	**64.21**	**39.77**	**31.77**

Table 6.8: The table illustrates the Top-1 accuracy for different configurations addressing missing modalities. The upper bound corresponds to the scenario where both modalities (RGB and Optical Flow) are present during testing, thus representing the maximum achievable performance limit using TSM-Slowfast as the backbone. The "No Implementation" scenario refers to the absence of any strategic adjustments to handle missing modalities during testing. "Wo. Cross" stands for "without Cross-Attention".

Test Modalities	Test Modalities		Top-1 Accuracy		
Missing rate %	RGB	Optical Flow	Verb	Noun	Action
0%	✓	✓	69.96	55.26	44.44
10%	✓	×	69.70	54.90	43.87
50%	✓	×	68.98	54.72	43.63
70%	✓	×	68.49	54.25	43.16
90%	✓	×	68.03	54.11	42.95
100%	✓	×	67.70	54.06	42.60
10%	×	✓	69.15	54.02	43.07
50%	×	✓	67.64	50.05	39.75
70%	×	✓	66.47	46.71	37.21
90%	×	✓	65.18	43.19	34.46
100%	×	✓	64.21	39.77	31.77

Table 6.9: The table presents the performance analysis under various missing rates for RGB and Optical Flow modalities. As the missing rate increases, there is a noticeable decrease in the Top-1 accuracy for both modalities, highlighting the model's sensitivity to missing information, utilizing TSM as the backbone for optical flow and SlowFast for RGB.

Chapter 7

Conclusion

In this thesis, we investigated the advantages of using Graph Neural Networks (GNNs) for high-level temporal aggregation among video clips in order to extract complex patterns within sequences, thus extending temporal aggregation to the frame level offered by the backbones used. By using cross-attention mechanisms, the modes are able to communicate more deeply within the network. Our results demonstrate the flexibility of the implementation, which adapts to different backbones and effectively handles highly heterogeneous features. The use of observation windows for both the temporal reasoning module and cross-attention proves to be a strategic choice to filter out irrelevant information and maximise contextual insights. From our analysis of the results, it is evident that the implementation of MARE-Graph with TSM-SlowFast as backbones has led to a significant improvement in performance compared to the considered baselines for the verb, noun, and action classes. Compared to other architectures, our model demonstrated an average increase of 4.65% for the verb class, 6.17% for the noun class, and 7.20% for the action class. This underscores the effectiveness of our approach in capturing the complexities of human activities in egocentric video sequences, enabling a more thorough and accurate analysis of user actions.

Finally, to further underscore the effectiveness of our approach in leveraging multimodal synergies, we conducted experiments to assess the model's performance under scenarios where specific modalities were unavailable during testing due to computational constraints or efficiency requirements. Our cross-modal interaction mechanism augmented by strategic implementations such as DropGraph and the reconstruction module, demonstrated robustness in learning representations, showcasing resilience in the face of potential modality loss. This step aims to drive attention towards realistic settings that challenge the effectiveness and robustness of a solution, moving beyond settings that often diverge from real-world scenarios.

Bibliography

[1] Karen Simonyan and Andrew Zisserman. «Two-Stream Convolutional Networks for Action Recognition in Videos». In: (2014). arXiv: 1406.2199 [cs.CV] (cit. on pp. 1, 14, 26).

[2] Alireza Fathi, Ali Farhadi, and James M. Rehg. «Understanding egocentric activities». In: (2011), pp. 407–414. DOI: 10.1109/ICCV.2011.6126269 (cit. on p. 1).

[3] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. «EGO-TOPO: Environment Affordances from Egocentric Video». In: *CoRR* abs/2001.04583 (2020). arXiv: 2001.04583. URL: https://arxiv.org/abs/2001.04583 (cit. on p. 1).

[4] Taein Kwon, Bugra Tekin, Jan Stuhmer, Federica Bogo, and Marc Pollefeys. «H2O: Two Hands Manipulating Objects for First Person Interaction Recognition». In: (2021). arXiv: 2104.11181 [cs.CV] (cit. on p. 1).

