V-Trans4Style: Visual Transition Recommendation for Video Production Style Adaptation

Pooja Guhan[1,2], Tsung-Wei Huang[2], Guan-Ming Su[2], Subhadra Gopalakrishnan[2], and Dinesh Manocha[1]

[1] University of Maryland, College Park MD 20742, USA
[2] Dolby Laboratories, Sunnyvale CA 94085, USA

Abstract. We introduce V-Trans4Style, an innovative algorithm tailored for dynamic video content editing needs. It is designed to adapt videos to different production styles like documentaries, dramas, feature films, or a specific YouTube channel's video-making technique. Our algorithm recommends optimal visual transitions to help achieve this flexibility using a more bottom-up approach. We first employ a transformer-based encoder-decoder network to learn recommending temporally consistent and visually seamless sequences of visual transitions using only the input videos. We then introduce a style conditioning module that leverages this model to iteratively adjust the visual transitions obtained from the decoder through activation maximization. We demonstrate the efficacy of our method through experiments conducted on our newly introduced AutoTransition++ dataset. It is a 6k video version of AutoTransition Dataset that additionally categorizes its videos into different production style categories. Our encoder-decoder model outperforms the state-of-the-art transition recommendation method, achieving improvements of 10% to 80% in Recall@K and mean rank values over baseline. Our style conditioning module results in visual transitions that improve the capture of the desired video production style characteristics by an average of around 12% in comparison to other methods when measured with similarity metrics. We hope that our work serves as a foundation for exploring and understanding video production styles further. [Project Website]

Keywords: video transitions · video style · transition recommendation · video editing

1 Introduction

With the growing popularity and consumption of digital content, creating high-quality videos still remains a time-consuming, expensive, and highly creative task. To meet the ever-growing demand for diverse and captivating multimedia experiences, the strategic act of *adapting* videos into various production styles emerges as an invaluable resource. *Video production styles* encompass distinct approaches and techniques employed in crafting videos. These styles aim to convey

specific messages, elicit emotions, or achieve particular aesthetics. While there is no finite set of video styles, they can be attributed to genre-specific approaches, such as those used in documentaries, feature films, news/reportage, drama, and vlogs. Additionally, video styles can be specific to famous directors and content creators, or they may reflect an organization's distinctive marketing styles [2]. These video styles can be achieved with a mix of multiple visual elements, including camera angles, scene organization, and visual transitions [25]. Adapting videos into these varied styles can help content creators create videos for diverse audience preferences and consistently deliver captivating content that stands out in the competitive landscape of digital media. However, editing videos to specific styles remains a laborious task. In this work, we limit our scope of understanding the dependence of video styles on visual transitions. *Visual transitions* serve

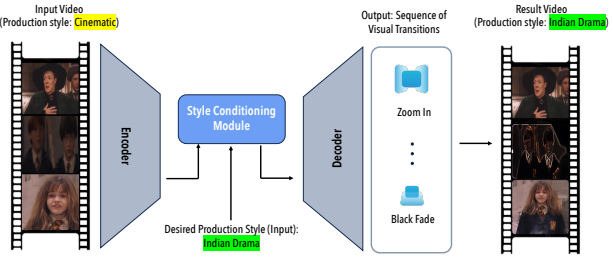


Fig. 1: Our goal is to recommend the optimal visual transition sequence for enabling the adaptation of a given video to any desired production style. We propose V-Trans4Style, a novel bottom-up approach consisting of an encoder-decoder architecture and a style conditioning module.

