

Wanet: weight and attention network for video summarization

Arpan Basu¹ · Rishav Pramanik¹ · Ram Sarkar¹

Received: 25 August 2023 / Accepted: 2 January 2024

Published online: 11 January 2024

© The Author(s) 2024 [OPEN](#)

Abstract

In this paper, we propose a deep learning-based model, called Weight and Attention Network (WANet), for video summarization. The network comprises a simple multi-head attention mechanism, followed by a feed-forward network to obtain the frame importance scores. Summary keyshots are obtained from the scores using a combination of kernel temporal segmentation and the knapsack algorithm. Contrary to past methods, we first enrich the input frames with similar information as opposed to letting the model learn all the features by itself. A novel weight assignment mechanism is introduced to assign weights to the input frames based on their similarity before passing the same to the model. Experimental results on the SumMe and TVSum datasets indicate the effectiveness of the present method when compared to state-of-the-art methods applied to the same datasets.

Keywords Video Summarization · Multi head attention · Weight Assignment Mechanism · Deep Learning · SumMe · TVSum

1 Introduction

In recent years, there has been a rapid growth in the amount of data stored as videos. To deal with this influx of video data, video summarization has emerged as a useful tool for extracting a summary of the entire content in the video. The main idea of video summarization is to compress the information contained in all the frames of the video into a smaller subset. This can be in the form of smaller extracts of the original video (keyshots), selected frames of the original video (keyframes), etc. The reason for doing so is that a video may contain repetitive, redundant and unimportant content in many of its frames. As a result, if a machine-aided process is developed to filter out these frames, then it can enable humans to quickly process a very large video in a short amount of time. Applications of video summarization can be found in the domains of surveillance (for detecting suspicious activities), live streams of sports (for capturing important events), etc.

However, video summarization is a very challenging task. To summarize a video, any algorithm would need to process the entire video to select the most informative frames. At the same time, there should be a restriction on the number of frames that are selected. Oftentimes, this means that a choice has to be made among candidate frames. This choice becomes hard, as it should emulate the subjective behaviour of a human from features extracted from the video itself. A good summarization algorithm would retain important frames while sticking to the restriction on the number of frames. Recent research in summarization follows a two-stage approach, where first the importance of frames is found and then

Arpan Basu and Rishav Pramanik contributed equally to this work.

✉ Rishav Pramanik, rishavpramanik@gmail.com; Arpan Basu, arpan0123@gmail.com; Ram Sarkar, ramjucse@gmail.com | ¹Department of Computer Science and Engineering, Jadavpur University, Kolkata, India.



the key shots are generated by leveraging the importance scores of the previous stage. In many cases, this involves the use of a related algorithm, kernel temporal segmentation (KTS), to find the shot boundaries. Considerable progress in video summarization has been made in recent years due to advances in deep learning (DL) and computer vision. Most approaches nowadays use convolutional neural networks (CNNs) to first extract the features from the frames before proceeding with one of the three approaches mentioned below.

From a DL perspective, video summarization methods broadly fall into three basic approaches: supervised, semi-supervised and unsupervised. Supervised approaches utilize the training data as well as the ground-truth labels in the model training set. When compared to the other two categories, supervised approaches generally obtain higher performance since they directly use ground-truth annotations. State-of-the-art methods in video summarization mostly follow a supervised approach. However, this requirement of labels or annotations is also a drawback as in many cases, it is not possible to obtain human-annotated labels. Methods in this category, generally use DL techniques such as Long Short Term Memory (LSTM), and attention mechanisms, and recently, we found the use of graph neural networks [1, 2]. A few methods in this category are discussed in sect 2.

Coming to the other two categories, semi-supervised approaches require a smaller amount of ground truth data while unsupervised approaches generally do not need the ground-truth labels. Semi-supervised methods do not directly need all the ground-truth labels. Instead, they try to leverage some auxiliary data that is easily available in the training process. For example, web-image priors [3], video categories [4], etc. can be used for training as opposed to the actual frame importance scores. Unsupervised methods, on the other hand, try to use some heuristics to find the keyshots. Reinforcement learning-based methods [5] use loss functions modelled on the diversity and representativeness of frames modelled using a suitable distance function. Regularization methods are also applied to the length of the keyshots and the model weights. Generative adversarial network (GAN) based models [6–8] also exist which use reconstruction and adversarial losses in addition to the regularization based on diversity, summary length, etc.

Contributions: In the present work, we present a DL network which comprises a multi-head attention layer followed by a feedforward network to predict the frame importance scores from the given input frames. A novel weight calculation mechanism is also developed which assigns weights to the frames based on their similarity before they are passed through the network. This causes the features from the dissimilar frames to become more prominent than the features from similar frames. Experimental evaluation of the proposed method on the benchmark SumMe and TVSum datasets indicates the effectiveness of the approach.

Precisely, the contributions of this paper are as follows:

- A novel weight assignment mechanism is proposed to enrich the raw features. The present method introduces a guided search over the input features and shows the effectiveness of the same.
- The use of noise is introduced to improve the robustness of the model. We provide an online data augmentation technique, which significantly improves the performance of the model. The use of noise reduces overfitting and helps to generalize the model.
- A modified hybrid loss function is proposed for optimization of the deep learner. A common loss function used in regression tasks, mean squared error, is combined with the benefits of the Kullback-Leibler Divergence.

The remainder of the work is structured as follows: sect 2 presents some recently published methods pertaining to the present topic. The proposed approach is discussed in detail in sect 3. Sect 4 contains the experimental findings and corresponding analysis. Finally, we end with the conclusion in sect 5.

