Technical Summary: Frame Order Matters: A Temporal Sequence-Aware Model for Few-Shot Action Recognition

Kai-Yu Lu

2025/10/19

1 Research Problem and Motivation

Few-shot action recognition is concerned with recognizing human actions in videos when only a few labeled examples per class are available. The task requires learning robust video representations that capture both spatial appearance and temporal dynamics under severe data scarcity. The study focuses on the following central research problem: how to construct a few-shot action recognition model that is explicitly sensitive to frame order, while leveraging powerful pre-trained vision language backbones.

Existing few-shot action recognition frameworks often rely on frame-level features extracted from image backbones and aggregate these features through temporal pooling, temporal relation modeling or frame matching. However, many of these methods treat frames as largely exchangeable and do not explicitly enforce sensitivity to temporal order. In practice, rearranging video frames can change the semantic category of an action. For example, a sequence that corresponds to *catching a ball* differs semantically from *throwing a ball* primarily through temporal order rather than static content. Empirical observations show that several previous models produce nearly identical predictions when the input video is reversed, indicating that temporal order is not sufficiently encoded.

At the same time, the emergence of large-scale contrastive vision language models such as Contrastive Language Image Pre-training (CLIP) provides strong spatial and semantic representations that can be transferred to video. Prior work has adapted CLIP to few-shot action recognition by fine-tuning the backbone or introducing parameter-efficient adapters. These approaches demonstrate strong performance but still emphasize frame relationships more than explicit modeling of sequential order. Furthermore, few-shot settings provide only a limited number of labeled videos per class, so purely visual prototypes can be semantically weak and fail to fully exploit class-level knowledge.

The motivation of the study is therefore threefold. First, there is a need to design a temporal modeling mechanism that is explicitly sequence-aware and sensitive to frame order changes. Second, there is a need to enrich class prototypes with external semantic information so that few-shot learning can benefit from textual descriptions beyond the small labeled video set. Third, there is a need for a robust frame matching mechanism that can mitigate the influence of background or class-irrelevant frames when comparing support and query videos.

The main goals are: to develop a temporal sequence-aware adapter that integrates temporal dynamics into a pre-trained vision backbone, to construct enriched multimodal class prototypes by combining visual and textual information, and to design an unbalanced optimal transport based matching strategy that selectively downweights noisy frames during few-shot matching. The study aims to demonstrate that this combination yields state-of-the-art performance on standard few-shot action recognition benchmarks and to provide evidence that the resulting model is genuinely sensitive to temporal order.

2 Related Work

2.1 Few-Shot Action Recognition

Few-shot action recognition extends few-shot learning to the video domain. Early methods focus on metric learning, where support and query videos are embedded into a feature space and compared through a learned metric. Representative approaches include methods based on temporal alignment, multi-frame matching, optimal transport and set matching, which attempt to align temporal fragments between support and query videos.

A second line of work emphasizes spatial motion modeling. These methods learn global motion patterns, enrich local patch features for joint spatial temporal modeling or explicitly extract motion dynamics within a unified network. They often rely on 2D or 3D convolutional encoders pre-trained on large-scale image or video datasets and then adapt these encoders to the few-shot regime.

More recently, several approaches leverage large vision language models, particularly CLIP. CLIP-based methods transfer pre-trained image encoders to the video domain and construct prototypes that combine visual features with textual prompts. These methods show that integrating class-level textual semantics into the prototype space significantly improves few-shot action recognition performance.

2.2 Parameter-Efficient Fine-Tuning

Parameter Efficient Fine Tuning (PEFT) aims to adapt large pre-trained models using a small number of additional parameters while keeping the backbone frozen. In computer vision, typical PEFT strategies introduce prompts, adapters or lightweight modules that modulate intermediate features. For vision language models, visual prompt tuning and textual prompt tuning inject learnable prompt tokens into the image or text streams, while adapter based approaches add compact bottleneck modules into transformer blocks.