as the creative glue during post-production, seamlessly connecting video clips and enhancing the video's flow and style. These effects enable smooth shifts between shots, often introducing a visual and sometimes thematic link between them [1]. Fades, dissolves, wipes, cross-fades, etc are among the various types of transitions, and they have a significant impact on the overall look and feel of the video [30]. Today, state-of-the-art video editing softwares [5] offer capabilities for implementing these transitions in videos. For non-professionals lacking expertise in cinematography and design, the challenge lies in selecting transitions from numerous possibilities and exploring various combinations to adapt videos for different production styles. Even seasoned editors grapple with its time-consuming nature. Video templates, i.e., pre-designed and pre-structured video layouts, available today are limited, rigid, and are not tailored for the user's video content on which it is being applied. This thereby, reduces the overall quality of the adapted video. Therefore, *we introduce an automated method of recommending transitions for adapting videos for different video production styles*. To attain our objective, we must navigate through certain challenges. Firstly, *no* publicly available dataset provides the different sets of transitions

that need to be applied to the same video to adapt it for different styles. Developing such a dataset is also an extremely tedious task. Additionally, unlike measurable aspects of video production such as frame rate or resolution, video style is abstract and lacks concise formulation. Apart from the desired video style, the sequence of transitions being predicted for adapting a video should also match the dynamics and rhythm of the video content as much as possible to allow for video continuity.

Main Contributions: To overcome these challenges, we present V-Trans4Style. Targeting video editing applications, our goal is to provide content creators with a user-friendly approach that can enable them to adapt their videos to different production styles, ultimately enhancing engagement and broadening audience appeal. Our main contributions can be summarized as follows:

1. We present the first algorithm that facilitates video adaptation to various styles during video editing. The input is a video comprised of an ordered set of video clips and the name of the desired video production style. The output is the recommended sequence of visual transition classes that the user should use to adapt the input video to the desired video production style.
2. We present a novel method that follows a bottom-up approach to recommend transitions suitable for video production style adaptation. We first train a transformer encoder-decoder architecture to learn to recommend temporally and visually consistent visual transitions. We then use our style conditioning module to perform a controlled update of its latent embedding during inference time via activation maximization. This leads to the introduction of video production style-based characteristics in the sequence of transitions obtained from the decoder.
3. We create AutoTransition++, a mini representative dataset of AutoTransition [23] consisting of $6k$ videos. Additionally, in a bid to fortify our research foundation, we also include additional annotations to categorize the videos within it into five distinct video production styles.

We show quantitative evaluations on AutoTransition++. Our encoder-decoder model shows an improvement of at least 10% in Recall@K and mean rank metrics as compared to the baseline. Additionally, we show that our style conditioning module can help better capture the video production style characteristics by 12% as compared to baselines.

2 Related Works

Video Editing. The recent surge in the popularity of video editing is largely attributed to advancements in generative AI technologies. Within this field, four main categories of work have emerged. The first category focuses on content manipulation, where existing visual elements are modified on a frame-by-frame basis. This empowers users to substitute specific objects with their preferred choices [8, 15, 20]. Another aspect involves altering the visual appearance of frames, such as transforming them into sketches or anime, without removing or

replacing any objects [26]. This closely aligns with existing literature on image style transfer in terms of their end objectives. Text-driven methods for modifying visual content have also gained considerable popularity recently. The third category encompasses tasks related to video summarization and highlight detection. In this context, the goal is to pinpoint specific frames in the video that satisfy particular queries [17, 27, 29] directly linked to the video's content. Finally, the fourth category pertains to the fundamental elements of video editing, including cuts [4,19], transitions [23,24], and camera placements [21]. Additionally, there is a wealth of research focused on video frame interpolation [12, 18] and enhancing the overall rendering quality of videos [9].

Visual Transitions in Videos. Visual transitions in video editing play a crucial role in maintaining continuity, influencing the perception of time, and amplifying the video's ability to convey specific moods and emotions. Previous research has primarily focused on identifying transitions in videos [16] and understanding their impact on the viewing experience [28]. Among the existing literature, the work closest to our objectives is [23], where they introduce a visual transition recommendation system. While both their work and ours address the problem of recommending visual transitions, *our focus is specifically on ensuring coherence and conformity of the recommended transitions with the intended production style characteristics of the video.*

Video Editing Datasets. Current video editing-related datasets are primarily based on classifying video shots [3, 10], visual scene recognition, detecting and identifying cuts in videos [19], multi-camera editing [21], detecting transitions and recommending transitions [23]. However, there are no existing datasets, that also give information regarding what video style category these fall under. The lack of this information has made the task of solving problems related to how style associates with the different video editing non-trivial. We, therefore, extend the Autotransition dataset [23] to incorporate the video style-related labels.

