Abstract: In the domain of video recognition, video transformers have demonstrated remarkable performance, albeit at significant computational cost. This paper introduces TSNet, an innovative approach for dynamically selecting informative tokens from given video samples. The proposed method involves a lightweight prediction module that assigns importance scores to each token in the video. Tokens with top scores are then utilized for self-attention computation. We apply the Gumbel-softmax technique to sample from the output of the prediction module, enabling end-to-end optimization of the prediction module. We aim to extend our method on hierarchical vision transformers rather than single-scale vision transformers. We use a simple linear module to project the pruned tokens, and the projected result is then concatenated with the output of the self-attention network to maintain the same number of tokens while capturing interactions with the selected tokens. Since feedforward networks (FFNs) contribute significant computation, we also propose linear projection for the pruned tokens to accelerate the model, and the existing FFN layer progresses the selected tokens. Finally, in order to ensure that the structure of the output remains unchanged, the two groups of tokens are reassembled based on their spatial positions in the original feature map. The experiments conducted primarily focus on the Kinetics-400 dataset using UniFormer, a hierarchical video transformer backbone that incorporates convolution in its self-attention block. Our model demonstrates comparable results to the original model while reducing computation by over 13%. Notably, by hierarchically pruning 70% of input tokens, our approach significantly decreases 55.5% of the FLOPs, while the decline in accuracy is confined to 2%. Additional testing of wide applicability and adaptability with other transformers such as the Video Swin Transformer was also performed and indicated its progressive potentials in video recognition benchmarks. By implementing our token sparsification framework, video vision transformers can achieve a remarkable balance between enhanced computational speed and a slight reduction in accuracy.

Keywords: action recognition; transformer; efficient; token sparsification

1. Introduction

Exploring action recognition has always held a significant place in the field of video understanding. Past research trends indicate that many researchers have favored using 3D convolutional neural networks (CNNs) or 2D CNNs equipped with complex temporal information modules [1–6]. However, these convolution-based networks have a notable issue: they possess a limited receptive field, implying that they might struggle to capture broader, global information dependencies when processing videos. In contrast, visual transformers have received widespread attention in recent years [7–17], primarily because they have demonstrated superior performance over other networks in a series of benchmark...
tests. However, these transformer-based networks have inherent problems too. They are typically very computationally expensive because the transformer structure necessitates calculating attention relationships between all input tokens, causing the computational cost of the self-attention mechanism to grow quadratically. To address this issue, some of the latest research in the image processing field has begun attempting to simplify the computational process [18–24]. Specifically, the core idea of these methods is to retain only those tokens that genuinely contribute to the final prediction while ignoring other irrelevant tokens. While this approach has been validated in the image domain, applying it to video transformers might pose new challenges. This is because current designs of video transformers, especially those integrated with convolutional modules and shift windows, usually process the input tokens hierarchically. In such designs, discarding tokens arbitrarily could disrupt the network’s internal feature map dimensional structure. Therefore, to successfully implement this simplification strategy in video processing, we need to adopt a more refined approach in selecting the most informative tokens, thereby maintaining the network’s structural integrity. In essence, this means that when removing unnecessary tokens, it is imperative to ensure that the network can continue to function efficiently without compromising its performance.