2 Related work

The method proposed by Zhang et al. [9] was one of the pioneering methods in video summarization. It was one of the first applications of DL in this domain and laid the groundwork for future DL-based works. In the said work, the authors used a bidirectional LSTM network for supervised video summarization to model the variable-range temporal dependencies among video frames to derive both representative and compact video summaries. The input to the LSTM comprised spatial features which were extracted from the input frames using a pre-trained GoogleNet model. The performance of the LSTM network was further enhanced using the determinantal point process (DPP) to increase the diversity among the selected frames. The method achieved state-of-the-art results as compared to other approaches at the time of its

publication. In addition, it also laid down several guidelines on the datasets, testing protocols and interconversions between ground-truth annotations which are relevant even today.

In the work reported by Rochan et al. [10], the authors have framed the problem as a sequence labelling problem and have used a fully convolutional sequence model for summarization. As the receptive field size increases, going deeper into the network, long-range dependencies among the frames can be captured effectively. When compared to LSTM-based networks, the architecture allows for parallelization over input frames. The authors obtained better performance when compared to [9]. The work also presented an extension of the model which could be trained in an unsupervised setting using diversity and reconstruction losses.

The work by Mahesseni et al. [6] was among the first works in video summarization to use a generative adversarial framework. The basic framework used by the authors comprises two networks: a summarizer and a discriminator. The summarizer takes as input the deep features corresponding to the input frames. After that, it uses selector and encoder LSTMs to select and encode the frames. Finally, it uses a decoder LSTM to obtain a sequence of features from the deep features of the encoder LSTM. The discriminator network consists of a classifier LSTM with binary output. It helps to estimate the representation error between original frames and summarizer outputs. The training is accomplished using commonly applied adversarial and reconstruction losses. In addition, the authors have also explored the use of several regularizations like summary length regularization and diversity regularization using DPP and repelling regularizer (REP). Finally, the authors have also trained the model in a supervised setting using a keyframe-based regularization which uses the ground truth summaries.

The above-mentioned works showed the applicability of LSTMs, Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs) for video summarization, and recent works also build on top of similar ideas. Conventional LSTMs and similar networks are computationally taxing due to their recurrent nature. To overcome this, Transformers [11] were proposed in natural language processing (NLP). At their core, Transformers still perform sequence-to-sequence translation but only use attention to do so. Similar to this, an important experiment was performed by [12] in their work on video summarization using an attention mechanism. The authors used a simple soft self-attention mechanism to obtain state-of-the-art results in supervised video summarization. An encoder-decoder entirely substituted LSTMs with an attention network followed by a two-layer fully connected network for the prediction of frame-importance scores. The authors used the process reported in [9] to convert the importance scores to keyshot summaries. Following the work, several other authors have adopted the concept of attention either directly or as an intermediate step.

The work by Huang et al. [13] was a recent work, where a novel approach was proposed for video content and motion summarization. Here, the authors used a Capsule network as the initial feature extractor as opposed to GoogleNet which has been used by many authors. From these features, the authors have generated an inter-frame motion curve. Following this, the authors have used a novel transition effects detection (TED) method based on the application of G^1 -continuity error and cubic Bezier curves within local sliding windows to segment the video stream into keyshots. Finally, a self-attention network is used to obtain the final keyframes for content summarization. The authors have obtained competitive results on datasets such as SumMe and TVSum.

The work by Ji et al. [14] was another recent work, where the entire summarization process was treated as a sequence-to-sequence translation problem. The authors have considered the input frame features as the source sequence and the output keyshots as the target sequence. An encoder-decoder-based network with attention has been applied to the task. The encoder and decoder are bidirectional and unidirectional LSTMs respectively. The decoder contains the attention mechanism, and the authors have proposed two variants of their model based on the attention mechanism. A-AVS model uses additive attention, whereas the M-AVS model uses multiplicative attention. It is interesting to note that additive [15] and multiplicative [16] attention mechanisms have been used in NLP as well for similar tasks. The models proposed by the authors obtained good results on the SumMe and TVSum datasets.

The work by Zhu et al. [17] was a recent work that holds state-of-the-art results for both the SumMe and TVSum datasets. In the paper, the authors have presented a novel Detect-to-Summarize network (DSNet) framework for supervised video summarization. DSNet comprises anchor-based and anchor-free variants. In the anchor-based approach, the authors first extracted deep features from the input frames. Then, long-range features extracted via self-attention are integrated with the deep features using summation to produce the final features. Thereafter, temporal interest proposals are generated at multiple scales, which is followed by temporal average pooling. This is ultimately followed by frame importance classification and location offset regression and subsequent keyshot generation. The anchor-based approach is perhaps very related to object detection where similar region proposals are used. Additionally, the approach is the first to consider the variable-length duration of keyshots. The anchor-free approach was proposed by the authors to address some drawbacks of the anchor-based approach like the class imbalance caused due to densely sampled proposals and