Despite these advancements, there is a notable absence of work exploring the distinct styles of video production and their unique compositions. Our work aims to bridge this gap by examining the intersection of video editing and production style, contributing to a deeper understanding of their unique characteristics. Moreover, *while previous works have primarily focused on manipulating the original visual content, we instead only concentrate on altering the manner in which different video clips are stitched together to compose the video.*

3 Task Definition

A video V comprises of ordered video clips represented by $V = \{c_1, c_2, ..., c_n\}$, where c_i, for $i \in \{1, 2, ..., n\}$, represents the i^{th} video clip out of a total of n video clips in V. Now, consider the task of adapting video V to adapt to a different production style s. Our objective here is to estimate the sequence of visual transitions $T = \{tr_1, tr_2, ..., tr_{n-1}\}$ that will facilitate the transformation of the video for the given desired video style s. Here, tr_k for $k \in \{1, ..., n-1\}$ is the

transition being recommended between video clips c_k and c_{k+1} in the adapted video. T can be an altered version of the transitions that V might have had before production style adaptation or an entirely new sequence. In our method, we take V and s as input and generate T as the output.

4 Dataset

4.1 AutoTransition Dataset

This dataset [23] primarily consists of 35k videos with each video annotated for the set of transitions that have occurred. They provide information regarding the class of transition, the timestamp of when it begins, when it ends, duration, and type of transition. Overall, the dataset has 93 transitions. However, most of the dataset transitions are spread across 30 different visual transitions. Some of these transitions include pull-in, mix, dissolve, and black fade.

AutoTransition++ Dataset. We release a mini representative dataset of the AutoTransition dataset comprising $6k$ videos to handle computational constraints. The transition distribution within AutoTransition++ mirrors that of the AutoTransition dataset. Also, in our endeavor to enable the seamless adaptation of videos across various styles using transitions, autotransition presents two significant limitations. Firstly, it *only* provides a *single set* of transition annotations for each video, leaving a gap in the availability of transition information tailored to diverse video styles. This absence restricts its direct applicability for adaptation across different style variations. Secondly, the dataset does not disclose the video style for each entry, impeding our ability to leverage this crucial information for more effective and nuanced video adaptation.

A careful manual examination of the production styles of the videos within the dataset resulted in categorizing the videos into five main categories: vlog, influencer-centric, nature/urban-related scenes, photo slideshows, and anime. The videos in these categories differed not only in terms of content but also in the distribution of transitions. Therefore, the categorization was found reasonable to be considered as five different video styles available in the dataset. We enlisted the expertise of 8 seasoned annotators who possess extensive experience in evaluating a diverse range of visual content. They were initially made to undergo a training phase where they evaluated our style categorization using a curated selection of videos spanning various styles. This not only refined their understanding of the task but also ensured uniformity and comprehension of the distinct style labels. 1379 samples from the AutoTransition++ dataset were annotated with video-style labels. The distribution of visual transitions observed in the annotated samples for the different video styles has been plotted in Fig. 2. Fig. 2 also indicates that the different styles may incorporate common transitions, potentially blurring the lines that distinguish one style from another. The discerning factor between different video styles need not be only the presence of a few unique transitions but also the frequency with which specific transitions are employed in one style as compared to the other. This behavior is observed