In this paper, we propose an end-to-end framework that adaptively determines token usage for exploring the sparsity of hierarchical models. In particular, we propose a selection network to dynamically determine which tokens to select for the transformer block. The selection network is designed as a single-layer perceptron and is capable of predicting the importance score for each token. Importantly, this selection network can be seamlessly integrated into any transformer block within the model architecture. Even though the selection network adds to overall computational overhead, the extra cost generated is substantially outweighed by the savings gained by masking uninformative tokens. Traditional binary decision methods, due to their nondifferentiable nature, make end-to-end training highly challenging. To overcome this, we employed the Gumbel-softmax [25] trick, which converts binary decisions into differentiable continuous decisions, enabling end-to-end training. We adopted the mask method instead of directly setting the redundant tokens to zero vectors, as this would affect the calculation in the self-attention block. The redundant tokens are projected by a linear layer to keep the number of the total tokens for downstream processing. The transformer backbone network, together with the selection network and projection module, are jointly optimized with a normal cross-entropy loss for classification and a usage loss that encourages reducing of the overall computational cost. The effectiveness of the proposed method is evaluated on the Kinetics-400 [26] using UniFormer-S [17] and VideoSwin-T [12] as the backbones. The results indicate that a remarkable trade-off is attained in terms of increasing speed while simultaneously diminishing accuracy. Specifically, when applied to the UniFormer backbone, there is a significant 55.5% reduction in GFLOPs when 70% of the input tokens are pruned, while the decrease in accuracy remains within a 2% range. Additionally, allocating more computational resources ensures that performance closely matches that of the original model, while saving over 13% of computational cost. Similarly, when applying pruning to 70% of the input tokens with VideoSwin-T as the backbone, we observe a notable reduction of 54% in GFLOPs, while the accuracy reduction remains within 3%. Again, by allocating more computational resources, the pruned model achieves only negligible performance loss compared to the original model while saving over 17% of the computational cost. Our framework showcases the potential of utilizing temporal and spatial sparsity to accelerate transformer-like models.

2. Related Work

Video vision transformers. The impressive achievements of vision transformers (ViTs) in image processing [7] have served as a catalyst for numerous research endeavors focused on spatiotemporal learning in videos. VTN [10] introduces a method for action recognition by incorporating a temporal attention encoder onto existing 2D spatial networks. This combination leverages self-attention mechanisms on high-level features. In the
pursuit of enhancing efficiency, ViViT [14] explores different variations of factorizing the pure-Transformer model across spatial and temporal dimensions. This primarily entails investigating input embedding methods and self-attention mechanisms for spatial and temporal dimensions. TimeSformer [15] delves into the study of several self-attention schemes and proposes that factorized space–time attention yields the most effective results for video recognition. MViT [13] draws inspiration from the multiscale pyramid structure in 2D CNNs and presents a hierarchical structure that expands the channel capacity while concurrently reducing spatial resolution. Video Swin Transformer [12] applies self-attention in local windows, reducing a significant amount of cost compared to global attention calculations. UniFormer [17] combines 3D convolutions and spatiotemporal self-attention mechanisms to improve efficiency. However, despite the outstanding performance of vision transformers, they still have high computational costs, which escalate rapidly with an increasing number of patches. This challenge serves as a driving force for exploring redundancies within videos, with the aim of achieving efficient video recognition.

Efficient video recognition. Efficient video recognition methods are in high demand due to the computational complexity associated with comprehensive studies in this field. Compact models for video recognition have been developed through the compression of 3D CNNs, as explored in multiple studies [27–29]. Although these approaches yield substantial memory savings, they usually necessitate examining each temporal clip within the input video, which results in no decrease in computational complexity. Recent approaches [30–32] have proposed to enhance video recognition efficiency by selecting the most salient video frames as input. However, it is important to note that these approaches primarily concentrate on accelerating CNN-based video recognition models. In contrast, the SpatioTemporal Token Selection (STTS) framework [33] presents an innovative approach for efficient video recognition that dynamically chooses informative tokens separately in the temporal and spatial dimensions rather than simultaneously. Park et al. [34] utilized the greedy K-center search, in which patches that are farthest apart from each other in terms of spatial distance are iteratively selected. This strategy forces the model to sample patches that are distinct. As a result, this model can learn features with less redundant information. Furthermore, TokenLearner [35] aims to accelerate ViTs by learning a limited quantity of tokens through the aggregation of the entire feature map, weighted by a dynamic attention map dependent on the feature map. TokenLearner can be viewed as an advanced tokenization approach for input samples. However, our research diverges from TokenLearner as it concentrates on the progressive selection of informative tokens.