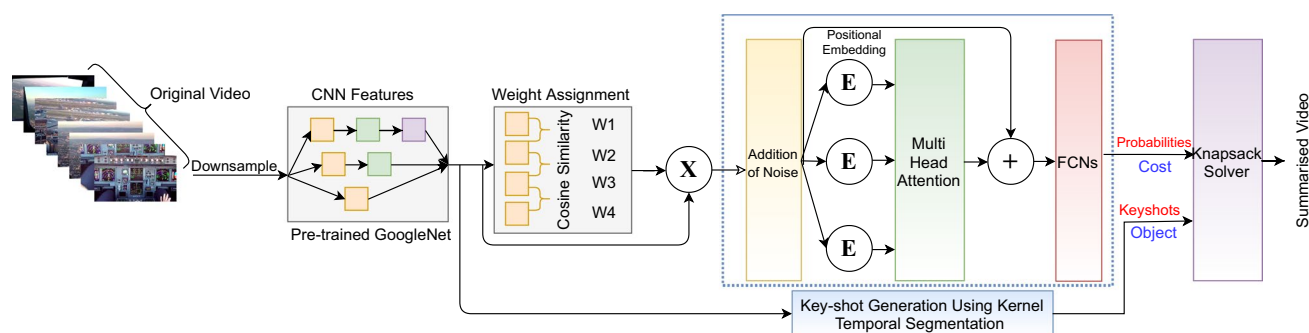


Fig. 1 A pipeline of the proposed WANet architecture. Initially, the videos are downsampled from the original 30fps to 2fps. Frame-level deep features are extracted from *pool5* layer of the GoogleNet model (pre-trained on the ImageNet dataset). Frame-level features are then assigned weights based on cosine similarity which is fed to a deep model comprising of multi-head Attention and a feedforward neural network to extract the probability of each frame. Finally, we use these extracted probability scores as cost parameters, and keyshots are generated using the KTS algorithm which is used as the object parameter for the classical knapsack problem. At the end, the summarized video is obtained

the fine-tuning required for better performance. On the whole, the network is similar to the anchor-based approach, but it does not contain multi-scale proposals. Instead, the importance score, segment boundaries and the centerness score are directly predicted. The authors have also conducted a detailed ablation study where various aspects of their model like the choice of long-range feature extractor, use of temporal pooling, etc. have been discussed.

Very recently, there have also been some works that attempt to use graph neural networks for video summarization. For example, the work by Zhao et al. [1] proposed a novel Reconstructive Sequence-Graph Network (RSGN) for video summarization which is trained using reinforcement learning. RSGN can be trained in both supervised and unsupervised modes: the only difference being the use of ground-truth summaries in the loss function. Another similar work is by Zhong et al. [2] where a graph attention network is used in combination with a bidirectional LSTM network for unsupervised video summarization

3 WANet approach

In this section, we highlight the detailed approach followed for generating summarized videos from original videos. Initially, frame-level deep features are extracted using a pre-trained CNN model. The frame level features are weighted using a weight assignment mechanism which is based on cosine similarity. These weighted features are further fed to a multi-head attention layer. Then these attentive features are fed to fully connected neural network layers to extract probability scores corresponding to each frame. Finally, to obtain the summarised videos, the KTS algorithm along with the knapsack algorithm is used. Figure 1 presents a schematic of the overall approach while the subsections below elaborate on the steps in more detail.

3.1 Feature extraction from videos using googleNet

A video is a collection of multiple sequences of events. The summarised video is to be based on a set of features extracted from the videos. Because of this, it is a challenging task to extract a meaningful and distinctive set of features for each of the frames.

For this task, we employ a well-known CNN architecture, called GoogleNet [18], to extract frame-level features from input videos. The CNN model is pre-trained on the ImageNet dataset. Following some previous methods [9, 17], we remove the last three layers of the pre-trained CNN model while extracting the features. The dimension of each of the feature vectors corresponding to each frame is 1024.

The hallmark of the GoogleNet architecture is to utilize the resources inside the model in the best possible manner. One fundamental change, when compared to earlier state-of-the-art CNN architectures, is the increase in the number of filters while reducing their size at the same time. This modification enabled the model to extract very deep features without using excessive computational resources. In addition, to tackle the vanishing gradient problem, the use of auxiliary classifiers was introduced for training the weights in the inner layers. The concept of global average pooling

Table 1 Comparison with state-of-the-art models in terms of F-Score and number of trainable parameters (in millions) on SumMe and TVSum datasets for canonical setting

Method	Approach	SumMe	TVSum	Parameters
SUM-GAN _{dpp} [6]	Unsupervised	0.391	0.517	295.86
DR-DSN [5]	Unsupervised	0.414	0.576	2.63
RSGN _{uns} [1]	Unsupervised	0.423	0.580	-
AC-SUM-GAN [8]	Unsupervised	0.508	0.606	-
vsLSTM [9]	Supervised	0.376	0.542	2.63
dppLSTM [9]	Supervised	0.386	0.547	2.63
SUM-GAN _{sup} [6]	Supervised	0.417	0.563	295.86
A-AVS [14]	Supervised	0.439	0.594	4.40
M-AVS [14]	Supervised	0.444	0.610	4.40
DR-DSN _{sup} [5]	Supervised	0.421	0.581	2.63
HSA-RNN [26]	Supervised	0.441	0.598	-
TC-STN [25]	Supervised	0.461	0.600	16.18
RSGN [1]	Supervised	0.450	0.601	-
SUM-FCN [10]	Supervised	0.488	0.584	-
VASNet [12]	Supervised	0.497	0.614	7.35
DSNet _{AB} [17]	Supervised	0.502	0.621	8.53
DSNet _{AF} [17]	Supervised	0.512	0.619	8.53
Ours	Supervised	0.560	0.644	5.25

The scores in bold indicate the best scores

was also presented in the work. It reduces the computational cost and also includes diverse feature maps for the fully connected layers to perform efficiently.