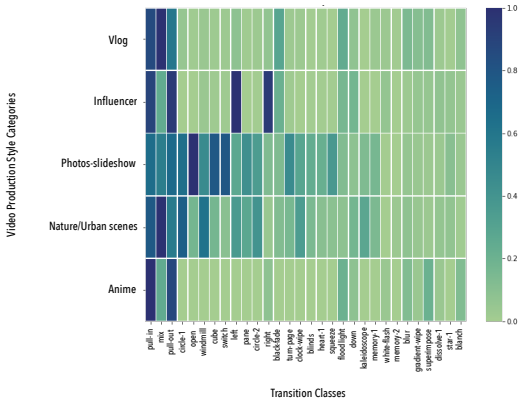


Fig. 2: Bivariate distribution observed between the different styles and the visual transitions deployed across the 1379 video production style annotated videos within Auto-Transition++.

not only among the samples annotated by us but also in video style analysis shown by [23]. However, [23] have not released the video style data.

5 V-Trans4Style

Our overarching goal is to recommend visual transitions to facilitate the adaptation of a video to a different production style. The top transitions obtained from our algorithm should ideally satisfy *three* conditions. They should (a) *match the video content and dynamics* (b) *ensure video continuity* and (c) *exhibit the desired production style characteristics*. To this end, we opt for a bottom-up approach. Our algorithm consists of *three* main components: an encoder ($\mathcal{E}$), a decoder ($\mathcal{D}$), and a style conditioning module ($\mathcal{SCM}$). Our encoder-decoder is neural-network-based. Their combination is employed to learn a generative model of visual transitions solely from video clips. Given an input V (set of ordered video clips used in input video, Sec. 3), we feed it through the encoder to obtain its embedding vector z. z is then fed to the decoder. The decoder will then generate the corresponding sequence of $n-1$ transitions that *only satisfy* conditions (a) and (b). n is the number of video clips in V. The trained encoder-decoder network is subsequently used by the most crucial part of our algorithm, the style conditioning module. It uses a gradient-based optimization approach through activation maximization [6] to update z. This will favorably influence the decoder outputs (i.e., the sequence of $n-1$ visual transitions) and thereby, also satisfy condition (c).

5.1 Pre-trained Transition and Style Embeddings

This step serves as a pre-processing stage in the algorithm, aiming to derive meaningful representations for distinct visual transitions and video styles. As

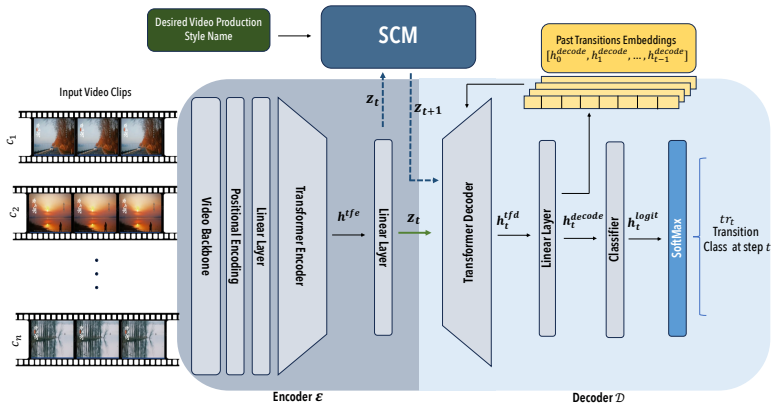


Fig. 3: V-Trans4Style: Ordered clips $\{c_1, c_2, .., c_n\}$ in V is fed to an Encoder $\mathcal{E}$ to obtain z_t. Decoder $\mathcal{D}$ takes in z_t and outputs a sequence of transitions in $n-1$ steps. At each step t, the $\mathcal{D}$ outputs tr_t. The masked transformed decoder uses past transition embeddings in every step to compute h_t^{tfd}. z_t is same (i.e., $z = z_t$) across all steps during the joint training of $\mathcal{E}$ and $\mathcal{D}$. Only components connected by $\rightarrow$ are active during training. $\mathcal{SCM}$ is an inference-time module. At inference, for every step t, $\mathcal{SCM}$ takes in current z_t to estimate the appropriate z_{t+1} for the next step to produce production style favorable transition tr_t.