3. Methods

3.1. Overview

Our proposal introduces an adaptive computation framework that aims to reduce the computational burden of vision transformers while maintaining end-to-end functionality. An overview of our method is shown in Figure 1. Given input frames, we split the input into patches and a linear embedding layer is then applied to project the features of each token to an arbitrary dimension. The subsequent network layers are similar to the standard transformer structure, which consists of multiple transformer blocks containing multihead self-attention (MSA) block and feedforward network (FFN). The difference is that we added a selection module before each MSA block and FFN, selecting some tokens to be processed by the standard transformer of the original network. To maintain a consistent number of tokens for compatibility with the general transformer architecture, we introduced a linear mapping layer to handle the remaining tokens. To emphasize our sparse strategy, the hierarchical structure of the general architecture was omitted. TSNet is designed to dynamically learn policies for activating specific patches in the transformer backbone based on the input, promoting the use of less computation while preserving classification accuracy. This approach enables an end-to-end optimization process of the whole network during training. In addition, our proposed framework keeps the number of tokens unchanged.
Figure 1. The overall diagram of the proposed method. The introduced token selection module is placed between the transformer blocks to effectively choose tokens based on the features generated by the preceding layer. $\gamma_s$ and $\gamma_t$ denote the selection ratio and target ratio, respectively. $\mathcal{L}_{ce}$ denotes the cross-entropy loss. $\mathcal{L}_\gamma$ is devised to reduce the overall computational cost. Best viewed in color.

3.2. Token Sparsification

Given a sequence of input tokens $X \in \mathbb{R}^{L \times D}$, where $L$ is the number of total patches and $D$ is the number of total channels, we adopt a linear layer to generate an importance score for each token based on input. Formally, given the input to $l$-th block $Z_l$, the selection policy matrix for this block is computed as follows:

$$ \text{Score} = Z_l W_l $$

where $W_l$ denotes the projection weight. The $\text{Score}$ is subsequently passed to a sigmoid function, yielding the probability of keeping tokens. The ensuing decisions, exhibiting a binary nature, offer a straightforward mechanism to determine whether to activate or inactivate tokens by merely establishing an inferential threshold. However, this process lacks differentiability during training, thereby hindering the optimization potential of the decision network. To this end, we leverage the Gumbel-softmax [25] technique to draw samples from the output of the $\text{Score}$ network. Formally, the decision of $\text{Score}$ is derived as

$$ M_k = \frac{\exp(\log(\text{Score}_k + G_k)/\tau)}{\sum_{j=1}^{K} \exp(\log(\text{Score}_j + G_j)/\tau)} $$

for $k = 1, 2, \ldots, K$

where $K$ is the total number of decision categories. In our case of binary decision, $K = 2$. $G_k$ is the Gumbel distribution with the formula $-\log(-\log(U_k))$, where $U_k$ is a random variable sampled from the uniform distribution. The parameter $\tau$ represents a smoothing
constant. However, in the process of token elimination, the original size of the feature maps can be disrupted. This disruption occurs due to the alteration of the feature space topology caused by the removal of specific tokens. Moreover, the attempt to simply zero out the tokens that are intended to be dropped is found to be unproductive. Specifically, the conventional single-head self-attention is computed as follows:

\[ R = QK^T / \sqrt{d} \]  

\[ \text{Attn}_{i,j} = \text{softmax}(R)V = \frac{\exp(R_{i,j})}{\sum_{k=1}^{L} \exp(R_{i,k})} V \quad 1 \leq i, j \leq L \]  

where \( Q, K, \) and \( V \) are query, key, and value matrices, respectively, and \( d \) is a scaling factor. Obviously, the attempt to simply zero out the tokens that are intended to be dropped is found to be unproductive. Notwithstanding their nominal absence, these zeroed tokens can still influence the remaining tokens because of their contribution to softmax operation, which basically normalizes scores into a probability distribution. This is due to the overall sum of the exponentiated elements in the softmax denominator including these zero elements, thereby affecting the ultimate distribution. To resolve this issue, a strategy involving the use of a mask is implemented. By applying a mask, the interactions between the pruned tokens and the remaining tokens can be effectively reduced. This mask essentially acts as a filter that shields certain tokens from being operated upon, thereby preserving the integrity and independence of the tokens that are not to be pruned. It provides a way to deal with the impracticality of merely zeroing out tokens and thus creates a more effective method for the pruning process. Formally, we compute the output of self-attention by

\[ \text{Attn}_{i,j} = \frac{\exp(R_{i,j})M_j}{\sum_{k=1}^{L} \exp(R_{i,k})M_k} V \quad 1 \leq i, j \leq L \]  