3.2 Weight assignment mechanism

The main aim of the weight assignment mechanism is to assign an attention factor to the dissimilar frames. The term “dissimilar” is relative in nature. From the video summarization perspective, dissimilar frames are of special interest since, in the end, we generate a probability score for each of the frames. This probability score determines how likely it is for a frame to be included in the summarized video. Therefore, this implies that certain dissimilar frames are more likely to be included in the summarized video. For this, dissimilar frames should be assigned some sort of attention or in other words, should be given higher scores. A frame can be dissimilar w.r.t its previous frame and the frame immediately after it. Considering this, we design a novel weight assignment scheme in this work. We assign weights based on relative dissimilarity at the frame level. The cosine similarity between two consecutive frames is computed based on the features extracted from the pre-trained GoogleNet model. The cosine similarity ($\text{sim}(x, y)$) is the similarity measure between two vectors x and y of an inner product space. We give the formulation in Eq. 1, wherein $\|x\|$ refers to the Euclidean length of the vector.

$$\text{sim}(x, y) = \frac{x \cdot y}{\|x\| \|y\|} \quad (1)$$

First, let us consider that there are N distinct frames for a given video. Cosine similarity scores for each of the consecutive frames are calculated. After taking into consideration all such scores we get $N - 1$ similarity measures mapped to N frames, with each measure corresponding to two consecutive frames. We then rank these measures in decreasing order and then assign a constant weight W to the first $R\%$ of pairwise frames corresponding to the ranked measures. The values of R and W are manually tuned. While tuning the value of W it should not be either too small or too big. If the value is too small, the motive of weight assignment will not be fulfilled, whereas if it is too big there will be a high chance of getting a sub-optimal solution. Algorithm 1 shows the detailed procedure of the proposed weight assignment mechanism. Through this, some attention (or more importance) is given to the dissimilar frames.

3.3 Deep model architecture

A deep learning based method is proposed to convert the weighted frame level features to probability scores for each of the frames. A detailed description of the model can be found in this subsection.

3.3.1 Noisy features

Typically, deep learners require an enormous amount of data to be properly trained. It is often observed that such models overfit when trained with a small quantity of data. This issue is relevant here as, in the present work, we have experimented on comparatively small-sized datasets. This causes the model to overfit the training data. Therefore, to overcome this problem and to improve the robustness of the model, Gaussian noise (having zero mean and a variance of 0.01) is added to the GoogleNet-based features in each training iteration. This form of online data alteration [19] introduces variability in the training data and reduces the risk of overfitting.

3.3.2 Loss

In the present work, we use a hybrid loss function to train our model. The proposed loss value is a hybrid of two popular loss functions: Mean Squared Error (MSE) and Kullback–Leibler Divergence (KLD). The loss value used to optimize the weights of the deep model can be found in Eq. 4. The loss is measured between the ground truth and the extracted probabilities from the DL model. The main motive for using a hybrid loss value is to incorporate the advantages of both losses. We combine the performance of MSE in supervised learning with the ability of KLD to distinguish between probability distributions. This leads to better performance as indicated in the experiments. MSE loss, on the one hand, gives a generic measure of divergence from the original distribution. Although this has been a powerful measure to optimize the deep models, MSE is symmetric and cannot provide directional values for proper optimization (equation 2). KLD, on the other hand, unlike MSE is not symmetric and it provides a direction [20] for better optimization of the internal weights (equation 3).

$$MSE(p(x), t(x)) = \int (p(x) - t(x))^2 dx \quad (2)$$

$$KLD(p(x), t(x)) = \int p(x) \log\left(\frac{p(x)}{t(x)}\right) dx \quad (3)$$

$$Loss = MSE + 2 \times KLD \quad (4)$$

3.3.3 Network architecture

The network architecture comprises a multi-head attention layer (Vaswani et al. [11]) followed by fully connected neural network layers. As the name itself suggests, a multi-head attention layer comprises multiple heads. The input sequence is split among n heads so that each head can learn specific features independently. These attentive outputs of these attention heads are finally concatenated and transformed to a dimension of the required size with the help of fully connected neural network layers. Multi-head attention, in this way, helps in capturing both long and short-range dependencies of the given sequence efficiently.

$$MultiHeadAttention = [head_1, head_2, \dots, head_n]W_0 \quad (5)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^K) \quad (6)$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (7)$$

For n heads, the attention mechanism works following eqs 5, 6 and 7. The parameter W is learnable. d_k is the dimension of K . Equation 7 is also known as the scaled dot product attention method. The output dimension of the multi-head attention is the same as the dimension of input features. In our case, it is 1024 for each frame. Q , K and V refer to query, key and value respectively according to the formulation in [11]. Here, the attention layer is followed by fully connected neural network layers. This is used to convert the attention features to a probability score. We use dropout as a regularization technique and we use layer-normalization as a re-scaling and re-centring method for fully connected neural network layers.

3.4 Summarized video generation

To generate the summarized video, we first use the KTS algorithm [4] to segment the video into several non-intersecting keyshots. The set of optimal change points for a particular video w.r.t the frame-level features are treated as the shot boundaries. These change points are based on frame-level similarities determined using a positive definite kernel.

After obtaining the keyshots, we assign probability scores to them based on the frame-level importance scores that are obtained from the model. As a keyshot comprises a group of continuous frames, we simply assign a score by taking the average of the importance scores of all the frames in that particular keyshot. Equation 8 shows how the shot level importance scores are generated.

$$s_i = \frac{1}{l_i} \sum_a y_{i,a} \quad (8)$$

In the above equation, s_i is the score of the i -th keyshot. l_i denotes the length of the keyshot, which is equal to the number of frames in the shot. $y_{i,a}$ is the importance score for the a -th frame. Each shot-level score indicates the importance of that particular keyshot.