In the video domain, PEFT schemes for action recognition include temporal adapters based on 3D convolutions and attention based temporal modules that reuse the attention layers of pre-trained transformers. Recent CLIP-based few-shot action recognition approaches employ multimodal adapters guided by textual information. While these methods capture temporal dependencies and integrate text, many of them still treat frames as a set and do not explicitly encode frame order as a first-class temporal signal.

2.3 Perceiver-based Structures

The Perceiver architecture introduces an asymmetric attention mechanism that maps high-volume multimodal inputs into a small set of latent tokens. This enables efficient processing, feature compression and cross-modal communication. Perceiver-based modules have been used to resample dense feature tokens, to bridge multi-modal feature spaces and to aggregate video features in large multimodal models.

In the context of few-shot action recognition, a Perceiver-based adapter provides an attractive way to inject temporal information into a pre-trained spatial backbone. However, a simple Perceiver adapter with fully learnable temporal queries may homogenize temporal features and suffer from information leakage between future and past frames. The presented study therefore extends the Perceiver idea into a sequential Perceiver adapter that models temporal order in a recurrent manner.

3 Dataset Construction

3.1 Benchmarks

The experimental evaluation covers five few-shot action recognition benchmarks. These datasets differ in scene complexity, temporal dependence and scale.

Dataset	Classes	Train / Val / Test	Modality	Focus	Characteristics
HMDB-51	51	31 / 10 / 10	RGB video	Scene understanding	Human actions in movies and web videos, diverse backgrounds and camera viewpoints.
UCF101	101	70 / 10 / 21	RGB video	Scene understanding	Sports, musical instrument playing and daily activities from online videos.
Kinetics subset	100	64 / 12 / 24	RGB video	Scene understanding	Subset of a large-scale action dataset with a wide range of everyday actions.
SSv2-Small	174	64 / 12 / 24	RGB video	Temporal reasoning	Temporally abstract labels with a small number of videos per class.
SSv2-Full	174	64 / 12 / 24	RGB video	Temporal reasoning	Same label set as SSv2-Small, with approximately ten times more training videos per class.

Table 1: Overview of the few-shot action recognition benchmarks used for evaluation. The first three datasets emphasize scene-level understanding, while Something-Something V2 (SSv2) requires stronger temporal modeling.

HMDB-51 and UCF101 contain short trimmed videos of human actions. Both are widely used in traditional action recognition and few-shot video classification. They primarily evaluate the model's ability to recognize semantic categories from relatively rich scene cues. The Kinetics subset is derived from a larger action recognition dataset and is used to assess cross-dataset generalization when training on one dataset and evaluating on another.

Something-Something V2 (SSv2) is designed to focus on temporal common-sense reasoning. Labels describe object-centric interactions such as pushing, pulling or pretending to perform an action. Single frames often appear ambiguous, and correct recognition requires understanding how objects move or change over time. The SSv2-Small split provides limited training samples per class, while SSv2-Full provides a significantly larger number of training videos, enabling analysis of how the proposed model scales with more data.

3.2 Preprocessing and Episode Sampling

Videos are uniformly sampled into a fixed number of frames. Each input video is represented as $v \in \mathbb{R}^{T \times H \times W \times 3}$, where T is the number of sampled frames, H is the frame height, W is the frame width and the last dimension corresponds to RGB channels. In the implementation, $T = 8$ frames are sampled per video and each frame is cropped or resized to a spatial resolution of 224×224.

Few-shot tasks are constructed using episodic sampling. For each episode, a subset of classes is selected and split into support and query sets according to an N-way K-shot protocol. During evaluation, 10 000 episodes are sampled for each dataset. The reported performance is the mean accuracy over all episodes for both 5-way 1-shot and 5-way 5-shot settings.

4 Query Protocol and Task Definitions

4.1 Few-Shot Episodic Task

The video dataset is divided into non-overlapping training and testing splits. Few-shot learning is performed through episodic training. An episode corresponds to an N-way K-shot classification task. In each episode, a set of N action classes is sampled from the training split. For each class, K labeled videos are sampled to form the support set and a number of unlabeled videos are sampled to form the query set.