detailed in [23], specific transitions share common visual effects, such as pull-in and pull-out. Additionally, as depicted in Fig. 2, certain sets of transitions are more prevalent in videos associated with one style compared to the other. It is anticipated that the learned embeddings corresponding to these transitions will capture these characteristics. To achieve this, we employ a multitask learning-based network, $\mathcal{MLN}$ (Fig. 4). The network optimizes on two tasks: (1) classifying visual transitions and (2) predicting the video style in which the transition will be employed. The network takes in only the video segment associated with a visual transition and obtains its visual features using a video backbone. These features are then linearly transformed and normalized to obtain a unit vector, $\mathcal{U}$. The unit vector $\mathcal{U}$ is then sent to two linear classifiers for transition class prediction and video style prediction respectively. The embeddings corresponding to each transition in the dataset and style are sampled from $\mathcal{U}$ (Refer to appendix for details). Therefore, this will give us the set of visual transition embeddings $E_{tr} = \{e_1^{tr}, e_2^{tr}, ..., e_{N_{tr}}^{tr}\}$ where e_i^{tr} for $i \in \{1, 2, ..., N_{tr}\}$ represents the embedding corresponding to the i^{th} transition class. N_{tr} denotes the total number of visual transition classes. With transition embeddings, we will also get the set of style embeddings, $E_{style} = \{e_1^{style}, e_2^{style}, ..., e_{N_{style}}^{style}\}$. Here, e_i^{style} for $i \in \{1, 2, ..., N_{style}\}$ refers to the embedding corresponding to the i^{th} style and N_{style} denotes the total number of video styles available.

5.2 Encoder and Decoder

Our algorithm strongly depends on the encoder $\mathcal{E}$ and decoder $\mathcal{D}$. Their collaborative operation to generate visual transitions using clips forms the core of V-Trans4Style. The transitions obtained using $\mathcal{E}$ and $\mathcal{D}$ should take into account not only the visual content but also the temporal dependency between transitions to ensure video continuity. Taking this into consideration, we employ a transformer encoder-decoder-based architecture. The input is a sequence of video clips in the order they appear in the input video. The output is a sequence of transitions *satisfying* transition conditions (a) and (b).

We begin by uniformly sampling w frames from each video clip in V. These are then fed into a video backbone following standard practices [23] to obtain an embedding e^V. In our experiments, we use SlowFast [7] as our video backbone. Before sending the visual features of the video clips to the transformer encoder, learnable position embeddings are element-wisely added to them. This is to encourage the model to consider the sequence of the video clips in the input video into consideration while recommending the sequence of transitions. The transformer encoder outputs h^{tfe} that is fed to a fully connected layer to obtain z. Therefore, $\mathcal{E}(V) = z$. To ensure video continuity, the decoder $\mathcal{D}$ is designed to recommend one transition at a time, relying on previously generated transitions. It uses a masked transformer decoder using a masked multi-head self-attention on the target transition sequence. The attention for each transition is restricted to cover only those preceding it in the sequence by using a triangular mask. This ensures that $\mathcal{D}$'s transition recommendation is causal and therefore, usable during inference time when the full target transition sequence is not available. For n video clips in V, $\mathcal{D}$ recommends transitions across $n-1$ steps. At each step t, z along with transitions observed till step $t-1$ (obtained after masking) fed to the transformer decoder outputs h_t^{tfd}. Subsequently, h_t^{decode} is fed into another linear layer to produce h_t^{logit}. To train the encoder-decoder part of the algorithm, we make use of the following loss function:

$$L_V = L_{classification} + \lambda L_{masked_triplet} \tag{1}$$

where $L_{classification}$ corresponds to the cross entropy loss computed between h_t^{logit} and y_t^{trans} (the ground truth transition classes for step t) across all time steps. λ is a scalar hyperparameter. $L_{masked_triplet}$ is the masked triplet loss computed as:

$$L_{masked_triplet} = \frac{1}{N_{tr}-1} \sum_{i \neq gt, i \in 1,\ldots,N_{tr}} \mathcal{T}(h_t^{decode}, e_{gt}^{tr}, e_i^{tr}) \tag{2}$$

where e_{gt}^{tr} represents the transition embedding corresponding to the ground truth transition class and e_i^{tr} represents the transition embedding corresponding to the i^{th} class of transitions, except ground truth class. $\mathcal{T}$ calculates the triplet margin loss for each triplet $(h_t^{decode}, e_{gt}^{tr}, e_i^{tr})$

$$\mathcal{T}(a, p, n) = max(\phi(a, p) - \phi(a, n) + m, 0) \tag{3}$$

m is the soft margin, a, p, n are anchor, positive sample and negative samples respectively. N_{tr} is the total number of visual transition classes available. $\phi(x, y)$ denotes the similarity metric between x and y. In this case, it is taken to be the dot product between the two. We can define ϕ therefore as,

$$\phi(x, y) =< x, y > \tag{4}$$

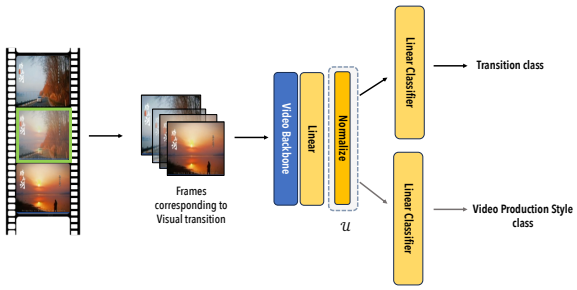


Fig. 4: $\mathcal{MLN}$: The green box shows the region where the visual transition occurred in the video. The frames corresponding to this transition are fed to a video backbone and then through a linear layer to obtain $\mathcal{U}$ after normalization. This is then fed to linear classifiers for predicting the transition class and the video style.

5.3 Style Conditioning Module

$\mathcal{E}$-$\mathcal{D}$ network construction and training objective will only enable attainment of visually and temporally coherent transitions. Incorporating video production style information within the $\mathcal{E}$-$\mathcal{D}$ network itself can be complex due to the absence of task-appropriate datasets (as discussed in Sec. 1 and 4). Therefore, we propose a *style conditioning module*($\mathcal{SCM}$) to alter (if needed) the transition choices made by the decoder above to align them with the requirements of the desired video style. It is designed to eliminate the need to train $\mathcal{E}$-$\mathcal{D}$ for each style separately or have access to transition annotations for each style for each video sample. $\mathcal{SCM}$ is activated after training the encoder-decoder network during inference time. Its operational principle relies on the fundamental recognition that the outputs derived from the decoder are driven by z. Therefore, this implies that any alteration to z will change the decoder outputs. We leverage this simple yet crucial property to achieve the purpose of this module in V-Trans4Style. We implement the style conditioning module by employing a gradient-based optimization strategy, guided by *activation maximization(AM)* [6] to update z. The update is designed to ensure that the transition sequence generated by the decoder $\mathcal{D}$ exhibits the desired production style characteristics. We execute $\mathcal{SCM}$ at every decoding step during inference. Therefore, let z_t be the input to $\mathcal{D}$ at step t. In our case, AM refers to the process of using backpropagation through

the decoder weights to identify a permutation of z_t to bring about a desired change in the output of the decoder at step $t+1$. This therefore requires us to establish one or more distinct differentiable loss functions that can guide the optimization of z_t. These loss functions are equivalent to setting constraints that the updated z_t should try to satisfy. In our case, we have two loss functions.
Embedding Loss ($\mathcal{L}_\mathcal{E}$)**.** This ensures that z_t is updated in a direction that results in the decoder generating a transition sequence that facilitates the adaptation of the input video to the intended style.

