electronics

MDPI

Article

Toward Unified and Quantitative Cinematic Shot Attribute Analysis

Yuzhi Li, Feng Tian *, Haojun Xu and Tianfeng Lu

Shanghai Film Academy, Shanghai University, Shanghai 200073, China; shadowmcv@shu.edu.cn (Y.L.); kyoani@shu.edu.cn (T.L.)
* Correspondence: ouman@shu.edu.cn

Abstract: Cinematic Shot Attribute Analysis aims to analyze the intrinsic attributes of movie shots, such as *movement* and *scale*. In previous methods, specialized architectures were designed for each specific task and relied on the use of optical flow maps. In this paper, we consider shot attribute analysis as a unified task of motion–static weight allocation, and propose a motion–static dual-path architecture for recognizing various shot attributes. In this architecture, we design a new action cue generation module for adapting the end-to-end training process instead of a pre-trained optical flow network; and, to address the issue of limited samples in movie shot datasets, we design a fixed-size adjustment strategy to enable the network to directly utilize pre-trained vision transformer models while adapting to shot data inputs at arbitrary sample rates. In addition, we quantitatively analyze the sensitivity of different shot attributes to motion and static features for the first time. Subsequent experimental results on two datasets, MovieShots and AVE, demonstrate that our proposed method outperforms all previous approaches without increasing computational cost.

Keywords: shot attribute analysis; shot type classification; unified model; end-to-end architecture; deep learning

Citation: Li, Y.; Tian, F.; Xu, H.; Lu, T. Toward Unified and Quantitative Cinematic Shot Attribute Analysis. *Electronics* **2023**, *12*, 4174. https://doi.org/10.3390/electronics12194174

Academic Editors: Haibin Wu, Aili Wang and Yuji Iwahori

Received: 11 September 2023
Revised: 4 October 2023
Accepted: 7 October 2023
Published: 8 October 2023

1. Introduction

Frames, shots, and scenes are different entities or units that constitute a movie. A frame is a still image, the basic unit of a movie; a shot consists of a series of frames displaying related action or plot; and a scene is a collection of consecutive shots at the same time or place used to show a coherent movie story line.

Generally, in movie analysis [1], shot segmentation algorithms [2,3] are used to divide movies into thousands of distinct shots to facilitate understanding at the shot level. Unlike action recognition [4–6], shot attribute analysis focuses more on common attributes of all shots, such as *scale*, *movement*, and *angle*, which we refer to as the **intrinsic attributes** of movie shots.

By extracting and comprehending intrinsic shot attributes, we can semantically index and search movie archives based on cinematographic characteristics. This also facilitates high-level analysis of film styles by identifying patterns in the use of *scale*, *movement*, and *angle* throughout a film. Furthermore, automatic shot attribute analysis can potentially enable AI-assisted editing and intelligent cinematography tools that provide suggestions based on learned film shot patterns and conventions.

In previous shot attribute analysis methods, as shown in Figure 1, each shot attribute is often treated as an independent classification task, such as *shot movement classification* [7] and *shot scale classification* [8]. These methods utilize task-specific network architectures tailored for predicting each property. Consequently, they are applicable to single-property prediction and cannot be easily generalized to other properties.

Moreover, prior research [9] has shown that action cues are critical features of shot attribute analysis; thus, most methods [1,7,10,11] employ pre-trained optical flow networks

to extract video optical flow map. However, since the training datasets of optical flow networks [12–15] are mostly generated in virtual environments, distortions may occur when extracting optical flow from real movie shots, and the end-to-end training process cannot be achieved using additional optical flow networks.

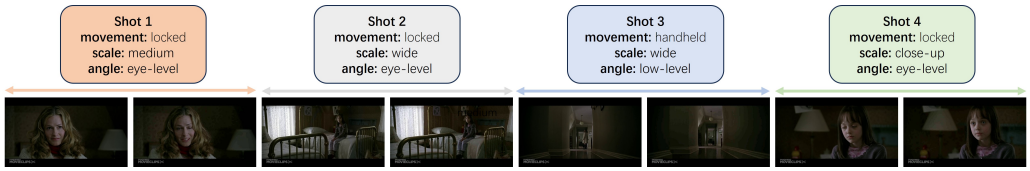

Figure 1. Demonstration of the movie shot analysis task. After splitting the movie into shots, several intrinsic attributes of the shots need to be accurately analyzed.

The above analysis identifies some of the problems in the shot attribute analysis task and points to our main research motivation: how to design a unified architecture for multiple shot attribute analysis, and how to capture the motion cues of a shot without relying on optical flow networks while achieving an end-to-end training process.

Inspired by [6,16], we introduce a learnable frame difference generator to replace the pre-trained optical flow network, i.e., using frame difference maps as motion cues, and successfully enable the architecture for end-to-end training. However, certain shot attributes like shot scale are less sensitive to motion cues; thus, overly relying on motion cues when analyzing these attributes may affect the performance [1]. Therefore, inspired by the dual-stream networks, we balance motion and static features and design a motion and static branch to independently analyze motion cues and static frames, and further quantify the weights of the motion and static feature in the fusion module. Then, given the effectiveness of transformers in computer vision tasks [17,18], we attempt to replace traditional convolutional neural networks with them. However, since transformers require a large amount of training data, and the sample size of movie shot datasets [19–22] is usually less than 30 K, we introduce transformer models pre-trained by Kinetics 400 [23], and design a fixed-size adjustment strategy for the motion branch and a keyframe selection strategy for the static branch to adapt to fit the input size; meanwhile, this design also allows our architecture to accommodate shot data inputs at any video sample rate.

We validate our proposed method on two large movie shot datasets, MovieShots [1] and AVE [24], and compare the performance with all previous methods. Given the sample imbalance problems of movie shot datasets, we employ Macro-F1 for evaluation in addition to the regular Top-1 accuracy. The results show that our model performs significantly better than other methods while maintaining computational efficiency.

Our contributions are as follows:

1. We summarize the main issues in the task of shot attribute analysis. Building upon these, we propose a unified dual-branch end-to-end architecture capable of analyzing all movie shot attributes, and further quantify the motion/static feature weights of different attributes.