The output of masked attention is computed by concentrating on the active tokens during training. For pruned tokens, we adopt linear projection to keep the token number unchanged during inference. Finally, the output of the MSA block is the sum of the output of masked attention and linear projection with decision \( M_j \):

\[ \text{output} = \text{Attn}_{i,j}M_j + WX(1 - M_j) \]  

where \( W \) denotes the projection weight. During the inference, the input tokens are divided into two distinct groups. This division is based on the probabilities produced by the Score network. After the division, different processing techniques are applied to each group. The first group is treated with self-attention. The other group is subjected to linear projection. After the respective treatments, there is a need to combine the two groups back to maintain the structure of the output. For this, the features from both the groups are reassembled in a way that aligns with the original spatial positions of all the tokens. Formally, the proposed method is as follows:

\[ x \xrightarrow{\text{Split}_{M}} x_1, x_2 \]

\[ x_1 \xrightarrow{\text{MSA}} x_1, \quad x_2 \xrightarrow{\text{Linear}} x_2 \]

\[ x_1, x_2 \xrightarrow{\text{Unify}} x \]  

where \( M \) represents the probabilities generated by the Score network. \( x_1 \) and \( x_2 \) are two groups of tokens, where \( x_1 \) undergoes \( \text{MSA} \) and \( x_2 \) undergoes linear projection. Subsequently, the outputs from these different processes are unified in a manner that aligns with the original spatial positions of all the tokens. We also propose to introduce asymmetric
processing to the feedforward network (FFN) while keeping output structures unchanged. Formally, the proposed method is as follows:

$$x \xrightarrow{\text{Split}} M \xrightarrow{\text{FFN}} x_1, x_2 \xrightarrow{\text{Linear}} x_1, x_2 \xrightarrow{\text{Unify}} x$$

This is similar to the process described in Equation (7), with the difference that a group of the tokens undergoes FFN processing while the rest undergo linear projection. Then, the outputs of these different processes are merged together in a way that preserves the original spatial positions of all tokens.

3.3. Objective Function

Given an input video $I$ with the ground truth $y$, the final prediction is produced by the model $F$ with parameters $\theta$. We adopt the standard cross-entropy loss:

$$L_{ce} = -y \log(F(I; \theta))$$

To encourage reducing the overall computational cost, we set the ratio of the kept tokens to a predefined value. If no value is set, the training of the network tends to prioritize higher accuracy rather than discarding tokens. In extreme cases, it may hardly discard any tokens to maintain performance consistency with the original model. Given a target ratio $\gamma$, we employ an MSE loss as follows:

$$L_{\gamma} = \left( \frac{1}{N} \sum_{i=1}^{N} M_i - \gamma \right)^2$$

where $N$ denotes the total numbers of patches of the entire transformer. The hyperparameter $\gamma \in (0, 1]$ indicates target ratio of tokens to keep. The final loss function is a combination of the two objectives:

$$L = L_{ce} + \lambda L_{\gamma}$$

where we set $\lambda = 10$ in our experiments.

4. Experiments

4.1. Dataset and Backbone

We conduct experiments on action recognition datasets Kinetics-400 [26]. Kinetics-400 encompasses an extensive collection of 400 distinct human action classes, accompanied by a substantial collection of training videos totaling approximately 240 k as well as 20 k validation videos. Our method is trained on the training set and evaluated on the validation set. Consistent with prior studies, we adhere to the convention of presenting the top-1 and top-5 recognition accuracy as performance metrics.

The primary model used in this study is the video transformer UniFormer. The choice of this model is attributed to its superior performance coupled with a reasonable computational cost. Notably, UniFormer’s structure is characteristic of a hierarchical vision transformer, which includes a convolution in its self-attention block. This effectively validates our method’s efficacy, as it operates without compromising the original structure of the transformer. UniFormer [17] is intricately crafted, combining both local and global multihead relation attention (MHRA) blocks. Convolutions replacing self-attention at the earlier stages of this design do not allow for the implementation of naive token pruning. Consequently, token sparsification and asymmetric processing are co-applied in the deeper layers while in the shallower layers, only asymmetric processing is utilized in the feedforward network (FFN). To further affirm the versatility and adaptability of our approach
across varying transformer architectures, we carry out additional experiments with Video Swin Transformer [12] as well.