To select the final keyshots, a maximization problem is formed. The sum of scores of the selected shots needs to be maximized while having a constraint on the combined duration of the selected keyshots. The previous problem is equivalent to the classic knapsack problem. The knapsack problem is a maximization problem where there are n different objects and each object has a cost and a weight. The problem is framed as follows: “How to put certain items in a knapsack capable of holding items up to a fixed weight limit so the cost is maximized?”. The keyshots generated using the KTS algorithm are treated as objects. The score associated with each keyshot corresponds to the cost of an object. The duration of each keyshot corresponds to the weight of an object. We want to restrict the duration to about 10% to 15% of the original video. For solving the knapsack problem, we use the dynamic programming-based approach.

4 Experimental findings

In this section, we first highlight the datasets and performance metrics used in the experiments. After that, we discuss the evaluation protocols and settings that are used in the present work. Finally, we present the experimental results and ablation studies, with suitable discussions on the observed results.

4.1 Dataset and performance metrics

4.1.1 Dataset description

We evaluated our proposed method on two popular benchmark datasets: the SumMe [21] and TVSum [22] datasets. The SumMe dataset comprises 25 video sequences from diverse backgrounds such as Jumps, Eiffel Tower, Car Rail-crossing and many more genres. Each video ranges from 0 m 39 s to 6 m 29 s in length. The TVSum dataset contains 50 video sequences (downloaded from YouTube¹) of 10 different categories such as making a sandwich, beekeeping, bike trips

¹ <https://www.youtube.com>.

Table 2 Comparisons with state-of-the-art approaches in terms of F-Score for augmented and transfer settings on SumMe and TVSum datasets

Method	Approach	SumMe			TVSum		
		Canonical	Augmented	Transfer	Canonical	Augmented	Transfer
SUM-GAN _{dpp} [6]	Unsupervised	0.391	0.434	-	0.517	0.595	-
DR-DSN [5]	Unsupervised	0.414	0.428	0.424	0.576	0.584	0.578
RSGN _{uns} [1]	Unsupervised	0.423	0.436	0.412	0.580	0.591	0.597
vsLSTM [9]	Supervised	0.376	0.416	0.407	0.542	0.579	0.569
dppLSTM [9]	Supervised	0.386	0.429	0.418	0.547	0.596	0.587
SUM-GAN _{sup} [6]	Supervised	0.417	0.436	-	0.563	0.612	-
A-AVS [14]	Supervised	0.439	0.446	-	0.594	0.608	-
M-AVS [14]	Supervised	0.444	0.461	-	0.610	0.618	-
DR-DSN _{sup} [5]	Supervised	0.421	0.439	0.426	0.581	0.598	0.589
HSA-RNN [26]	Supervised	0.423	0.421	-	0.587	0.598	-
RSGN [1]	Supervised	0.450	0.457	0.440	0.601	0.611	0.600
SUM-FCN [10]	Supervised	0.475	0.511	0.441	0.568	0.592	0.582
VASNet [12]	Supervised	0.497	0.511	-	0.614	0.624	-
DSNet _{AB} [17]	Supervised	0.502	0.507	0.465	0.621	0.639	0.594
DSNet _{AF} [17]	Supervised	0.512	0.533	0.476	0.619	0.622	0.580
Ours	Supervised	0.560	0.607	0.482	0.644	0.648	0.633

The scores in bold indicate the best scores

Table 3 Effect of addition of the weight module for both SumMe and TVSum datasets

Dataset	Weights	F1 Score
SumMe	No	0.470
SumMe	Yes	0.560
TVSum	No	0.607
TVSum	Yes	0.644

Reported results are for the canonical setting

and others. The SumMe and TVSum datasets contain frame-level importance scores annotated by 15-20 individual users. Further, to augment the datasets, we use two more datasets: OVP [23] and YouTube [23]. The OVP and YouTube datasets contain 50 and 39 videos, respectively. The OVP and YouTube datasets contain keyframes annotated by five users.

4.1.2 Performance metrics

We use the F_β measure to assess the performance of the generated summaries. We calculate the precision (P), recall (R) and F_β values according to Eqs. 9, 10 and 11, respectively.

$$P = \frac{\text{Length}(S_O \cap S_G)}{\text{Length}(S_O)} \quad (9)$$

$$R = \frac{\text{Length}(S_O \cap S_G)}{\text{Length}(S_G)} \quad (10)$$

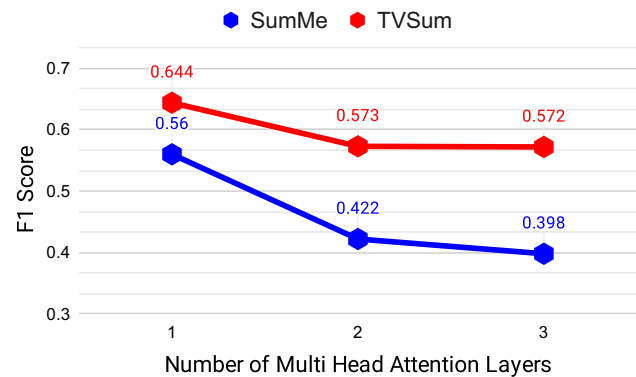
$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R} \quad (11)$$

In the above equations, S_O is the generated summary and S_G is the ground truth summary. We calculate the F_β score (where β equals 1) to measure the similarity between the generated summaries and the multiple ground-truth summaries for each video. The F_1 score is theoretically the harmonic mean of precision and recall.