2. We design a learnable frame difference generator to replace the pre-trained optical flow network, and through specific strategies make the network compatible with vision transformer pre-trained models, effectively solving the problem of lacking samples in the movie shot dataset.

3. Experiments on MovieShots and AVE prove that our shot attribute analysis architecture significantly outperforms all previous methods in various shot intrinsic attributes, and exhibits notable advantages in computational efficiency.

2. Related Work

2.1. Two-Stream Architecture

A dual-stream network [9,25–27] consists of two parallel networks: one processes spatial information (spatial stream), while the other handles temporal information (temporal

stream); this architecture performs well in processing video data and significantly improves the accuracy of action recognition. Here, we refer to them as the motion branch and static branch. In the traditional dual-stream network architecture, the motion branch uses optical flow maps as input to capture dynamic information in videos, such as the movement path and speeds of objects. The static branch uses video frame inputs to understand the low-level texture features and high-level semantic information in the video.

In our method, we optimize the two-stream architecture for shot attribute analysis: (1) We use the frame difference generator instead of the optical flow network to extract the dynamic information. (2) We choose the visual transformer architecture in lieu of the traditional convolutional backbone. (3) We select one key frame as input due to the characteristics of the shot analysis task, which significantly reduces the amount of computation. (4) In the resulting fusion module, we add an adaptive parameter to balance the contribution weights of the dual branches in different shot attributes instead of concatenating the two features directly.

2.2. Shot Arrtibute Analysis

The goal of the shot attribute analysis task is to analyze the intrinsic attributes of movie shots. Traditional methods [10,11,19–22,28,29] rely on hand-crafted low-level features (e.g., dominant color regions, camera motion histogram descriptors) as well as traditional machine learning methods (e.g., support vector machines, decision trees), which are constrained by the priori hypothesis of the selected features, leading to analysis of the results with low accuracy and a lack of generalization ability.

In following CNN-based approach [1,7,8,30–33], the baseline SGNet [1], for instance, employs video frames and optical flow maps as inputs to generate subject maps, which segment the foreground and background of the video. The foreground and background are then analyzed using a dual-stream network structure to examine *movement* and *scale* within a unified architecture.

Subsequent approaches like MUL-MOVE-Net [7] analyze shot *movement* using 1D angular histograms generated from optical flow maps. Bias CNN [8] introduces vertical and horizontal pooling methods to analyze shot *scale*. SCTSNet [30] combines the recognition of *movement*, *scale*, and *angle* to assist in shot boundary detection.

The CNN-based methods mentioned above achieve much higher accuracy in shot attribute analysis tasks compared to traditional machine learning methods. However, due to the local assumption inherent in the design of small convolutional kernels, these methods might not capture global features adequately. Furthermore, film shot attributes encompass both strong temporal properties (e.g., *movement*) and weak temporal properties (e.g., *scale*, *angle*). Vision transformers [17], owing to their absence of inductive bias, can automatically learn global features across the temporal and spatial dimensions of shot clips during training. This property potentially makes them more suitable for shot attribute analysis tasks.

2.3. Movie Shot Dataset

Due to copyright restrictions, most movie shot datasets publicly released, and the limited number of shot samples [19–21,33], pose a challenge to the advancement of shot attribute analysis research. MovieShots [1] is the first large dataset to provide multi-category (*movement* and *scale*) shot attribute annotations along with complete video data. It has been utilized several times in subsequent studies. In this paper, alongside MovieShots, we additionally incorporate another larger movie shot dataset, AVE [24], and select four shot attributes - *shot motion*, *shot scale*, *shot angle*, and *shot type*, for analysis.

Subsequently, we conduct extensive experiments using these two datasets. To our knowledge, AVE has been employed for the first time in shot attribute analysis tasks. We believe that compare to previous studies that use non-open datasets or solely rely on MovieShots, our experiments conducted on two large datasets can demonstrate the effectiveness of our architecture.

3. Approach

In this section, we present a unified end-to-end training architecture suitable for various shot attribute tasks. Despite prior attempts to employ a unified architecture for analyzing multiple shot attributes, adjustments have been made in the structural design to cater to distinct classification tasks (e.g., Var Block in SGNet [1]). In our architecture, as shown in Figure 2a, we conceptualize the analysis tasks as a balance between motion and static aspects, leading to the design of corresponding feature extraction backbones.

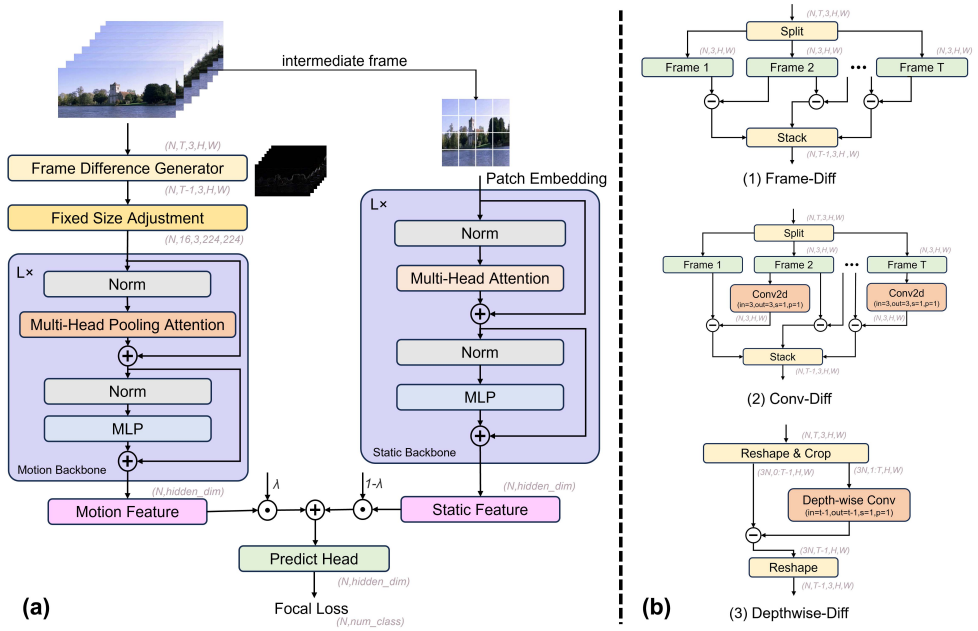

Figure 2. (**a**) Overview of our movie shot analysis architecture. The motion branch accepts shot data inputs and employs the frame difference generator to generate difference maps. These maps are subsequently passed through the fixed size adjustment to resize the features to the dimensions $(16, 224, 224)$, then outputs the motion feature by passing motion backbone. The static branch selects the intermediate frame of the shot clip and extracts static feature using static backbone. (**b**) We propose three frame difference generation methods, wherein the input and output sizes remain unaltered except for a reduction in the temporal dimension by one.