4.2. Implementation Details

In our experiments, we refine pretrained video transformers using selection networks, which have randomly initialized parameters. We sample clips of 16 frames with a temporal stride of 4 following UniFormer-S. The spatial size of each frame is cropped to 224 × 224 pixels. The network is trained using the AdamW optimizer [36], under a cosine learning rate schedule [37], for a total of 20 epochs. The first three epochs are allocated for linear warmup [38]. The initial learning rate is set to 1 × 10^{-6} for the backbone network, and 1 × 10^{-4} for the selection network. The learning rate for the backbone layers is determined to be 0.01 times that of the selection network’s learning rate, with a mini-batch consisting of eight video clips. With regard to the implementation of the Video Swin Transformer, a clip of 32 frames is sampled with a temporal stride of 2 and a spatial size of 224 × 224. Like the UniFormer-S, AdamW is also used as an optimizer for a total of 20 epochs, with the initial 3 epochs being used for linear warmup. The learning rate for the selection networks is set to 1 × 10^{-4}, while the backbone model employs a learning rate of 0.01 times that. The mini-batch size is reduced to four video clips for this network. The networks are implemented in PyTorch and the experiments are carried out on four Tesla V100 GPUs that each have a memory of 32 GB. During inference, we adopt the same testing strategies employed in the original backbones to ensure a fair comparison. To gauge the performance of classification conducted by the model, metrics such as top-1 and top-5 accuracy are reported on the validation set. Measuring the computational costs is performed through the use of FLOPs, or floating-point operations, a standard metric used in data science to assess the computational demands of an algorithm regardless of the specific hardware being used. To achieve a suitable balance between computation and accuracy, a multiclip testing approach is adopted when testing for Kinetics-400.

4.3. Main Results

Effectiveness of TSNet. We compare TSNet with the original UniFormer-S (baseline) and the random selection method, which randomly samples the same number of tokens using the selection policy produced by our TSNet. The comparison focuses on different selection ratios set as 0.8, 0.5, and 0.3. We denote the variants of methods by method/r, where r refers to the selection ratio. From the obtained results, as shown in Table 1, it is apparent that even when a small number of tokens are removed, the classification accuracy does not fall notably, implying that some tokens can be entirely redundant, contributing minimal value to the final classification. These results accentuate the importance of strategically lessening computational burdens and improving processing speed by selecting the most informative tokens to process. Our TSNet framework perform competitively, delivering similar results but with a lesser computational requirement — reducing computation by approximately 13.4%. As the selection rate of tokens decreases, the computational workload will significantly decrease while only incurring marginal accuracy loss. Even when 70% of input tokens are pruned, there is a significant reduction in computation by 55.5% in terms of GFLOPs, with a manageable accuracy loss of only 1.7%. When the token selection rate is 0.8, the random method only experiences a 1.5% decrease in accuracy. This, once again, demonstrates that some tokens in the transformer structure are redundant, as a portion of tokens can be completely replaced by another set of tokens while maintaining a certain number of tokens. TSNet’s notable superiority over random selection methods is evident when operating under comparable computational budgets. For instance, TSNet/0.5 significantly outperforms Random/0.5 by 49.8%, while utilizing approximately the same computation measured at 99.2 GFLOPs. Further, when a substantial number of tokens are discarded, as in the case of Random/0.3, the accuracy result is extremely low. This supports the hypothesis that our dynamic token selection module more effectively preserves informative tokens as opposed to a random approach. Importantly, the incremental computational
overhead added by token selection module is marginal. In fact, the parameters and FLOPs of the scorer network are a minute proportion (0.038% and 0.022%, respectively) compared to those incorporated in the original UniFormer-S backbone. This speaks to the significant computational efficiencies that our TSNet model presents.

Table 1. Comparisons with baseline and random selection method on Kinetics-400.