Table 4 Effect of addition of Gaussian Noise for both SumMe and TVSum datasets

Dataset	Noise	F1 Score
SumMe	No	0.385
SumMe	Yes	0.560
TVSum	No	0.576
TVSum	Yes	0.644

Reported results are for the canonical setting

Fig. 2 Effect on performance for different number of multi-head attention layers for both SumMe and TVSum datasets. Reported results are for the canonical setting**Table 5** Difference while evaluating with different numbers of heads for the multi-head attention layers for both SumMe and TVSum datasets

Dataset	No. of Heads	F1 Score
SumMe	4	0.368
SumMe	8	0.560
SumMe	16	0.432
TVSum	4	0.533
TVSum	8	0.644
TVSum	16	0.543

Reported results are for the canonical setting

Following some past methods [9, 12], we evaluate the SumMe dataset using the ‘maximum’ setting. The reported metrics for the SumMe dataset are the maximum of all the samples used for testing purposes. On the other hand, for the TVSum dataset, the evaluation is based on an ‘average’ setting. This means that the reported metrics for the TVSum dataset are the average of all the samples used for testing purposes.

4.2 Evaluation protocols and settings

To deal with temporal redundancy among the frames, we downsample the original videos to 2fps from the original frame rate of 30fps. In addition, we summarize the videos to 15% of their actual length. To evaluate the effectiveness of our method we train and test the model using three different settings highlighted below.

- **Canonical:** We used one single dataset for both training and evaluation. 80% of the data is used to train the model and the remaining 20% data is used to test the performance of the model. The metrics reported in this case are the mean of the 5-fold cross-validation scheme.
- **Augmented:** We used All four datasets in this case. One primary dataset is selected from which 20% of the data is used to test the model performance. The remaining 80% of the data, along with the other three datasets, are used for training. The metrics reported in this case are the mean of the 5-fold cross-validation scheme.
- **Transfer:** Three out of four datasets are used for training, and we used the remaining dataset for testing. The metrics reported in this case are the mean of 5 separate runs.

Table 6 Difference while evaluating with varied loss functions for both SumMe and TVSum datasets

Dataset	Loss	F1 Score
SumMe	KLD	0.407
SumMe	MSE	0.554
SumMe	MSE & KLD	0.560
TVSum	KLD	0.571
TVSum	MSE	0.620
TVSum	MSE & KLD	0.644

Reported results are for the canonical setting

4.3 Implementation setup

For implementation, a machine with Nvidia Tesla T4 GPU is used. The Python 3.8 programming environment is used and the PyTorch library² is used for implementing the network. The loss function used can be found in Eq. 4. The AdamW [24] optimizer with a learning rate of 5×10^{-5} is used for training. The dropout probability is set to 0.5 and the model is trained for 200 epochs. For the weight assignment mechanism, the value of W is set as 3 and the value of R is set as 0.6

4.4 Results

4.4.1 Results in the canonical setting

In this section, we compare the proposed WANet with some recently published works in literature. We consider unsupervised methods [1, 5, 6, 8] as well as supervised methods [1, 5, 6, 9, 12, 14, 17, 25, 26]. Table 1 gives an overview w.r.t to recent methods results for canonical setting.

From the results shown in Table 1, it is clear that the proposed method outperforms the existing methods by a considerable margin. Also, it can be seen that the number of parameters is less than that of some recent works. We can attribute this to the fact that the model architecture is much simpler as compared to the other architectures. Therefore, it can be concluded that the model learns fewer redundant features. This finally leads to a better aggregation of the features within the fully connected layers. We can also observe that the performance in the TVSum dataset is higher than that in the SumMe dataset for every instance. This difference can be explained by the fact that the length and number of videos in the TVSum dataset are much higher than that in the SumMe dataset. The greater number of samples and the greater number of frames in the TVSum dataset helps to generalize the model. In addition, the TVSum dataset has 50 videos from 10 different categories (5 videos from each of the categories) whereas the SumMe dataset contains 25 videos all from different categories. Therefore, the size of the dataset is greater for the TVSum dataset.

4.4.2 Results in the augmented and transfer settings

To further evaluate the robustness and practicality of our model, we train and test the same in two challenging configurations: the Augmented and Transfer protocols. Table 2 provides detailed comparison of several existing models with our proposed model for both of these settings.

We can observe from Table 2 that our model outperforms state-of-the-art models by a considerable margin. We can also observe that, for both of the datasets, the augmented setting gives the maximum performance when compared to the canonical and transfer settings. Moreover, it is noted that the results considerably differ when any of the two settings are compared. While comparing the canonical and augmented settings, it must be considered that the number of samples used for training in the augmented setting is much higher than in the canonical setting. Thus the model is much more generalized in the augmented setting. Furthermore, in the case of the transfer setting, the performance is lower than that of both the canonical and augmented settings. This can be explained by the fact that one dataset is left out for testing the remaining three datasets are used for training. This is the reason why the

² <https://pytorch.org>.