$$\mathcal{L}_\mathcal{E} = 1 - \sigma_{cos}(e_\mu, e_k^{style}) \tag{5}$$

where $\sigma_{cos}(a, b)$ refers to the cosine similarity computed between two d-dimensional vectors a, b $\in R^d$, i.e.,

$$\sigma_{cos}(a, b) = \frac{< a, b >}{||a||_2.||b||_2} \tag{6}$$

Additionally e_k^{style} is the embedding corresponding to the k^{th} video style among N_{style} video styles (Refer Sec. 5.1) and we define e_μ as the average of embeddings corresponding to transitions that have been observed till decoder step t for the video.

$$e_\mu = \sum_{i=1}^{t} h_i^{decode} \tag{7}$$

Reconstruction Loss($\mathcal{L}_\mathcal{R}$)**.** This ensures that there isn't a significant deviation in the values of z_t that can compromise the integrity of the information derived from the input video. To compute this, we trained a decoder $\mathcal{D}_\psi$ that takes in z_t to retrieve back h^{tfe}. Referring to Sec. 5.2, h^{tfe} was obtained after encoding the input and z_t after processing it further. Therefore, even after the alteration of z_t, we can be sure that the input information has still been preserved in z if we can reliably reconstruct h^{tfe}. More details about $\mathcal{D}_\psi$ can be found in the appendix. Essentially, let

$$\mathcal{D}_\psi(z_t) = \hat{h}^{tfe} \tag{8}$$

Therefore, the reconstruction loss $\mathcal{L}_\mathcal{R}$ can be computed as

$$\mathcal{L}_\mathcal{R}(z) = |\hat{h}^{tfe} - h^{tfe}|_1 \tag{9}$$

h^{tfe} is not updated in this process. Therefore $\forall$ decoding step $t = 1$ to $n-1$, z_t will be optimized to reconstruct the same h^{tfe}. The overall loss function used for the optimization of z_t

$$\mathcal{L}(z_t) = \alpha_\mathcal{E}\mathcal{L}_\mathcal{E} + \alpha_\mathcal{R}\mathcal{L}_\mathcal{R} \tag{10}$$

where $\alpha_\mathcal{E}$ and $\alpha_\mathcal{R}$ are the step sizes corresponding to $\mathcal{L}_\mathcal{E}$ and $\mathcal{L}_\mathcal{R}$ respectively. z_k is initially assigned to be equal to z_t. The following optimization process is then run for 5000 iterations with a learning rate β to finally obtain z_{t+1} (Ref. Fig 3).:

$$z_{k+1} \leftarrow z_k - \beta\nabla\mathcal{L}(z_k) \tag{11}$$

Finetuning the transition sequence: After $n-1$ decoder steps, we obtain the preliminary sequence of transitions T that can enable video style adaptation. Optionally, we can finetune this sequence further if needed without affecting any parameters or disrupting any required conditions drastically. We take inspiration from RRT [13], a path-planning algorithm to do so. We consider the sequence of h_t^{decode} for $t \in \{1, 2, ...n-1\}$ that was generated in the process to obtain tr_t. By the training construction of the decoder, tr_t corresponds to the transition class whose pre-trained embedding is closest to h_t^{decode}. For RRT instead, we consider its K nearest pre-trained embeddings using a distance metric. K value should not be too high or too low. A high K value can change T more than we need and a very low K might not make much difference to T. K denotes the extent of exploration we wish to do. Each of the K transitions is then iteratively evaluated to compute e_μ that is closest to the desired video style embedding, i.e., e_k^{style}.

6 Experiments and Results

We first discuss implementation details, the metrics used, and baselines compared with in Sec. 6.1. Our algorithm is a bottom-up approach to obtain visual transitions that can facilitate the adaptation of a video to a different production style. Each component plays a crucial role in building the entire algorithm. Sec 6.2 analyzes each of their performance in detail.