[5] Karen Simonyan and Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». In: (2015). arXiv: 1409.1556 [cs.CV] (cit. on p. 3).

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep Residual Learning for Image Recognition». In: (2015). arXiv: 1512.03385 [cs.CV] (cit. on pp. 3, 23, 48).

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV] (cit. on p. 3).

[8] Frank Rosenblatt. «The perceptron: a probabilistic model for information storage and organization in the brain.» In: *Psychological review* 65 6 (1958), pp. 386–408. URL: https://api.semanticscholar.org/CorpusID:12781225 (cit. on p. 4).

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. «Gradient-based learning applied to document recognition». In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791 (cit. on p. 5).

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention Is All You Need». In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762 (cit. on pp. 7, 8).

[11] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV] (cit. on p. 10).

[12] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. «Graph neural networks: A review of methods and applications». In: *AI Open* 1 (2020), pp. 57–81. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012 (cit. on p. 11).

[13] Thomas N. Kipf and Max Welling. «Semi-Supervised Classification with Graph Convolutional Networks». In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907 (cit. on p. 13).

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. «Inductive Representation Learning on Large Graphs». In: (2018). arXiv: 1706.02216 [cs.SI] (cit. on p. 13).

[15] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9.8 (1997), pp. 1735–1780 (cit. on p. 13).

[16] Alex Sherstinsky. «Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network». In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. URL: http://dx.doi.org/10.1016/j.physd.2019.132306 (cit. on p. 14).

[17] Joao Carreira and Andrew Zisserman. «Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset». In: (2018). arXiv: 1705.07750 [cs.CV] (cit. on pp. 14, 22, 25).

[18] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. *Omnivore: A Single Model for Many Visual Modalities*. 2022. arXiv: 2201.08377 [cs.CV] (cit. on p. 14).

[19] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. «SlowFast Networks for Video Recognition». In: (2019). arXiv: 1812.03982 [cs.CV] (cit. on pp. 14, 29, 36, 39, 48).

[20] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. «Temporal Segment Networks: Towards Good Practices for Deep Action Recognition». In: (2016). arXiv: 1608.00859 [cs.CV] (cit. on pp. 14, 26–28, 39, 48).

[21] Evangelos Kazakos, Arsha Nagrani, Andrew Zisserman, and Dima Damen. «EPIC-Fusion: Audio-Visual Temporal Binding for Egocentric Action Recognition». In: (2019). arXiv: 1908.08498 [cs.CV] (cit. on pp. 14, 27).

[22] Santosh Kumar Yadav, Kamlesh Tiwari, Hari Mohan Pandey, and Shaik Ali Akbar. «A review of multimodal human activity recognition with special emphasis on classification, applications, challenges and future directions». In: *Knowledge-Based Systems* 223 (2021), p. 106970. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2021.106970. URL: https://www.sciencedirect.com/science/article/pii/S0950705121002331 (cit. on p. 15).

[23] Weiyao Wang, Du Tran, and Matt Feiszli. *What Makes Training Multi-Modal Classification Networks Hard?* 2020. arXiv: 1905.12681 [cs.CV] (cit. on p. 15).

[24] Peng Xu, Xiatian Zhu, and David A. Clifton. *Multimodal Learning with Transformers: A Survey*. 2023. arXiv: 2206.06488 [cs.CV] (cit. on p. 16).

[25] Junyang Lin, An Yang, Yichang Zhang, Jie Liu, Jingren Zhou, and Hongxia Yang. *InterBERT: Vision-and-Language Interaction for Multi-modal Pretraining*. 2021. arXiv: 2003.13198 [cs.CL] (cit. on p. 17).

[26] Chaoqi Chen, Yushuang Wu, Qiyuan Dai, Hong-Yu Zhou, Mutian Xu, Sibei Yang, Xiaoguang Han, and Yizhou Yu. *A Survey on Graph Neural Networks and Graph Transformers in Computer Vision: A Task-Oriented Perspective*. 2022. arXiv: 2209.13232 [cs.CV] (cit. on p. 20).