The support set is denoted as

$$S = \{(v_j^{(c)}, y_j^{(c)}) \mid c = 1, \ldots, N,\ j = 1, \ldots, K\}$$

where $v_j^{(c)}$ is the j-th support video of class c and $y_j^{(c)}$ is the corresponding class label. The query set is denoted as

$$Q = \{(v_m^{\mathrm{q}}, y_m^{\mathrm{q}}) \mid m = 1, \ldots, M\}$$

where M is the number of query videos in the episode. During training, query labels are used to optimize the model. During testing, query labels are used only for evaluation.

The goal in each episode is to predict the class $\hat{y}$ for each query video v_m^{q} using only the support set of that episode. Performance is measured by the proportion of correctly classified query videos.

4.2 Distance Metric and Prediction

Each video is encoded into a sequence of frame-level feature vectors by the proposed temporal sequence-aware encoder. For a query video q and a support video s_i belonging to class i, the encoder produces latent representations that are aggregated by a metric function $M(\cdot,\cdot)$ into a scalar distance. This is formalized in Equation (1).

$$\mathrm{Dis}(q, s_i) = M(\tilde{f}_q, f_i^s) \tag{1}$$

In Equation (1), $\mathrm{Dis}(q, s_i)$ denotes the distance between the query video q and the i-th class prototype derived from its support videos. The symbol $\tilde{f}_q$ represents the aggregated feature vector of the query video after temporal modeling, and f_i^s represents the prototype feature for class i. The function $M(\cdot,\cdot)$ is a metric function that takes two feature vectors and outputs a scalar distance, for example a distance based on unbalanced optimal transport. The purpose of this metric is to map query-support pairs to a scalar that reflects their similarity; smaller values indicate higher similarity.

Given distances to all class prototypes, the few-shot prediction distribution over classes for a query video q is defined in Equation (2).

$$p_{\hat{y}=i|q}^{\mathrm{fsl}} = \frac{\exp\big(\mathrm{Dis}(q, s_i)\big)}{\sum_{j=1}^{N} \exp\big(\mathrm{Dis}(q, s_j)\big)} \tag{2}$$

In Equation (2), $p_{\hat{y}=i|q}^{\mathrm{fsl}}$ denotes the probability that query video q belongs to class i according to the few-shot branch. The numerator exponentiates the distance between q and prototype s_i, and the denominator sums these exponentials over all classes. In practice, the implementation uses a monotonically decreasing transformation of distance, so larger similarity corresponds to larger logits; the softmax operation then converts these logits into a normalized probability distribution.

The model also uses a zero-shot prediction branch based on CLIP, which computes a probability distribution $p_{\hat{y}=i|q}^{\mathrm{zsl}}$ for each query video by aligning video features with textual prompts. The final prediction distribution combines the few-shot and zero-shot predictions through a multiplicative fusion controlled by a scalar parameter $\lambda \in [0, 1]$, as shown in Equation (3).

$$p_{\hat{y}=i|q} = \big(p_{\hat{y}=i|q}^{\mathrm{fsl}}\big)^{\lambda} \cdot \big(p_{\hat{y}=i|q}^{\mathrm{zsl}}\big)^{1-\lambda} \tag{3}$$

In Equation (3), $p_{\hat{y}=i|q}$ is the fused prediction probability for class i. The exponent λ weighs the contribution of the few-shot probability and the exponent $1-\lambda$ weighs that of the zero-shot probability. When λ is close to one, the model relies more heavily on few-shot evidence; when λ is smaller, zero-shot knowledge from CLIP plays a larger role. The product is re-normalized implicitly when computing the predicted label.