<table>
<thead>
<tr>
<th>Method</th>
<th>GFLOPs</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>167</td>
<td>80.8</td>
<td>94.7</td>
</tr>
<tr>
<td>Random/0.8</td>
<td>144.4</td>
<td>79.3</td>
<td>93.7</td>
</tr>
<tr>
<td>Random/0.5</td>
<td>99.2</td>
<td>29.7</td>
<td>44.8</td>
</tr>
<tr>
<td>Random/0.3</td>
<td>74</td>
<td>1.1</td>
<td>3.5</td>
</tr>
<tr>
<td>TSNet/0.8</td>
<td>144.4</td>
<td>80.5</td>
<td>94.3</td>
</tr>
<tr>
<td>TSNet/0.5</td>
<td>99.2</td>
<td>79.5</td>
<td>93.8</td>
</tr>
<tr>
<td>TSNet/0.3</td>
<td>74</td>
<td>79.1</td>
<td>93.7</td>
</tr>
</tbody>
</table>

Comparison with state of the art. Table 2 provides a comprehensive comparison of TSNet with state-of-the-art video recognition models on the Kinetics-400 dataset. The comparison includes both CNN-based and transformer-based models. In order to showcase the generalizability of our approach across different transformer architectures, we use both UniFormer-S [17] and VideoSwin-T [12] as the base models. For clear comparison, we separate the models into three groups. Our TSNet demonstrates favorable trade-offs between complexity and accuracy. X3D takes the modification and decomposition of 3D models to the extreme and is a comprehensive 3D CNN model. However, while X3D-M has similar computational complexity to UniFormer-S, its performance differs by 4.8%. By implementing our method to further reduce the computational workload of UniFormer-S, we achieve better accuracy than X3D with less than half the computational requirements. Notably, our TSNet(UniFormer-S)/0.5 achieves results comparable to TimeSformer, but requires only 2% of the computational cost. TSNet(UniFormer-S)/0.3 outperforms MViT-B16 [13] while saving over 80% in computation. It is important to highlight that for backbone models UniFormer-S and VideoSwin-T, our TSNet is capable of saving 13~55% of the computational cost with less than a 3% drop in accuracy. Compared to UniFormer-S, it appears that applying our method on VideoSwin-T might result in a more significant performance decline. This is because UniFormer-S utilizes a shallow network with convolutional layers that do not implement token selection, preserving the complete features in the shallow layers. In contrast, the redundancy in deep feature maps is more pronounced since crucial features are concentrated in a small portion of the regions.

Table 2. Comparison with the state-of-the-art models on Kinetics-400.

<table>
<thead>
<tr>
<th>Method</th>
<th>#Frames</th>
<th>GFLOPs</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X3D-M [28]</td>
<td>16 × 3 × 10</td>
<td>186</td>
<td>76.0</td>
<td>92.3</td>
</tr>
<tr>
<td>X3D-XL [28]</td>
<td>16 × 3 × 10</td>
<td>1452</td>
<td>79.1</td>
<td>93.9</td>
</tr>
<tr>
<td>VTN [10]</td>
<td>250 × 1 × 1</td>
<td>3992</td>
<td>78.6</td>
<td>93.7</td>
</tr>
<tr>
<td>TimeSformer [15]</td>
<td>96 × 3 × 1</td>
<td>5110</td>
<td>79.7</td>
<td>94.7</td>
</tr>
<tr>
<td>MViT-B16 [13]</td>
<td>16 × 1 × 5</td>
<td>353</td>
<td>78.4</td>
<td>93.5</td>
</tr>
<tr>
<td>ViViT-L [14]</td>
<td>16 × 3 × 4</td>
<td>17352</td>
<td>80.6</td>
<td>94.7</td>
</tr>
<tr>
<td>VideoSwin-T [12]</td>
<td>32 × 3 × 4</td>
<td>1056</td>
<td>78.8</td>
<td>93.6</td>
</tr>
<tr>
<td>TSNet(VideoSwin-T)/0.8</td>
<td>32 × 3 × 4</td>
<td>871.9</td>
<td>78.3</td>
<td>92.9</td>
</tr>
<tr>
<td>TSNet(VideoSwin-T)/0.5</td>
<td>32 × 3 × 4</td>
<td>628.1</td>
<td>77.0</td>
<td>91.4</td>
</tr>
<tr>
<td>TSNet(VideoSwin-T)/0.3</td>
<td>32 × 3 × 4</td>
<td>484.8</td>
<td>76.2</td>
<td>90.7</td>
</tr>
<tr>
<td>UniFormer-S [17]</td>
<td>16 × 1 × 4</td>
<td>167</td>
<td>80.8</td>
<td>94.7</td>
</tr>
<tr>
<td>TSNet(UniFormer-S)/0.8</td>
<td>16 × 1 × 4</td>
<td>144.4</td>
<td>80.5</td>
<td>94.3</td>
</tr>
<tr>
<td>TSNet(UniFormer-S)/0.5</td>
<td>16 × 1 × 4</td>
<td>99.2</td>
<td>79.5</td>
<td>93.8</td>
</tr>
<tr>
<td>TSNet(UniFormer-S)/0.3</td>
<td>16 × 1 × 4</td>
<td>74</td>
<td>79.1</td>
<td>93.7</td>
</tr>
</tbody>
</table>
4.4. Ablation Studies