In the following four subsections, we first define the lens attribute analysis task with a formula specification in Section 3.1. Subsequently, we elaborate on the design of the motion and static branches in Sections 3.2 and 3.3. In Section 3.4, a quantifiable strategy for fusing motion and static features is introduced.

3.1. Problem Definition

Consider a given shot $S = \{f_1, f_2 ... f_n\}$ consists of a sequence of frames, where $f_i \in R^{3\times H\times W}$represents the ith frame of the shot, and n denotes the number of frames in the shot. Each frame can be regarded as a 2d image that encompasses the static visual information of the shot. The objective of shot attribute analysis is to identify a set of functions $\Theta = \{\theta_1, \theta_2 ... \theta_n\}$, which can map movie shot S to a pre-defined set of movie shot types $C = \{C_1, C_2 ... C_m\}$. Here, c_j denotes the jth movie shot attribute, while m signifies the total number of the attributes (for instance, in MovieShots, $C = \{movement, scale\}$). The entire process can be expressed by the following equation: $\{C_1, C_2 ... C_m\} = \{\theta_1(S), \theta_2(S) ... \theta_n(S)\}$, or alternatively as $C = \Theta(S)$.

3.2. Motion Branch

3.2.1. Drawbacks of Optical Flow

When analyzing motion clues in movie shots, previous shot attribute analysis methods [1,7,10,11] often employ optical flow maps as the motion cue, which show the differences between successive frames through pixel-level displacement vectors. However, the extraction of optical flow maps requires the introduction of additional optical flow pre-trained networks [13] and consumes a substantial amount of time for optical flow calculation, making it unsuitable for the end-to-end architecture. Moreover, we point out that since the datasets [12,14] used to train the optical flow network are generated in virtual environments, it might not fully and accurately simulate various complex scenes and object movements in the real world, especially in dynamic and variable movie shots. If there is any distortion in the obtained optical flow maps it may affect the accuracy of the analysis results.

3.2.2. Frame Difference Generator

In the motion branch, we build a unique frame difference generator module. This module accurately represents the dynamic variations between consecutive frames by generating frame difference maps. As shown in Figure 2b , we propose three methods for this module: (1) **Frame-Diff**: Firstly, we split the input clip into separate frames, subtract the latter frame from the former frame sequentially, and stitch the result directly. This method is advantageous as it directly showcases the differences between adjacent frames. (2) **Conv-Diff**: This function also entails splitting the input into separate parts, but the subsequent frame is first passed through a single convolutional layer (kernel size = 3, stride = 1, padding = 1) before being subtracted from the previous frame. This process allows the resulting features to represent motion changes while considering local feature information. (3) **Depthwise-Diff**: In this method, we first truncate the input data for transformation and rearrangement, combine the channel dimension with the batch dimension, and then extract frames 0 to T $-$ 1 and frames 1 to T for the preceding and subsequent frames, respectively. The latter is processed through a depthwise separable convolutional [34] layer and then subtracted from the former, followed by one deformation layer for output transformation.

Among these three methods, we particularly emphasize the Depthwise-Diff method, and subsequent experimental results in Section 4 have demonstrated the efficiency of this method. Firstly, in terms of implementation, this method is equivalent to processing each frame independently through a convolution block with only one convolution kernel. During this process, due to the dimension transformation that has already occurred, the convolution operation will be performed along the time dimension (dim = 2), which significantly differs from the typical operation along the channel dimension. This design allows for the parallel processing of all input frames, resulting in higher computational efficiency and parameter utilization compared to the other two methods. Additionally, it provides a more comprehensive way to capture and understand the temporal dependency features in dynamic motion cues. The entire process can be expressed as $D = S - \Gamma(S)$, where $D \in R^{(T-1)\times 3\times H\times W}$ represents the difference maps and Γ represents the generation method.

3.2.3. Motion Backbone

For the choice of the backbone in the motion branch, we deviate from the commonly used ResNet50 [35] in previous methods, and instead opt for introducing the vision transformer [36–40] backbone. Initially, we conduct preliminary experiments using several self-attention blocks on two movie shot datasets. After practical experimentation and analysis (we elaborate the process in detail in Section 4.3), we choose the Multiscale Visual Transformer [40] (MViT) with a multiscale feature hierarchies structure as our motion backbone.

The MViT backbone is a relatively optimal choice determined after extensive experimentation. Simultaneously, we discover that not all video transformer architectures can

serve as suitable motion backbones. For the task of shot attribute analysis, whether utilizing motion cues as input or directly employing frame sequences as input, the most critical aspect is the low-level semantic information. This observation might have been overlooked in prior related studies. Subsequent ablation experiments in Section 4.5 can further substantiate our conclusion.

The entire MViT Backbone can be represented by the following equation:

$$\hat{Q} = \overline{D}W_Q \quad \hat{K} = \overline{D}W_K \quad \hat{V} = \overline{D}W_V \tag{1}$$

$$V_{motion} = Softmax(\mathcal{P}(\hat{Q})\mathcal{P}(\hat{K})^T/\sqrt{d})\mathcal{P}(\hat{V}) \tag{2}$$

where $\overline{D}$ denotes the frame difference map after fixed-size adjustment; $\hat{Q}$, $\hat{K}$, $\hat{V}$ denote the query, key and value in the self-attention operator; W_Q, W_K, W_V denote the corresponding weight matrix; $\mathcal{P}$ denotes the pooling attention; and V_{motion} denotes the obtained motion feature vector.