The training objective combines two cross-entropy losses, one for the few-shot branch and one for the zero-shot branch, as defined in Equation (4).

$$\mathcal{L} = \mathrm{CE}\big(p^{\mathrm{fsl}}_{\hat{y}=i|q}, g^{\mathrm{fsl}}_{\mathrm{t}}\big) + \mathrm{CE}\big(p^{\mathrm{zsl}}_{\hat{y}=i|q}, g^{\mathrm{zsl}}_{\mathrm{t}}\big) \tag{4}$$

In Equation (4), $\mathcal{L}$ is the total loss. The function $\mathrm{CE}(\cdot,\cdot)$ is the cross-entropy loss. The term $g^{\mathrm{fsl}}_{\mathrm{t}}$ denotes the ground truth class distribution for the query under the few-shot branch, which is a one-hot vector over classes. The term $g^{\mathrm{zsl}}_{\mathrm{t}}$ denotes the corresponding ground truth distribution for the zero-shot branch. Minimizing $\mathcal{L}$ encourages both branches to assign high probability to the true class, thereby aligning video features with few-shot prototypes and textual prompts simultaneously.

5 Modeling Approach

5.1 Perceiver-based Video Encoder

The proposed Temporal Sequence-Aware Model (TSAM) builds upon a pre-trained vision transformer backbone. Given an input video $v \in \mathbb{R}^{T\times H\times W\times 3}$, each frame is processed by the frozen CLIP image encoder, which is implemented as a Vision Transformer (ViT) with L transformer blocks. The intermediate representation at block i is denoted as $V_i \in \mathbb{R}^{T\times U\times D}$, where T is the number of frames, U is the number of spatial tokens per frame and D is the feature dimension.

The first J transformer blocks of the backbone are kept fixed and act as a pure spatial encoder. For the remaining $L - J$ blocks, a perceiver adapter is added in parallel to each spatial encoder. The purpose of the adapter is to compress spatial information into a temporal representation that can capture sequential dynamics. The adapter uses two projection modules: a spatial down-sampling module and a temporal down-sampling module. These modules map high-dimensional feature maps to compact spatial and temporal embeddings.

Equation (5) formalizes this step.

$$V^s_i = \mathrm{SpatialD}\big(\mathrm{VisualB}(V_i)\big), \quad V^t_i = \mathrm{TemporalD}(V_i) \tag{5}$$

In Equation (5), V^s_i denotes the spatial feature embeddings at block i after applying the visual block $\mathrm{VisualB}(\cdot)$ followed by the spatial down-sampling function $\mathrm{SpatialD}(\cdot)$. The spatial down-sampling function reduces the spatial dimension U into a compact representation for each frame, for example by averaging or projecting patch tokens. The term V^t_i denotes the temporal embeddings obtained by applying the temporal down-sampling function $\mathrm{TemporalD}(\cdot)$ to V_i. This function reduces the temporal dimension T into an initial temporal token per frame or into a summarized temporal embedding. These embeddings serve as inputs to the sequential perceiver adapter.

5.2 Sequential Perceiver Adapter

A straightforward perceiver adapter with fully learnable temporal queries is prone to two issues: the temporal latent space may become too homogeneous when trained on limited data, and unstructured attention can cause information from future frames to leak into past frame representations. To address these issues, the model introduces a sequential perceiver adapter that treats the temporal query as a recurrent state evolving along the timeline.

The adapter processes frames in order. For frame index k, it takes the previous temporal embedding F^t_{k-1} as the query and the current spatial embedding V^s_k as key and value inputs. The cross-attention module $\mathrm{CrossAtt}(\cdot)$ updates the temporal query by aggregating information from the current frame. A residual term proportional to the current temporal embedding V_k^t is added to maintain frame-specific temporal information. This procedure is described in Equation (6).

$$F_k^t = \mathrm{CrossAtt}\big(F_{k-1}^t, V_k^s, V_k^s\big) + \beta \cdot V_k^t \tag{6}$$

In Equation (6), F_k^t is the updated temporal embedding after processing frame k. The variable F_{k-1}^t is the temporal embedding from the previous frame, which plays the role of the query in the cross-attention operation. The matrix V_k^s is the spatial embedding of the k-th frame, used as both keys and values. The scalar β is a learnable or fixed injection ratio that balances the contribution of the cross-attention output and the original temporal embedding V_k^t. For the first frame, the initial temporal query F_{-1}^t is set to the corresponding temporal embedding V_0^t. The cross-attention function computes attention weights between the query and spatial tokens and aggregates spatial features into a single temporal representation. The recurrent structure of Equation (6) ensures that temporal information is updated sequentially, which preserves the inherent order of frames and prevents future frame information from directly influencing earlier temporal states.