[27] Xiaolong Wang and Abhinav Gupta. *Videos as Space-Time Region Graphs*. 2018. arXiv: 1806.01810 [cs.CV] (cit. on p. 20).

[28] Yangjun Ou, Li Mi, and Zhenzhong Chen. «Object-Relation Reasoning Graph for Action Recognition». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 20133–20142 (cit. on pp. 20, 23).

[29] Jingran Zhang, Fumin Shen, Xing Xu, and Heng Tao Shen. «Temporal Reasoning Graph for Activity Recognition». In: *IEEE Transactions on Image Processing* 29 (2020), pp. 5491–5506. DOI: 10.1109/TIP.2020.2985219 (cit. on p. 20).

[30] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. «I Know the Relationships: Zero-Shot Action Recognition via Two-Stream Graph Convolutional Networks and Knowledge Graphs». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 8303–8311. DOI: 10.1609/aaai.v33i01.33018303. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4843 (cit. on p. 20).

[31] Yifan Zhao, Ke Yan, Feiyue Huang, and Jia Li. «Graph-Based High-Order Relation Discovery for Fine-Grained Recognition». In: (June 2021), pp. 15079–15088 (cit. on p. 20).

[32] Sijie Yan, Yuanjun Xiong, and Dahua Lin. «Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition». In: (2018). arXiv: 1801.07455 [cs.CV] (cit. on pp. 20, 21).

[33] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. *Two-Stream Adaptive Graph Convolutional Networks for Skeleton-Based Action Recognition*. 2019. arXiv: 1805.07694 [cs.CV] (cit. on pp. 20, 21).

[34] Bin Li, Xi Li, Zhongfei Zhang, and Fei Wu. «Spatio-Temporal Graph Routing for Skeleton-Based Action Recognition». In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 8561–8568. DOI: 10.1609/aaai.v33i01.33018561. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4875 (cit. on p. 20).

[35] Rui Zhao, Kang Wang, and Qiang Ji. «Bayesian Graph Convolution LSTM for Skeleton Based Action Recognition». In: (Oct. 2019). DOI: 10.1109/ICCV.2019.00698 (cit. on p. 20).

[36] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. «Skeleton-Based Action Recognition With Directed Graph Neural Networks». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019 (cit. on p. 20).

[37] Ziyu Liu, Hongwen Zhang, Zhenghao Chen, Zhiyong Wang, and Wanli Ouyang. *Disentangling and Unifying Graph Convolutions for Skeleton-Based Action Recognition*. 2020. arXiv: 2003.14111 [cs.CV] (cit. on pp. 20, 21).

[38] Chenyang Si, Wentao Chen, Wei Wang, Liang Wang, and Tieniu Tan. *An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition*. 2019. arXiv: 1902.09130 [cs.CV] (cit. on p. 20).

[39] Ke Cheng, Yifan Zhang, Xiangyu He, Weihan Chen, Jian Cheng, and Hanqing Lu. «Skeleton-Based Action Recognition With Shift Graph Convolutional Network». In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 180–189. DOI: 10.1109/CVPR42600.2020.00026 (cit. on pp. 20, 21).

[40] Ke Cheng, Yifan Zhang, Congqi Cao, Lei Shi, Jian Cheng, and Hanqing Lu. «Decoupling GCN with DropGraph Module for Skeleton-Based Action Recognition». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2020 (cit. on p. 20).

[41] Yunho Jeon and Junmo Kim. *Constructing Fast Network through Deconstruction of Convolution*. 2018. arXiv: 1806.07370 [cs.CV] (cit. on p. 21).

[42] Svebor Karaman, Lorenzo Seidenari, and Alberto Del Bimbo. «Fast saliency based pooling of fisher encoded dense trajectories». In: *ECCV Thumos Workshop*. Vol. 1. 2. 2014, p. 5 (cit. on p. 22).

[43] Colin Lea, Austin Reiter, Rene Vidal, and Gregory D. Hager. *Segmental Spatiotemporal CNNs for Fine-grained Action Segmentation*. 2016. arXiv: 1602.02995 [cs.CV] (cit. on p. 22).