We conduct ablation studies with UniFormer as backbone on Kinetics-400 to assess the contribution of our proposed TSNet.

**Effectiveness of sparsification in MSA and FFN.** To evaluate the effectiveness of dynamic sparsification in both MSA and FFN, we conducted ablation studies using a selection rate of 0.5, with results demonstrated in Table 3. Our methodology is contrasted with a variety of token selection strategies, including (1) MSA_only, a strategy that exclusively samples tokens within the MSA block; (2) FFN_only, a strategy that restricts token sampling within the FFN; and (3) MSA/FFN, a methodology that equally samples tokens within both the MSA block and FFN using an identical selection policy. The resulting data suggest that dynamic sparsification proves beneficial within both MSA and FFN. Further, our proposed TSNet, which samples tokens within both the MSA block and FFN using different selection policies, provides an optimum balance between complexity and accuracy. Notably, TSNet leads MSA/FFN, implying the significance of utilizing diverse selection policies for the dynamic selection of tokens.

<table>
<thead>
<tr>
<th>Method</th>
<th>GFLOPs</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniFormer-S</td>
<td>167</td>
<td>80.8</td>
<td>94.7</td>
</tr>
<tr>
<td>MSA_only/0.5</td>
<td>121.4 (−27.3%)</td>
<td>79.8</td>
<td>93.3</td>
</tr>
<tr>
<td>FFN_only/0.5</td>
<td>140.5 (−15.8%)</td>
<td>80.2</td>
<td>94.1</td>
</tr>
<tr>
<td>MSA/FFN/0.5</td>
<td>98.4 (−41.0%)</td>
<td>79.3</td>
<td>93.6</td>
</tr>
<tr>
<td>TSNet(UniFormer-S)/0.5</td>
<td>99.2 (−40%)</td>
<td>79.5</td>
<td>93.8</td>
</tr>
</tbody>
</table>

**Redundancy vs. depth.** Our tests on the redundancy of patches in different layers or blocks indicates that the deeper the block is, the more patches that can be removed without significantly impacting performance. In our experimentation, we only sparse the tokens in the last two layers, which, respectively, consist of eight and three blocks. For the sake of simplicity, blocks 1, 5, 9, and 11 of the total 11 blocks are created as sparsifications. The same selection ratio is maintained when individual blocks are trimmed, with a scale setting chosen as 0.1. The data presented in Table 4 provide insights into the accuracy of models that have undergone pruning based on patch selection within specific blocks. The results indicate that removing patches from lower blocks tends to result in a substantial decrease in accuracy. This suggests that the patches in lower blocks contribute significantly to the overall performance of the models. Interestingly, when examining the sparsity within blocks belonging to the same layer (such as blocks 1 and 5 and blocks 9 and 11 in this particular case), the impact on accuracy was found to be relatively minor. This implies that the redundancy of patches within the same layer is relatively similar. On the other hand, redundancy of patches varies significantly among different layers, with deeper layers demonstrating higher redundancy. This can be attributed to the attention mechanism, which aggregates the characteristics of different patches, leading to a full communication between deeper patches. This essentially means that features of patches in deeper layers are more redundant since they have captured and shared more information through the attention mechanism. Therefore, these patches can be pruned without losing critical information, which justifies the better performance of our pruning strategy in deeper layers.