After performing this recurrent update across all frames, the final temporal embedding sequence F^t is fused back into the visual backbone. The fusion is applied at the output of block i, before feeding into block $i+1$, as shown in Equation (7).

$$V_{i+1} = \mathrm{VisualB}(V_i) + \alpha \cdot F^t \tag{7}$$

In Equation (7), V_{i+1} is the input representation for the next transformer block. The term $\mathrm{VisualB}(V_i)$ is the standard output of the visual block applied to V_i without temporal augmentation. The term F^t is the sequence of temporal embeddings produced by the sequential perceiver. The scalar α is a scaling factor that controls the intensity of temporal information injected into the spatial features. By adding $\alpha \cdot F^t$ to the visual features, the model enriches frame-level representations with sequential temporal context while preserving the original spatial backbone behavior.

5.3 Textual Corpus Enhancement

In few-shot scenarios, visual prototypes built from a small number of support videos may not sufficiently capture class semantics. To address this limitation, the model augments visual prototypes with class-level textual information. For each action class, a textual corpus is generated using a large language model. The corpus contains multiple descriptions related to the action, such as typical motions, objects involved and contextual cues. For temporally abstract datasets like SSv2, specific objects are replaced with generic terms such as "object" to keep descriptions aligned with the dataset's abstract labels.

The textual corpus for a class is concatenated into a single sentence and encoded using the CLIP text encoder, denoted as $\mathrm{TextEnc}(\cdot)$. The visual prototype for class n is denoted as f_n^v, obtained by aggregating the encoded support videos. The multimodal prototype f_n is then constructed by summing the visual prototype with its textual embedding, as in Equation (8).

$$f_n = f_n^v + \mathrm{TextEnc}\big(\mathrm{Corpus}(\mathrm{class}_n)\big) \tag{8}$$

In Equation (8), f_n is the augmented prototype for class n. The term f_n^v represents the class-level visual feature obtained from support videos after passing through the temporal sequence-aware encoder and a prototype aggregation module. The function $\mathrm{Corpus}(\mathrm{class}_n)$ returns the textual corpus string for class n generated by the language model. The function $\mathrm{TextEnc}(\cdot)$ encodes this text into a feature vector in the same embedding space as CLIP image features. Adding the textual embedding to the visual prototype injects high-level semantic information into the class representation and reduces ambiguity caused by limited visual data. The resulting multimodal prototypes are further refined by a transformer-based prototype merging module that allows prototypes to interact and improve discriminability.

5.4 Unbalanced Optimal Transport Matching

The similarity between support and query videos is computed using an unbalanced optimal transport (UOT) formulation. Each video is represented as a set of frame-level latent vectors. For a support video S and a query video Q, a cost matrix $D \in \mathbb{R}^{N\times N}$ is defined, where each entry $d(s_i, q_j)$ measures the Euclidean distance between the i-th support frame feature and the j-th query frame feature. The goal is to find a transport plan $T \in \mathbb{R}^{N\times N}$ that specifies how much mass is transported between each pair of frames.

In standard optimal transport, the transport plan is constrained to satisfy marginal constraints $(T\mathbf{1}) = \alpha$ and $(T^\top \mathbf{1}) = \beta$, where α and β are prescribed nonnegative vectors and $\mathbf{1}$ is a vector of ones. The inner product $\langle D, T\rangle$ denotes the total transport cost. An entropic regularization term $H(T)$ is included to make the optimization smoother and computationally efficient. This leads to the regularized optimal transport objective in Equation (9).

$$\mathrm{Dis}(Q,S) = \min_{T\in\Pi(\alpha,\beta)} \langle D,T\rangle + \lambda H(T) \tag{9}$$