Algorithm 1 Weight assignment mechanism

Input: N = Number of frame level features, W = Constant weight and R = Percentage of frames to be weighted
Output: Weight array of N frames.
Initialization :
1: $\text{weights} \leftarrow \text{ARRAY OF ONES}[N]$
2: $\text{record} \leftarrow \text{ARRAY}[N - 1][3]$
3: **for** $i \leftarrow 1$ to $N - 1$ **do**
4: $\text{cosine_sim} \leftarrow$ Calculate cosine similarity between i^{th} and $(i + 1)^{\text{th}}$ frames following Equation 1.
5: $\text{record}[i] \leftarrow (\text{cosine_sim}, i, i + 1)$
6: **end for**
7: $\text{record} \leftarrow$ Sort array in decreasing order according to scores in $\text{record}[[1]$
8: **for** $j \leftarrow 1$ to $N - 1$ **do**
9: $f1 \leftarrow \text{record}[j][2]$
10: $f2 \leftarrow \text{record}[j][3]$
11: **if** $j \leq (N \times (R/100))$ **then**
12: $\text{weight}[f1] \leftarrow \text{weight}[f1] * W$
13: $\text{weight}[f2] \leftarrow \text{weight}[f2] * W$
14: **else**
15: Pass
16: **end if**
17: **end for**
18: **return** weights

transfer setting is considered to be the toughest configuration for evaluation. There is a probability that the model might overfit, which can be better judged while evaluating on transfer setting.

4.5 Ablation study

4.5.1 Weights

Table 3 shows how the proposed method performs with and without weights assigned to the frames. We observe that the presence of weights has a significant impact on the performance of the model. This shows that the addition of weights to the selected frames makes the features more meaningful. We give some significant frames a greater priority, which makes the raw frame features more meaningful. This means we give some selected frames more prominence than others. Interestingly, it was experimentally observed that giving weights to the frame-level features also helps the model to converge relatively faster.

4.5.2 Importance of the addition of noise

In the present work, we extract frame-level features from a popular pre-trained CNN model: GoogleNet. CNNs are often referred to as discriminative architectures [27]. This enables CNNs to deal successfully with very large-scale datasets like the ImageNet dataset. However as stated earlier, the datasets used in this work are relatively smaller in size. The addition of noise is a type of augmentation which helps to improve the utility and robustness of the model. Hence, the use of noise is justified in our case to prevent overfitting. Such an application of noise is very common in the case of deep learners when dealing with relatively small datasets.

Table 4 gives a brief comparative overview of how the model performs both in the presence and absence of noise. This behaviour can be explained by the fact that the model overfits with a small amount of training data. The effect is much more consequential for the SumMe dataset as compared to the TVSum dataset because the SumMe dataset is smaller. In addition, the duration of videos in the SumMe dataset is relatively much smaller, leading to aggravating the model performance even more.

4.5.3 Multi-Head Attention

From Table 5 it can be observed the method performs at its optimum while using 8 heads. A single head in a multi-head attention layer focuses on a different part of the input. In an investigation conducted by Michel et al. [28] the authors showed that in many cases it is possible to remove a majority of attention heads without affecting the performance by a large margin. Following the results of the study, in the present case, it is observed that increasing the number of heads leads to a decrease in performance possibly due to a redundancy in the features learnt by different heads.

Additionally, the study conducted in the above experiment suggests that using less than eight heads reduces the learning capability of the model. This can also explain the decrease in performance in the present case with four heads.

We also observe in Fig. 2 that using one multi-head attention layer yields better results, possibly due to overfitting caused by increasing the number of layers.

4.5.4 Loss

MSE loss is very often used in regression-based tasks in the domain of DL. Whereas KLD loss calculates how much the obtained probability distribution differs from the expected probability distribution. A major difference between these two loss values is that, unlike MSE loss, KLD loss is not symmetrical in nature. In our case, we propose to use a hybrid of these loss values to get the best of these two. Results tabulated in Table 6 confirm our claim.

5 Conclusion

In the past few years, storing of videos has increased a lot. To handle this big load of video data, video summarization is now used to make shorter summaries of the whole video content. The main aim of video summarization is to squash all the details in the video frames into a smaller summary. In the present work, we have proposed a model, called WANet, which is a supervised video summarization technique. The deep network comprises a multi-head attention mechanism followed by feed-forward networks to obtain the frame-level probability scores. The deep network is fed with frame-level features extracted from pre-trained GoogleNet (pre-trained on the ImageNet dataset). Summarized keyshots are obtained using a combination of KTS and the knapsack algorithm. Contrary to the existing methods, we first enrich the input frames with similar information as opposed to letting the model learn all the features by itself. A novel weight assignment mechanism is used to assign weights based on similarity to the input frames based on their similarity before passing the frames to the model. Experimental results on two benchmark datasets: SumMe and TVSum indicate the effectiveness of the present method when compared to state-of-the-art methods in video summarization. Ablation studies also confirm the major role of the weight assignment scheme in boosting the performance of the model. The main limitation in our approach is that the weight assignment mechanism is a static process. Future work can explore the effect of using a dynamic or trainable weight assignment scheme to obtain improved performance. For example, the weights can be found using an auxiliary neural network that takes the frames as input. Another limitation of our method is the random initialization of weights for the model. In future, we can explore transfer learning to have good initial weights. In addition, changes to the model itself (ensembling [29, 30], use of recent architectures [31–33]) can also be explored to improve the performance.