3.2.4. Fixed-Size Adjustment

Training a vision transformer requires a large-scale annotated dataset. in the design of the motion backbone, movie shot datasets usually have a lack of shot samples (<10 K), which is insufficient to support training a transformer backbone from scratch. To better utilize various backbones pre-trained by Kinetics 400, we set $t = 16, c = 3, h = 224, w = 224$ as the standard backbone input size. Then, we introduce a fixed-size adjustment module to resize the frame difference map D to this size. Specifically, we used linear interpolation to adjust D, ensuring that the input size of the motion backbone matches the input size of the pre-trained model. This process can be represented as $\overline{D} = F_{interpolate}(D)$. Additionally, the fixed-size adjustment module indicates that our architecture can use shot clips of arbitrary length and sample rate as input.

3.3. Static Branch

Based on practical experience, for some intrinsic attributes of shots, such *scale*, we can directly judge whether it is a close-up or a medium shot from any frame of the shot. Therefore, for the static branch, we believe that using the entire sequence of frames as input, like a traditional two-stream network, would introduce a significant amount of redundant information. Instead, it is common to select key frames from the shot as input. However, we have found that in movie shots, there are rarely meaningless frames; thus, utilizing one keyframe to represent static information within a shot is a highly intuitive solution. In our method, we directly select the intermediate frame of the shot as the static input. For the static backbone, we choose ViT [17] pre-trained by Image21K. The process of the static branch can be expressed as $V_{static} = ViT(S_{[:,t//2,:,:]})$, where V_{static} denotes the static feature outputted from the static branch.

3.4. Quantitative Feature Fusion

Upon obtaining the motion feature V_{motion} and the static feature V_{static}, a direct approach would involve concatenating these two vectors and then passing them through a classification layer to obtain predictive outcomes. However, we point out that this could lead to interference between two features (as the weight parameters of the fully connected layer would be applied to all elements of both vectors simultaneously). In order to maintain the independence of motion and static information, while also quantitatively assessing the contribution from the branches, we add a trainable parameter that allows the network to automatically learn the contribution weights of the branches (refer to results in Section 4.4). The entire process can be expressed using the following formula: $C_i = Classifier(\lambda * V_{motion} + (1 - \lambda) * V_{static})$, where λ denotes the trainable parameter.

4. Experiment

4.1. Datasets

In this paper, two large-scale movie shot datasets, MovieShots [1] and AVE [24], are utilized. MovieShots is the first large public movie shot dataset with complete shot sample clips constructed by the CUHK—SenseTime Joint Lab. It contains 46 K shots extracted from 7 K movie trailers and is meticulously annotated for *scale* and *movement*. AVE is a dataset constructed by Adobe Research and KAIST for AI-assisted video editing, containing 196 K shots taken from 5 K movie clips, annotated with shot attributes such as *shot size*, *shot angle*, *shot motion*, and *shot type*. Table 1 shows the statistics of both dataset.

Table 1. Statistics of MovieShots and AVE.

Dataset		Train	Val	Test	Total	Avg. Duration of Shots (s)
Movieshots	Num. of movies	4843	1062	1953	7858	——
	Num. of shots	32,720	4610	9527	46,857	3.95
AVE	Num. of scenes	3914	559	1118	5591	——
	Num. of shots	151,053	15,040	30,083	196,176	3.81

4.2. Experiment Setup

4.2.1. Data

Following [1,24], we split MovieShots and AVE into training, validation, and test sets at a ratio of 7:1:2. All scenes are unique across the training, validation, and test sets. Table 1 displays the statistical information of the two datasets.

4.2.2. Implementation Details

Due to the prevalent sample imbalance in MovieShots and AVE (e.g., static shots account for over 60% in *movement* and eye-level shots account for over 70% in *shot-angle*), we adopt focal loss [41] instead of conventional cross entropy as the loss function for shot attribute analysis tasks. The focal loss function is defined as:

$$\begin{aligned} \mathrm{FL}(p_\mathrm{t}) &= -\alpha(1-p_\mathrm{t})^{\gamma}\log(p_\mathrm{t}) \\ p_t &= \begin{cases} p, (y=1) \\ 1-p, (y=0) \end{cases} \end{aligned} \tag{3}$$

where p_t is the predicted probability for the ground truth class, α represents the class weight, and γ represents the tuning parameter. In our implementation, we set α to 1 and γ to 2.

To meet our goal of directly applying existing video classification pre-trained models for shot classification, we favor publicly accessible pre-trained models for the motion backbone and static backbone without altering model hyperparameters like layer numbers. After extensive experiments (Tables 2, 3, and 5), we select the MViT-B [40] pre-trained by Kinetics 400 as the motion backbone, and ViT-Base [17] pre-trained by Image21K as the static backbone. For the frame difference generator, Depthwise-Diff achieves better performance than the other two methods in most tasks, so we set it as the default function. For the Frame Difference Generator, Depthwise-Diff achieves better performance than the other two methods in most tasks, so we set it as the default. For quantitative feature fusion, we find that a feature dimension of 768 provides a good trade-off between model complexity and representational capacity.

Following [24], we uniformly sample 16 frames from a shot clip as input. During the training process, we configure the batch size to be 8 and employ the mini-SGD optimizer with a momentum of 0.9. The initial learning rate is set at 0.002, and we utilize an exponential learning rate decay strategy with a gamma value of 0.9. All models are trained for 10 epochs. All experiments are conducted on one RTX4090 and utilizing Pytorch Lightning to construct the entire shot attribute analysis system.

4.2.3. Evaluation Metrics

Due to the long-tail distribution problem, in addition to adhering to the practice of using Top-1 accuracy from previous works, we also adopt Macro-F1 as an evaluation metric.

Top-1 accuracy is defined as the percentage of test examples for which the model's top predicted class matches the true label. More formally, given a dataset of N examples $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$, Top-1 accuracy is calculated as:

$$\text{Top-1 Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{I}(argmax_c P(y = c|x_i) = y_i) \tag{4}$$

where $P(y = c|x)$ is the model's predicted probability distribution over classes c given input x, and $\mathbb{I}(condition)$ is an indicator function that equals 1 if the condition is true, else 0.

Macro-F1 is the macro-averaged F1 score over all classes. F1 score for a class is the harmonic mean of precision and recall:

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{5}$$

Macro-F1 averages the F1 scores across classes, treating each class equally:

$$\text{Macro-F1} = \frac{1}{C}\sum_{c=1}^{C} F1_c \tag{6}$$

where C is the number of classes and $F1_c$ is the F1 score for class c.