In Equation (9), $\mathrm{Dis}(Q,S)$ denotes the distance between the query video Q and the support video S in the optimal transport metric. The matrix D contains pairwise frame distances, and the transport plan T determines how frame features are matched. The set $\Pi(\alpha,\beta)$ is the set of transport plans whose row and column sums match the prescribed marginals α and β. The scalar λ controls the strength of the entropic regularization term $H(T)$, which is the negative entropy of T. Larger λ encourages more diffuse transport plans, which stabilizes optimization and allows the use of iterative algorithms such as the Sinkhorn algorithm.

However, video frames often contain background noise or redundant frames that do not need to be matched in a strict one-to-one manner. To model this, the distance is reformulated using unbalanced optimal transport, which relaxes the marginal constraints and adds Kullback–Leibler divergence penalties that measure the deviation between the actual marginals and a reference distribution. This is given in Equation (10).

$$\mathrm{Dis}(Q,S) = \min_{T\in\Pi(\alpha,\beta)} \langle D,T\rangle + \lambda H(T) + \tau \mathrm{KL}(T\mathbf{1} \parallel N) + \tau \mathrm{KL}(T^\top \mathbf{1} \parallel N) \tag{10}$$

In Equation (10), the terms $\mathrm{KL}(T\mathbf{1} \parallel N)$ and $\mathrm{KL}(T^\top \mathbf{1} \parallel N)$ are Kullback–Leibler divergences between the row sums and column sums of the transport plan and a reference distribution N derived from cross-inner products between support and query features. The scalar τ is a regularization coefficient that controls how strongly deviations from the reference marginals are penalized. When τ is positive, the optimization is allowed to partially violate the strict marginal constraints, effectively enabling the transport plan to ignore some frames or assign them small mass if they are redundant or noisy. This soft constraint is particularly suitable for video matching, where some frames may correspond to background or non-discriminative content. The distance $\mathrm{Dis}(Q,S)$ defined in Equation (10) is then used as the metric $M(\cdot,\cdot)$ in Equation (1).

The UOT problem is solved approximately using a generalized Sinkhorn algorithm that alternates scaling of rows and columns under the entropic and KL regularization terms. The resulting transport plan provides a structured alignment between support and query frames that simultaneously respects temporal order through the encoder and downweights class-irrelevant frames.

6 Empirical Results

6.1 Experimental Setup

6.1.1 Model Variants and Baselines

The TSAM framework is instantiated with two CLIP backbones: CLIP-ResNet50 and CLIP-ViT-B/16. In both cases, the image encoder is frozen and only the sequential perceiver adapters and associated modules are trained. The study compares TSAM against twelve state-of-the-art few-shot action recognition methods, including classical metric-based methods and CLIP-based methods. Representative baselines include:

- CMN, OTAM, AmeFuNet, TRX, SPRN, HyRSM, TA2N with sampler, MoLo, MGCSM and SA-CT, all based on ResNet pre-training.
- CLIP-FSAR, which fine-tunes CLIP for few-shot action recognition using full backbone tuning or partial tuning.
- MA-CLIP, a CLIP-based method with a textual guided adapter structure that integrates text into CLIP for few-shot action recognition.

6.1.2 Hyperparameters and Implementation Details

Each video is uniformly sampled into 8 frames and resized to 224×224. For ViT-based backbones, the last several transformer blocks are augmented with sequential perceiver adapters, while the earlier blocks remain frozen. The number of adapters is varied in ablation studies, and a configuration with four adapters in the last blocks provides a favorable trade-off between performance and parameter efficiency.

Training uses stochastic gradient descent as the optimizer. The pre-trained CLIP backbone is frozen, and only the adapter parameters and small additional modules are updated. Few-shot episodes are sampled for 5-way 1-shot and 5-way 5-shot settings, and 10 000 evaluation episodes are used for each dataset to estimate mean accuracy. Training is conducted on four NVIDIA RTX 3090 GPUs.