<table>
<thead>
<tr>
<th>Depth</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>79.4</td>
<td>93.3</td>
</tr>
<tr>
<td>5</td>
<td>79.5</td>
<td>93.5</td>
</tr>
<tr>
<td>9</td>
<td>80.5</td>
<td>94.5</td>
</tr>
<tr>
<td>11</td>
<td>80.7</td>
<td>94.6</td>
</tr>
</tbody>
</table>
Different location for token selection. The flexibility of TSNet allows us to reduce the similar computational cost by selecting different token selection configurations. For example, to reduce the computation of UniFormer-S by approximately 50%, we can apply one of the following strategies: (1) Apply token selection only to MSA or FFN in the early stages. (2) Apply token selection only to MSA or FFN in the later stages. (3) Apply a joint token selection strategies. In this section, we present a comprehensive analysis of these options, using UniFormer-S as a case study and evaluating its performance on the Kinetics-400 dataset. We represent different variants by $\mathbb{L}^{R_1 \mathbb{H}}^{R_2}$, where $L$ and $H$ represent the token selection performed in the third and fourth layers, respectively. $R_1$ and $R_2$ denote the selection ratios in MSA block and FFN, respectively. Table 5 shows the performance results of TSNet when token selection is applied at various locations in the architecture. In every setting, the selected token ratios are managed to ensure that the computational cost drops by around 20%. We observe that selection in deeper layers exceeds that in shallower layers, which demonstrates that features of patches in deeper layers are more redundant, and confirms the conclusion drawn in Table 4. Moreover, under the same calculation cost, the effectiveness of dynamic sparsification in MSA exceeds that in FFN. This is a supplement to Table 3. In addition, joint token selection achieves the best result.

Table 5. Ablation studies to assess different location for token selection.

<table>
<thead>
<tr>
<th>Method</th>
<th>GFLOPs</th>
<th>Top-1 (%)</th>
<th>Top-5 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{L}^{0.6}_{1} \mathbb{H}^{1}$</td>
<td>134.8 (−19.3%)</td>
<td>79.0</td>
<td>92.7</td>
</tr>
<tr>
<td>$\mathbb{L}^{0.8}_{1} \mathbb{H}^{1}$</td>
<td>134.3 (−19.6%)</td>
<td>79.5</td>
<td>93.1</td>
</tr>
<tr>
<td>$\mathbb{L}^{0.6}_{1} \mathbb{H}^{0.8}$</td>
<td>132.4 (−20.7%)</td>
<td>79.3</td>
<td>93.5</td>
</tr>
<tr>
<td>$\mathbb{L}^{0.8}_{1} \mathbb{H}^{1}$</td>
<td>134.1 (−19.7%)</td>
<td>79.6</td>
<td>93.5</td>
</tr>
<tr>
<td>$\mathbb{L}^{0.8}_{0.8} \mathbb{H}^{0.5}$</td>
<td>134.9 (−19.2%)</td>
<td>80.0</td>
<td>93.9</td>
</tr>
</tbody>
</table>

5. Conclusions

In this paper, we propose an innovative approach to expedite vision transformers by harnessing the sparsity of informative patches present in the input video frames. Our strategy, encapsulated within our TSNet framework, dynamically prunes less significant tokens for each input instance, guided by a personalized binary decision mask generated from a lightweight selection module. This selection module is introduced at multiple layers, allowing token pruning to occur hierarchically. Gumbel-softmax is employed in the selection module to achieve comprehensive end-to-end training. Attention masking techniques can mask redundant tokens during the training process to prevent interference with other tokens. A linear projection is utilized for pruned tokens in order to preserve the unchanged token quantity. During the inference process, our methodology effectively enhances efficiency by pruning the input tokens. Our framework proves to be versatile and applicable to a broad array of network structures, including, but not limited to, transformers equipped with convolutions or shifted windows. Although the key focus of this paper is on the task of action recognition, extending our methodology to other applications such as image classification and dense prediction tasks presents interesting prospects. In summary, our TSNet offers a significant contribution to transformer models’ optimization, highlighting the importance of dynamic selection and pruning within this context.

Author Contributions: Conceptualization, H.W.; methodology, H.W.; software, H.W.; validation, H.W. and W.Z.; formal analysis, H.W.; data curation, H.W. and W.Z.; writing—original draft preparation, H.W.; writing—review and editing, G.L.; visualization, H.W. and W.Z.; supervision, G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.
References


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.