4.3. Overall Results

4.3.1. MovieShots

Table 2 displays the overall results and computational costs of some methods on MovieShots [1]. Among them, only MUL-MOVE-Net [7] and SGNet (img + flow) [1] use optical flow maps as inputs, while other methods solely utilized shot data as input. To better compare with our designed architecture, we additionally reproduce CNN-based methods such as R3D [42], X3D [43], SlowFast-Resnet50 [44], and transformer-based methods like TimeSformer [36], ViViT [38], and MViT [40] (all pre-trained by the K400 dataset). SGNet (img + flow) is the baseline.

Three traditional methods, DCR [45], CAMHID [10], and 2DHM [11], are strictly limited in their performance in *scale* and *movement* since they use hand-crafted features. Among the CNN-based methods, we find that I3D-Resnet50, TSN-Resnet50, SlowFast-Resnet50, and X3D have relatively high accuracy in *scale*, while R3D have significantly higher results in *movement* than the other methods. These results can verify our previous hypothesis: shots have both strong temporal attributes (*movement*) and weak temporal attributes (*scale*). When undergoing end-to-end training, methods leaning towards strong temporal attributes tend to perform better in such attributes but worse in weak temporal ones.

SGNet (img + flow), by using both optical flow maps and shot frames as input, improves significantly by 9.35 in *movement* over SGNet (img) while maintaining accuracy in *scale*. It should be noted that although SGNet, like our proposed architecture, also uses motion cues as input and utilizes a dual-stream network structure, its emphasis lies in segmenting movie shots into foreground and background, i.e., analyzing foreground–background features, which is fundamentally different from our method's motion–static features. Following SGNet, Bias CNN and MUL-MOVE-Net w/ flow make some improvements in efficiency and accuracy, but are only suitable for single attribute analysis tasks.

Table 2. The overall results on MovieShots.

Models	Scale		Movement		GFLOPs
	Accuracy	Macro-F1	Accuracy	Macro-F1	
DCR [45]	51.53	——	33.20	——	——
CAMHID [10]	52.37	——	40.19	——	——
2DHM [11]	52.35	——	40.34	——	——
I3D-Resnet50 [4]	76.79	——	78.45	——	——
TSN-ResNet50 [6]	84.08	——	70.46	——	——
R3D [42]	69.35	69.76	83.73	42.48	163.417
X3D [43]	75.55	75.88	67.15	20.09	5.091
SlowFast-Resnet50 [44]	70.43	70.99	68.59	32.90	6.998
SGNet (img) [1]	87.21	——	71.30	——	——
SGNet (img + flow) [1] *	87.50	——	80.65		
MUL-MOVE-Net [7]	——	——	82.85	——	——
Bias CNN [8]	87.19	——	——	——	——
TimeSformer [36]	89.15	89.39	84.21	41.38	380.053
ViViT [38]	74.67	75.05	75.96	35.32	136.078
MViT [40]	87.54	87.84	86.24	43.13	56.333
Ours (depthwise-diff)	**89.46**	**89.73**	85.68	43.40	66.570
Ours (conv-diff)	89.00	89.28	**86.46**	43.18	66.610
Ours (frame-diff)	89.11	89.39	86.25	**44.02**	66.549

* baseline.

Among the transformer-based methods, we find that TimeSformer improves by 1.65 and 3.56 compared to the baseline, but its computational demand reaches a staggering 380GFLOPs. In contrast, ViTiT decreases by 12.83 and 4.69. We believe a possible reason is that ViTiT first uses ViT to convert the shot input into high-level semantic features before performing factorized self-attention. Therefore, it is essential to retain as much of the origin input (or low-level semantic features) as possible when extracting features for shot attribute analysis tasks. MViT improves by 0.04 and 5.59 over the baseline, further enhancing performance in strong temporal attributes. In our designed architecture **Ours(depth-wise-diff)**, while using MViT as the motion backbone, also has a static branch, effectively complementing weak temporal attributes. The results show that our method is 1.96 and 5.03 higher than the baseline, and the computational demand is only 1/6 of TimeSformer's. Moreover, although our method is 0.56 lower than MViT in Top-1 accuracy in movement, it is 0.27 higher in Macro-F1.

4.3.2. AVE

AVE [24], compared to MovieShots, has more shot attribute labels. Table 3 shows the experimental results on AVE. Since the original paper uses both video and audio as inputs, and the vision backbone is R3D, we also choose to use the reproduced R3D as our baseline. On AVE, due to the more uneven distribution of shot sample categories compared to MovieShots, we choose Macro-F1 as the analysis metric.

Among the CNN-based methods, R3D, X3D, and SlowFast-Resnet50 performed far below expectations for the four shot attributes. We believe that in shot attribute analysis tasks, once the imbalance in shot sample categories becomes too high, using the convolutional neural network structure can result in the analysis leaning more towards categories with a higher sample proportion. Among the transformer-based methods, ViViT's performance in *shot angle* and *shot motion* was 0.62 and 0.11 lower than the baseline, 3.98 and 5.84 higher in *shot size* and *shot type*, respectively. This conclusion, consistent with that on MovieShots, shows that it is actually not suitable for shot attribute analysis tasks. In contrast, TimeSformer sees significant improvements over the baseline: 19.55 (*shot size*), 21.65 (*shot angle*), 4.02 (*shot motion*), and 30.99 (*shot type*). Our proposed method, **Ours(depthwise-diff)**, also

improves by 19.64, 15.67, 9.09, and 31.57, achieving the best results in three shot attributes: *shot size*, *shot motion*, and *shot type*.

Table 3. The overall results on AVE.