[44] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. *Connectionist Temporal Modeling for Weakly Supervised Action Labeling*. 2016. arXiv: 1607.08584 [cs.CV] (cit. on p. 22).

[45] Dima Damen et al. *Scaling Egocentric Vision: The EPIC-KITCHENS Dataset*. 2018. arXiv: 1804.02748 [cs.CV] (cit. on p. 22).

[46] Yifei Huang, Yusuke Sugano, and Yoichi Sato. «Improving action segmentation via graph-based temporal reasoning». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 14024–14034 (cit. on p. 22).

[47] Yin Li, Miao Liu, and James M. Rehg. *In the Eye of the Beholder: Gaze and Actions in First Person Video*. 2020. arXiv: 2006.00626 [cs.CV] (cit. on p. 22).

[48] Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. *Temporal Relational Reasoning in Videos*. 2018. arXiv: 1711.08496 [cs.CV] (cit. on pp. 28, 36).

[49] Ji Lin, Chuang Gan, and Song Han. *TSM: Temporal Shift Module for Efficient Video Understanding*. 2019. arXiv: 1811.08383 [cs.CV] (cit. on pp. 30, 39, 48).

[50] Reza Azad, Nika Khosravi, Mohammad Dehghanmanshadi, Julien Cohen-Adad, and Dorit Merhof. *Medical Image Segmentation on MRI Images with Missing Modalities: A Review*. 2022. arXiv: 2203.06217 [eess.IV] (cit. on p. 31).

[51] Guido Montufar. *Restricted Boltzmann Machines: Introduction and Review*. 2018. arXiv: 1806.07066 [cs.LG] (cit. on p. 31).

[52] Mohammad Havaei, Nicolas Guizard, Nicolas Chapados, and Yoshua Bengio. *HeMIS: Hetero-Modal Image Segmentation*. 2016. arXiv: 1607.05194 [cs.CV] (cit. on p. 31).

[53] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015. arXiv: 1503.02531 [stat.ML] (cit. on p. 32).

[54] Reza Azad, Nika Khosravi, and Dorit Merhof. *SMU-Net: Style matching U-Net for brain tumor segmentation with missing modalities*. 2022. arXiv: 2204.02961 [cs.CV] (cit. on p. 32).

[55] Michael Tschannen, Josip Djolonga, Paul K. Rubenstein, Sylvain Gelly, and Mario Lucic. *On Mutual Information Maximization for Representation Learning*. 2020. arXiv: 1907.13625 [cs.LG] (cit. on p. 32).

[56] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML] (cit. on p. 33).

[57] Teodora Pandeva and Matthias Schubert. *MMGAN: Generative Adversarial Networks for Multi-Modal Distributions*. 2019. arXiv: 1911.06663 [cs.LG] (cit. on p. 33).

[58] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV] (cit. on p. 33).

[59] Dima Damen et al. «Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100». In: *International Journal of Computer Vision (IJCV)* 130 (2022), pp. 33–55. URL: https://doi.org/10.1007/s11263-021-01531-2 (cit. on pp. 34, 36, 38, 49).

[60] Dima Damen et al. «Scaling Egocentric Vision: The EPIC-KITCHENS Dataset». In: *European Conference on Computer Vision (ECCV)*. 2018 (cit. on p. 34).

[61] Kristen Grauman et al. «Ego4D: Around the World in 3,000 Hours of Egocentric Video». In: (2022). arXiv: 2110.07058 [cs.CV] (cit. on p. 38).

[62] Cheng-Zhi Anna Huang et al. *Music Transformer*. 2018. arXiv: 1809.04281 [cs.LG] (cit. on p. 44).

[63] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf (cit. on p. 48).

[64] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 48).

[65] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. «TV-L1 Optical Flow Estimation». In: *Image Processing On Line* 3 (2013). https://doi.org/10.5201/ipol.2013.26, pp. 137–150 (cit. on p. 48).

[66] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. *FractalNet: Ultra-Deep Neural Networks without Residuals*. 2017. arXiv: 1605.07648 [cs.CV] (cit. on p. 49).