Data availability Data shall be made available under reasonable request

Declarations

Competing interests The authors declare that they have no conflict of interest

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Zhao B, Li H, Lu X, Li X. Reconstructive sequence-graph network for video summarization. *IEEE Trans patt analysis mach intell*. 2021. <https://doi.org/10.1109/TPAMI.2021.3072117>.
2. Zhong R, Wang R, Zou Y, Hong Z, Hu M. Graph attention networks adjusted Bi-LSTM for video summarization. *IEEE Signal Proc Lett*. 2021;28:663–7.
3. Khosla A, Hamid R, Lin CJ, Sundaresan N. Large-scale video summarization using web-image priors. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*; 2013; p. 2698–2705.
4. Potapov D, Douze M, Harchaoui Z, Schmid C. Category-specific video summarization. In: *European conference on computer vision (ECCV)*. Springer; 2014; p. 540–555.
5. Zhou K, Qiao Y, Xiang T. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018; vol. 32.
6. Mahasseni B, Lam M, Todorovic S. Unsupervised video summarization with adversarial lstm networks. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*; 2017; p. 202–211.
7. Jung Y, Cho D, Kim D, Woo S, Kweon IS. Discriminative feature learning for unsupervised video summarization. *Proc AAAI Conf Artif Intell*. 2019;33:8537–44.
8. Apostolidis E, Adamantidou E, Metsai AI, Mezaris V, Patras I. Ac-sum-gan: connecting actor-critic and generative adversarial networks for unsupervised video summarization. *IEEE Trans Circuits Syst Video Technol*. 2020;31(8):3278–92.
9. Zhang K, Chao WL, Sha F, Grauman K. Video summarization with long short-term memory. In: *European conference on computer vision*. Springer; 2016; p. 766–782.
10. Rochan M, Ye L, Wang Y. Video summarization using fully convolutional sequence networks. In: *Proceedings of the European Conference on Computer Vision (ECCV)*; 2018; p. 347–363.
11. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: *Advances in neural information processing systems (NeurIPS)*; 2017; p. 5998–6008.
12. Fajtl J, Sokeh HS, Argyriou V, Monekosso D, Remagnino P. Summarizing videos with attention. In: *Asian Conference on Computer Vision (ACCV)*. Springer; 2018; p. 39–54.
13. Huang C, Wang H. A novel key-frames selection framework for comprehensive video summarization. *IEEE Trans Circ Syst Video Technol*. 2019;30(2):577–89.
14. Ji Z, Xiong K, Pang Y, Li X. Video summarization with attention-based encoder-decoder networks. *IEEE Trans Circ Systems Video Technol*. 2019;30(6):1709–17.
15. Bahdanau D, Cho K, Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. In: *3rd International Conference on Learning Representations, ICLR 2015*; 2015.
16. Luong T, Pham H, Manning CD. Effective Approaches to Attention-based Neural Machine Translation. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics; 2015; p. 1412–1421.
17. Zhu W, Lu J, Li J, Zhou J. DSNet: a flexible detect-to-summarize network for video summarization. *IEEE Trans Image Proc*. 2020;30:948–62.
18. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*; 2015; p. 1–9.
19. Shorten C, Khoshgoftaar TM. A survey on image data augmentation for deep learning. *J Big Data*. 2019;6(1):1–48.
20. Cilingir HK, Manzelli R, Kulis B. Deep Divergence Learning. In: *Proceedings of the 37th International Conference on Machine Learning*. vol. 119 of *Proceedings of Machine Learning Research*. PMLR; 2020; p. 2027–2037.
21. Gygli M, Grabner H, Riemenschneider H, Van Gool L. Creating summaries from user videos. *Comp Vision ECCV*. 2014;2014:505–20.
22. Song Y, Vallmitjana J, Stent A, Jaimes A. TVSum: Summarizing Web Videos Using Titles. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*; 2015.
23. De Avila SEF, Lopes APB, da Luz Jr A, de Albuquerque Araújo A. VSUMM: a mechanism designed to produce static video summaries and a novel evaluation method. *Patt Recognit Lett*. 2011;32(1):56–68.
24. Loshchilov I, Hutter F. Decoupled weight decay regularization. *arXiv*. 2017. <https://doi.org/10.48550/arXiv.1711.05101>.
25. Huang S, Li X, Zhang Z, Wu F, Han J. User-ranking video summarization with multi-stage spatio-temporal representation. *IEEE Trans Image Proc*. 2018;28(6):2654–64.
26. Zhao B, Li X, Lu X. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*; 2018; p. 7405–7414.
27. Arel I, Rose DC, Karnowski TP. Deep machine learning - a new frontier in artificial intelligence research [Research Frontier]. *IEEE Comput Intell Magaz*. 2010;5(4):13–8.
28. Michel P, Levy O, Neubig G. are sixteen heads really better than one? *Adv Neural Inform Proc Syst (NeurIPS)*. 2019;32:14014–24.
29. Altameem A, Mahanty C, Poonia RC, Saudagar AKJ, Kumar R. Breast cancer detection in mammography images using deep convolutional neural networks and fuzzy ensemble modeling techniques. *Diagnostics*. 2022;12(8):1812.
30. Mahanty C, Kumar R, Asteris PG, Gandomi AH. COVID-19 patient detection based on fusion of transfer learning and fuzzy ensemble models using CXR images. *Appl Sci*. 2021;11(23):11423.
31. Mijwil MM, Gök M, Doshi R, Hiran KK, Kösesoy I. Utilizing Artificial Intelligence Techniques to Improve the Performance of Wireless Nodes. In: *Applications of Artificial Intelligence in Wireless Communication Systems*. IGI Global; 2023; p. 150–162.
32. Mahanty C, Kumar R, Patro SGK. Internet of medical things-based COVID-19 detection in CT images fused with fuzzy ensemble and transfer learning models. *New Gener Comput*. 2022;40(4):1125–41.
33. Mahanty C, Kumar R, Mishra BK, Barna C. COVID-19 detection with X-ray images by using transfer learning. *J Intell Fuzzy Syst*. 2022;43(2):1717–26.