6.1.3 Evaluation Metrics

The primary evaluation metric is the average top-1 classification accuracy over the query videos in all episodes. Results are reported for 5-way 1-shot and 5-way 5-shot configurations. In addition to in-domain performance, cross-dataset evaluations are conducted by training on the Kinetics subset and testing on HMDB-51, UCF101 and SSv2. Temporal reverse experiments test model sensitivity to frame order by reversing the frame sequence of test videos and measuring the change in accuracy. Finally, t-SNE visualizations provide qualitative insight into the separability of feature distributions.

6.2 Overall Performance

Table 1 in the original study shows that TSAM achieves state-of-the-art performance across all five datasets. When using CLIP-ViT-B as the backbone, TSAM obtains the following 1-shot accuracies: 60.5% on SSv2-Small, 65.8% on SSv2-Full, 84.5% on HMDB-51, 98.3% on UCF101 and 96.2% on Kinetics. In the 5-shot setting, the corresponding accuracies are 66.7%, 74.6%, 88.9%, 99.3% and 97.1%.

Compared with the second-best method MA-CLIP under the same backbone, TSAM improves 1-shot accuracy by 1.4 percentage points on SSv2-Small, 2.5 on SSv2-Full, 1.1 on HMDB-51, 1.3 on UCF101 and 0.5 on Kinetics. For 5-shot tasks, TSAM improves 2.2, 2.3, 1.0, 0.2 and 1.1 percentage points on these datasets respectively. These gains are particularly pronounced on SSv2, which places strong demands on temporal modeling. The improvements indicate that explicitly modeling temporal order and integrating textual corpus and unbalanced optimal transport matching significantly strengthens few-shot action recognition.

6.3 Cross-Dataset Generalization

Cross-dataset experiments train the model on the Kinetics subset and evaluate on UCF101, HMDB-51 and SSv2. Under this setting, TSAM substantially outperforms CLIP-FSAR. Reported improvements are 3.7 percentage points on UCF101, 8.1 on HMDB-51 and approximately 3.9 on SSv2. These results demonstrate that the temporal sequence-aware encoder and multimodal prototype construction produce representations that generalize well across different datasets, not only within the training distribution.

6.4 Ablation Studies

6.4.1 Effect of Adapter Injection Depth

The effect of the number of sequential perceiver adapters is studied by inserting adapters into varying numbers of transformer blocks from the last block backward. On datasets such as UCF101 and Kinetics, which already achieve high accuracy, adding more adapters yields only marginal improvements. On HMDB-51, adding four adapters leads to a noticeable accuracy gain, while adding more provides diminishing returns. This supports the design choice of adapting only the later transformer blocks, which contain more abstract semantic features suitable for temporal augmentation, while keeping parameter growth and computational overhead moderate.

6.4.2 Contribution of Individual Modules

Table 2 in the original study evaluates the contribution of each major component of TSAM on SSv2-Small and HMDB-51 in the 5-way 1-shot setting. The baseline configuration, corresponding to CLIP-FSAR with no sequential perceiver, no textual corpus enhancement and no unbalanced optimal transport matching, achieves 54.6% accuracy on SSv2-Small and 77.1% on HMDB-51.

Adding only the Sequential Perceiver (S.P) increases accuracy to 58.2% on SSv2-Small and 82.2% on HMDB-51. This demonstrates that injecting sequence-aware temporal information into the CLIP backbone alone yields substantial improvements. When both Sequential Perceiver and Textual Corpus enhancement (T.C) are included, accuracy further improves to 59.0% and 83.9% on SSv2-Small and HMDB-51 respectively, indicating that class-level textual semantics refine prototypes in a complementary manner.

Combining Sequential Perceiver with Unbalanced Optimal Transport Matching (UOTM) produces accuracies of 58.7% and 82.6%, which is better than using Sequential Perceiver alone but slightly worse than adding textual corpus. Finally, when all three components S.P, T.C and UOTM are used together, TSAM achieves the highest accuracies of 60.2% on SSv2-Small and 84.5% on HMDB-51. These results validate that each module contributes positively and that the modules are complementary rather than redundant.