Models	Shot Size		Shot Angle		Shot Motion		Shot Type	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
Naive (V+A) [24]	39.1	——	28.9	——	31.2	——	62.3	——
Logit adj. (V+A) [24]	67.6	——	49.8	——	43.7	——	66.7	——
R3D [42] *	67.45	24.97	85.37	19.05	70.18	29.02	55.19	34.43
X3D [43]	65.44	15.82	85.44	18.43	69.41	28.74	46.39	10.56
SlowFast-Resnet50 [44]	66.10	20.38	81.13	19.72	68.17	29.40	52.25	33.55
TimeSformer [36]	72.94	44.52	**87.35**	**40.70**	71.16	33.04	75.94	65.42
ViViT [38]	69.30	28.95	85.44	18.43	70.11	28.91	57.96	40.27
MViT [40]	**73.54**	41.27	87.19	33.04	**71.63**	36.62	75.82	65.15
Ours(depthwise-diff)	73.39	**44.61**	86.91	34.72	70.85	**38.11**	76.28	66.00
Ours(conv-diff)	73.09	40.77	86.55	**35.42**	70.64	31.93	**76.29**	**66.80**
Ours(frame-diff)	73.34	41.32	86.86	30.85	71.35	31.11	74.06	63.52

* baseline.

4.3.3. Analysis of Frame Difference Methods

We further analyze the results of different frame difference methods on both MovieShots and AVE. For a fair comparison, we only modify the frame difference generation module in all experiments, keeping the rest of the network unchanged. On MovieShots, depthwise-diff surpasses conv-diff and frame-diff in Top-1 accuracy and Macro-F1 on *scale*, but is slightly lower than frame-diff in *movement*. On the AVE dataset, depthwise-diff significantly outperforms the other two frame difference methods in *shot size* and *shot motion*, and is less than 0.1 below the highest Macro-F1 in *shot angle* and *shot type*. This result demonstrates that our proposed depthwise-diff method is more suitable for our architecture in general compared to the other two methods.

4.4. Quantitative Analysis

In shot attribute analysis tasks, it is intuitive to feel that the motion branch is more suitable for strong temporal attributes, while the static branch is more suitable for weak temporal attributes. However, there has been a lack of numerical representation for this boundary. By visualizing the λ value in feature fusion module, we have quantitatively analyzed the contributions of motion and static features in the results of each shot attribute, as shown in Figure 3. We call the result of dividing the contribution percentage of motion branch by the contribution percentage of static branch **Temporal Tendency Coefficient of Shot Attributes**. The higher this value, the stronger the temporal dependency of the shot attribute. In MovieShots, the tendency coefficients for *movement* and *scale* are 2.00 and 1.03, respectively. In AVE, they are 2.01, 1.80, 2.56, and 1.90 for *shot size*, *shot angle*, *shot motion*, and *shot type*, respectively. We believe that this value can guide the design of methods for single shot attribute recognition tasks (e.g., shot motion recognition). Using 2 as a boundary, attributes >2 can choose to use a temporal model structure, while those <2 can opt for single-frame input and non-temporal structures.

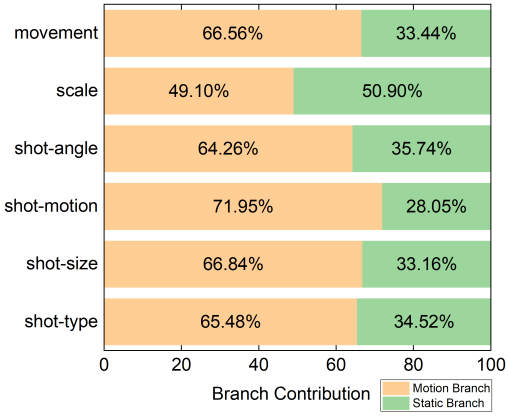

Figure 3. The quantitative contribution of motion and static feature to each shot attribute.

4.5. Ablation Study

We conducted the following ablation experiments to demonstrate the effectiveness of our proposed architecture: (1) input length (video sample rate); (2) motion/static branch; (3) vision backbone; (4) pre-trained models.

4.5.1. The Influence of Different Shot Input Duration

Our proposed fixed-size adjustment strategy allows the model to accept shot input of any duration. To analyze the impact of shot input duration on the results, we experiment with different counts of the sample frames . We respectively use 2 (video sampling rate 1/64), 4 (1/32), 8 (1/16), and 16 (1/8) as input duration. Figure 4 shows the accuracy and Macro-F1 variation curves on two datasets. In most cases, as the input duration increases (sampling rate decreases), the shot attribute analysis results are better. We find that the *temporal tendency coefficient* of lower *scale* and *shot angle* do not change significantly with input duration. Moreover, due to our fixed-size adjustment strategy, the computational load of the entire architecture does not decrease significantly as the video sampling rate increases, which is one of the drawbacks of our architecture. We believe that, similar to action recognition, using a 1/8 sampling rate and 16 frames as input is suitable for most movie shots.

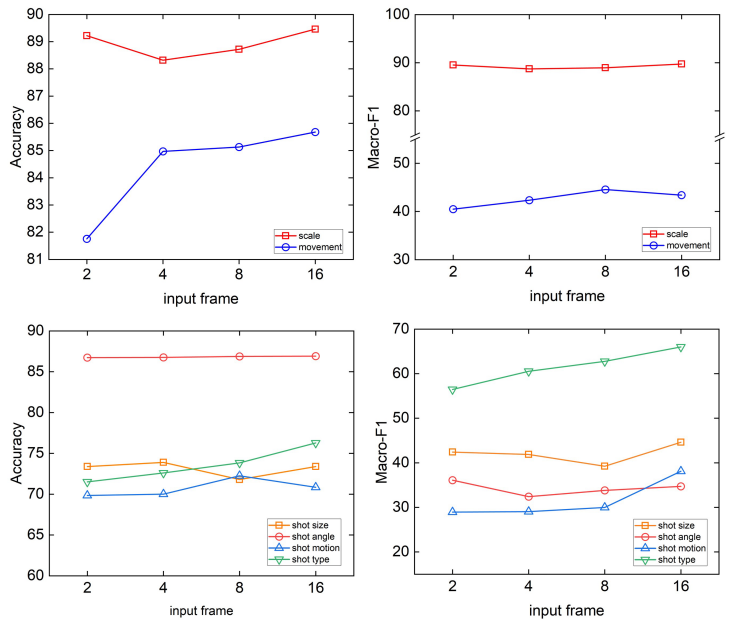

Figure 4. Accuracy and Macro F1 on MovieShots and AVE with sample # frames.