6.5 Temporal Reverse Comparison

Temporal reverse experiments test whether the model is genuinely sensitive to frame order. For each dataset, test videos are processed in both original and reversed frame order. An order-sensitive model is expected to exhibit performance degradation when sequences are reversed, especially on temporally demanding datasets.

The CLIP-FSAR baseline exhibits almost no change in accuracy between the original and reversed inputs. In contrast, TSAM shows noticeable performance drops when videos are reversed. On SSv2-Full, TSAM achieves 65.5% accuracy on original videos and 62.3% on reversed videos, a decrease of 3.2 percentage points. On HMDB-51, the accuracy drops from 84.5% to 83.4%, and on Kinetics100 from 96.2% to 95.4%. The reductions, particularly on SSv2-Full, demonstrate that TSAM relies on the correct temporal order to recognize actions, rather than only on static scene cues or unordered set statistics.

6.6 Feature Visualization

t-SNE visualizations compare the feature distributions of CLIP-FSAR and TSAM on HMDB-51 and SSv2. For each dataset, five classes are sampled and their test features are projected into two dimensions. The plots show that TSAM produces more compact intra-class clusters and more separated inter-class clusters than CLIP-FSAR, particularly on SSv2. This suggests that the temporal sequence-aware modeling and multimodal prototype construction lead to a more discriminative feature space.

7 Summary

7.1 Key Contributions

The study introduces a Temporal Sequence-Aware Model for few-shot action recognition that emphasizes explicit modeling of temporal order and multimodal prototype construction. The central contributions can be summarized as follows.

First, the model incorporates a sequential perceiver adapter into a pre-trained CLIP backbone. This adapter treats temporal information as a distinct modality and recurrently updates a temporal query along the frame sequence. The resulting temporal embeddings are injected back into the visual backbone, enabling frame-level representations that are sensitive to frame order.

Second, the model enhances class prototypes using textual corpora generated by large language models. By encoding detailed class descriptions with the CLIP text encoder and combining them with visual prototypes, the method leverages external semantic knowledge to strengthen few-shot learning under limited visual supervision.

Third, the study formulates support-query matching as an unbalanced optimal transport problem. By introducing KL divergence penalties on transport marginals, the UOT matching strategy can downweight background and redundant frames while preserving informative alignments. This yields a more robust distance function for few-shot metric learning.

Extensive experiments on five benchmarks demonstrate that TSAM sets new state-of-the-art performance in both 1-shot and 5-shot settings. Improvements are particularly notable on the temporally challenging SSv2 datasets and in cross-dataset generalization. Temporal reverse experiments further confirm that the model is genuinely sensitive to temporal order.

7.2 Limitations and Future Directions

Despite its strong performance, the approach retains several limitations. First, the encoder processes a relatively small number of frames per video. While eight frames capture coarse temporal dynamics, they may not fully represent long or complex actions, especially in videos with multiple sub-actions or slow temporal evolution. Extending the framework to handle longer sequences more efficiently would be a valuable direction.

Second, the method depends on pre-trained CLIP backbones and the quality of large language models for textual corpus generation. In domains where such pre-training is unavailable or where action classes are highly specialized, the benefits of textual enhancement might diminish or require domain-specific language modeling.

Third, optimal transport and unbalanced optimal transport computations introduce additional computational overhead, particularly as the number of frames increases. Although entropic and KL regularization enable efficient Sinkhorn-style algorithms, further work on approximate or structured transport methods could improve scalability.

Future research can investigate extensions of the sequential perceiver to multimodal video inputs such as depth, optical flow or audio, as well as adaptive frame selection strategies that dynamically allocate more temporal modeling capacity to complex segments. In addition, exploring joint training of textual corpus generation and visual encoding may lead to tighter alignment between language and video representations. Finally, the temporal sequence-aware framework could be applied to other video understanding tasks such as temporal localization, procedural video parsing and video question answering, where frame order plays a critical role.