4.5.2. The Influence of Motion Branch and Static Branch

Table 4 shows the impact on results when using a single branch. When only the motion branch is used, there is a slight (<3) decline in results on *scale*, *shot size*, *shot angle*, and *shot type*, while there is a very minor increase in accuracy on *movement* and *shot motion*, but a decrease in macro-F1. When only the static branch is used, *movement* and *shot motion* decrease by 13.77 (accuracy)/10.99 (macro-f1) and 12.66/15.46, respectively, while *shot type* decrease by 3.78/5.12, and *scale*, *shot size*, and *shot angle* remain essentially unchanged. This indicates that the motion branch plays a leading role in our architecture and using only the frame difference map as the input can also achieve good results, while the static branch complements the low-level semantic information ignored by the motion branch.

Table 4. Ablation study on utilizing single branch.

	Scale		Movement		Shot Size		Shot Angle		Shot Motion		Shot Type	
Branch	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**
motion	86.88	87.13	86.14	42.82	72.82	34.87	86.32	35.05	71.68	34.37	74.65	63.97
static	89.12	89.42	71.91	32.41	73.43	40.62	86.54	32.47	58.19	22.65	72.50	60.88

4.5.3. The Influence of Visual Backbones

Table 5 shows the impact on results when using different visual backbones. Specifically, we choose R3D [42] as a substitute for the motion backbone and ResNet50 [35] as a substitute for static backbone. Compared with our benchmark configuration (MViT + ViT), the results for *scale*, *shot size*, *shot angle*, and *shot type* declined when using MViT + ResNet50, while *movement* and *shot motion* increased. A possible explanation is that ResNet50, due to its convolutional block, tends to extract high-level semantic features of static images, while ViT, which does not have an inductive bias, is more likely to learn low-level semantic features of the images. When using R3D + ViT, the computational cost increases by 113GFLOPs, but the results on all shot attributes show a significant decline. As with action recognition tasks, after sufficient training, transformer-based methods generally outperform methods based on the CNN-based method. However, we also point out that not all transformer-based backbones are effective for shot attribute analysis tasks. The essence of using the vision transformer is to utilize its lack of inductive bias to make the structure learn more low-level semantic features during training.

Table 5. Comparison of different motion backbones and static backbones.

		Scale		Movement		Shot Size		Shot Angle		Shot Motion		Shot Type	
Motion Backbone	**Static Backbone**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**	**Macro-F1**
MViT	ResNet50	86.81	87.02	87.20	43.79	72.98	39.90	87.04	31.08	71.86	31.91	76.05	65.02
R3D	ResNet50	73.89	74.13	80.69	39.14	68.43	29.46	84.84	18.93	63.50	26.32	50.11	27.71
R3D	ViT	89.30	89.60	80.79	39.84	73.11	35.35	86.51	31.44	68.38	28.23	70.67	57.77

4.5.4. The Influence of Pre-Trained Models

To analyze the importance of pre-trained models in the shot attribute analysis task, we choose MovieShots representing movie shot datasets with fewer than 30 K shots and train the network after random initialization for 60 epochs. The results in Table 6 show that if training starts from random initialization, the results on scale decline by 23.2/22.97, while the performance on movement only decreases by 2.69/2.62. The statistics from [24] indicate that most movie shot datasets have a sample size of less than 10 K, which is far from the requirement for training transformer blocks from scratch. One of the purposes of fixed adjustment strategy is to make our architecture usable for most models pre-trained by action recognition datasets.

Table 6. Ablation study on utilizing pre-trained models.

	Scale		Movement	
	Accuracy	**Macro-F1**	**Accuracy**	**Macro-F1**
With Pretrain (10 epoch)	89.46	89.73	85.68	43.40
From scratch (60 epoch)	66.26	66.76	82.99	40.78

5. Conclusions

In this paper, we first identify two issues in the shot attribute analysis task. Starting from this, we design a unified, quantified end-to-end architecture. Through the motion-static dual-branch structure, our method can adaptively learn the feature weights of motion clues and static keyframes from input shots, thus adapting to various shot attribute classification tasks. Experiments on MovieShots and AVE show that our proposed architecture significantly outperforms all previous approaches. However, the long-tail problem of movie shot category distribution has not been resolved. In our next step, we plan to improve training strategies for unbalanced shot samples, so that the architecture can accurately analyze shot attributes with fewer samples.

Author Contributions: Conceptualization, Y.L.; methodology, Y.L. and T.L.; software, Y.L. and H.X.; validation, Y.L. and T.L.; formal analysis, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and F.T. 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.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rao, A.; Wang, J.; Xu, L.; Jiang, X.; Huang, Q.; Zhou, B.; Lin, D. A unified framework for shot type classification based on subject centric lens. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Part XI 16, pp. 17–34.
2. Souček, T.; Lokoč, J. Transnet v2: An effective deep network architecture for fast shot transition detection. *arXiv* **2020**, arXiv:2008.04838.
3. Rao, A.; Xu, L.; Xiong, Y.; Xu, G.; Huang, Q.; Zhou, B.; Lin, D. A local-to-global approach to multi-modal movie scene segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10146–10155.
4. Carreira, J.; Zisserman, A. Quo vadis, action recognition? A new model and the kinetics dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 6299–6308.
5. Liu, C.; Pei, M.; Wu, X.; Kong, Y.; Jia, Y. Learning a discriminative mid-level feature for action recognition. *Sci. China Inf. Sci.* **2014**, *57*, 1–13.
6. Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; Van Gool, L. Temporal segment networks: Towards good practices for deep action recognition. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 20–36.
7. Chen, Z.; Zhang, Y.; Zhang, L.; Yang, C. RO-TextCNN Based MUL-MOVE-Net for Camera Motion Classification. In Proceedings of the 2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall), Xi'an, China, 26–28 October 2021; pp. 182–186.
8. Chen, Z.; Zhang, Y.; Zhang, S.; Yang, C. Study on location bias of CNN for shot scale classification. *Multimed. Tools Appl.* **2022**, *81*, 40289–40309. [CrossRef]
9. Simonyan, K.; Zisserman, A. Two-stream convolutional networks for action recognition in videos. In Proceedings of the Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, Montreal, QU, Canada, 8–13 December 2014; p. 27.
10. Hasan, M.A.; Xu, M.; He, X.; Xu, C. CAMHID: Camera motion histogram descriptor and its application to cinematographic shot classification. *IEEE Trans. Circuits Syst. Video Technol.* **2014**, *24*, 1682–1695. [CrossRef]

11. Prasertsakul, P.; Kondo, T.; Iida, H. Video shot classification using 2D motion histogram. In Proceedings of the 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Phuket, Thailand, 27–30 June 2017; pp. 202–205.
12. Dosovitskiy, A.; Fischer, P.; Ilg, E.; Hausser, P.; Hazirbas, C.; Golkov, V.; Van Der Smagt, P.; Cremers, D.; Brox, T. Flownet: Learning optical flow with convolutional networks. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 2758–2766.
13. Hui, T.-W.; Tang, X.; Loy, C.C. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18-22 June 2018; pp. 8981–8989.
14. Ilg, E.; Mayer, N.; Saikia, T.; Keuper, M.; Dosovitskiy, A.; Brox, T. Flownet 2.0: Evolution of optical flow estimation with deep networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2462–2470.
15. Mayer, N.; Ilg, E.; Hausser, P.; Fischer, P.; Cremers, D.; Dosovitskiy, A.; Brox, T. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 4040–4048.
16. Wang, X.; Zhang, S.; Qing, Z.; Gao, C.; Zhang, Y.; Zhao, D.; Sang, N. MoLo: Motion-augmented Long-short Contrastive Learning for Few-shot Action Recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 18011–18021.
17. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
18. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-end object detection with transformers. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 213–229.
19. Bhattacharya, S.; Mehran, R.; Sukthankar, R.; Shah, M. Classification of cinematographic shots using lie algebra and its application to complex event recognition. *IEEE Trans. Multimed.* **2014**, *16*, 686–696. [CrossRef]
20. Canini, L.; Benini, S.; Leonardi, R. Classifying cinematographic shot types. *Multimed. Tools Appl.* **2013**, *62*, 51–73. [CrossRef]
21. Wang, H.L.; Cheong, L.-F. Taxonomy of directing semantics for film shot classification. *IEEE Trans. Circuits Syst. Video Technol.* **2009**, *19*, 1529–1542. [CrossRef]
22. Xu, M.; Wang, J.; Hasan, M.A.; He, X.; Xu, C.; Lu, H.; Jin, J.S. Using context saliency for movie shot classification. In Proceedings of the 2011 18th IEEE International Conference on Image Processing, Brussels, Belgium, 11–14 September 2011; pp. 3653–3656.
23. Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P. The kinetics human action video dataset. *arXiv* **2017**, arXiv:1705.06950.
24. Argaw, D.M.; Heilbron, F.C.; Lee, J.-Y.; Woodson, M.; Kweon, I.S. The anatomy of video editing: A dataset and benchmark suite for AI-assisted video editing. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 201–218.
25. Feichtenhofer, C.; Pinz, A.; Zisserman, A. Convolutional two-stream network fusion for video action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1933–1941.
26. Zhou, P.; Han, X.; Morariu, V.I.; Davis, L.S. Two-stream neural networks for tampered face detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Honolulu, HI, USA, 21–26 July 2017; pp. 1831–1839.
27. Liu, W.; Qian, J.; Yao, Z.; Jiao, X.; Pan, J. Convolutional two-stream network using multi-facial feature fusion for driver fatigue detection. *Future Internet* **2019**, *11*, 115. [CrossRef]
28. Bagheri-Khaligh, A.; Raziperchikolaei, R.; Moghaddam, M.E. A new method for shot classification in soccer sports video based on SVM classifier. In Proceedings of the 2012 IEEE Southwest Symposium on Image Analysis and Interpretation, Santa Fe, NM, USA, 22–24 April 2012; pp. 109–112.
29. Benini, S.; Canini, L.; Leonardi, R. Estimating cinematographic scene depth in movie shots. In Proceedings of the 2010 IEEE International Conference on Multimedia and Expo, Singapore, 19–23 July 2010; pp. 855–860.
30. Jiang, X.; Jin, L.; Rao, A.; Xu, L.; Lin, D. Jointly learning the attributes and composition of shots for boundary detection in videos. *IEEE Trans. Multimed.* **2021**, *24*, 3049–3059. [CrossRef]
31. Bak, H.-Y.; Park, S.-B. Comparative study of movie shot classification based on semantic segmentation. *Appl. Sci.* **2020**, *10*, 3390. [CrossRef]
32. Vacchetti, B.; Cerquitelli, T.; Antonino, R. Cinematographic shot classification through deep learning. In Proceedings of the 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 13–17 July 2020; pp. 345–350.
33. Vacchetti, B.; Cerquitelli, T. Cinematographic shot classification with deep ensemble learning. *Electronics* **2022**, *11*, 1570. [CrossRef]
34. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
36. Bertasius, G.; Wang, H.; Torresani, L. Is space-time attention all you need for video understanding? In Proceedings of the ICML, Virtual Event, 18–24 July 2021; p. 4.

37. Neimark, D.; Bar, O.; Zohar, M.; Asselmann, D. Video transformer network. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 3163–3172.
38. Arnab, A.; Dehghani, M.; Heigold, G.; Sun, C.; Lučić, M.; Schmid, C. Vivit: A video vision transformer. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 6836–6846.
39. Liu, Z.; Ning, J.; Cao, Y.; Wei, Y.; Zhang, Z.; Lin, S.; Hu, H. Video swin transformer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 3202–3211.
40. Fan, H.; Xiong, B.; Mangalam, K.; Li, Y.; Yan, Z.; Malik, J.; Feichtenhofer, C. Multiscale vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 6824–6835.
41. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
42. Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; Paluri, M. A closer look at spatiotemporal convolutions for action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6450–6459.
43. Feichtenhofer, C. X3d: Expanding architectures for efficient video recognition. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 203–213.
44. Feichtenhofer, C.; Fan, H.; Malik, J.; He, K. Slowfast networks for video recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6202–6211.
45. Li, L.; Zhang, X.; Hu, W.; Li, W.; Zhu, P. Soccer video shot classification based on color characterization using dominant sets clustering. In Proceedings of the Advances in Multimedia Information Processing-PCM 2009: 10th Pacific Rim Conference on Multimedia, Bangkok, Thailand, 15–18 December 2009; pp. 923–929.