6.1 Implementation Details

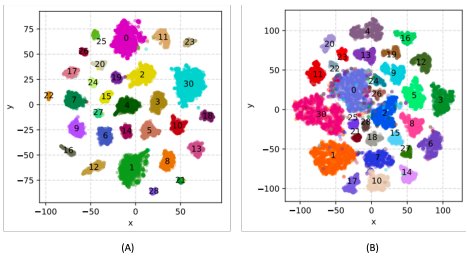
Model Details. Similar to AutoTransition [23], we have employed SlowFast 8x8 as the video backbone to extract visual features. This is used in the encoder component as well as for both obtaining pre-trained transition and style embeddings. $\mathcal{E}$ and $\mathcal{D}$ each use 2 transformer layers for encoding and decoding with $d_{model} = 512$ and $n_{head} = 8$.

Data Pre-processing and Training Details. The maximum number of ordered clips considered in a video is user defined. For experiment purposes, we have considered it as 8. For both, obtaining pre-trained transition embeddings and transition sequence recommendations, we sample 16 frames with an image size of 224 x 224 from each video clip in V. The former uses a batch size of 64 while the latter uses 16. Both training processes are run for a total of 60 epochs. The model parameters of the last epoch are used to obtain the embeddings corresponding to the $N_{tr} = 30$ transition classes and $N_{style} = 5$ video production style classes from $\mathcal{MLN}$(Fig. 4). We train all our networks using the Adam optimizer [14] with a learning rate 1e-5. The margin m in triplet loss is set to 0.5. We run our experiments on the AutoTransition++ dataset. For training $\mathcal{E}$ and $\mathcal{D}$, we use 80% of the data for training, validate the performance on 10% of the data, and test on the remaining 10% of the data. We train all models on 8 A5000 GPUs. In $\mathcal{SCM}$, we use an SGD optimizer [22] for optimization and a learning rate of 0.1, $\alpha_\mathcal{E}$ and $\alpha_\mathcal{R} = 1$. All codes were implemented using Pytorch [11].

Metrics and Baselines. For a fair evaluation of *only* our encoder-decoder model and pre-trained embeddings, we consider the metrics used by the only other relevant baseline, AutoTransition. These include Recall@K where K$\in \{1, 5\}$ and Mean Rank. The evaluation is done for every transition in the sequence. For $\mathcal{SCM}$, we consider cosine similarity, which informs us about how relevant a method's recommended transitions are for the video production style we wish to achieve.

Transition Embedding	**Evaluation Metrics** Recall@1↑	Recall@5↑
Transition Classifier [23]	97.85	99.78
$\mathcal{MLN}$ Task-1	96.67	98.77
$\mathcal{MLN}$ Task-2	93.45	99.54

(a)



(b)

Fig. 5: (a) Compares transition classification results between [23]'s classifier and MLN. MLN Task-1 is for Transition classification, and Task-2 is for video production style classification. **(b)** Displays t-SNE visualizations of the transition embeddings. Fig. (A) depicts embeddings from [23]'s method, while (B) represents embeddings from MLN.

6.2 Comparisons and Analysis

Pre-Trained Transition and Style Embeddings. Fig.5 provides insights into the quality of pre-trained transition embeddings used for training $\mathcal{E}$ and $\mathcal{D}$, and style embeddings for $\mathcal{SCM}$. $\mathcal{MLN}$ differs from [23]'s classifier in its ability to obtain transition embeddings that obeyed both visual and style similarities. Given the relatively smaller number of constraints, it is expected that the transition embeddings in the case of [23] will be comparatively more distinctly placed (as observed in Fig.5b) as compared to that obtained using $\mathcal{MLN}$. This comparison aims to highlight that the additional style constraint in $\mathcal{MLN}$ doesn't significantly compromise transition classification accuracy while maintaining high video style classification. Therefore, the differences between different transition and style classes have been captured by $\mathcal{MLN}$ reliably.

Encoder-Decoder. In this work, we have limited our scope to understanding visual content in videos and visual transitions. We, therefore, compare AutoTransition's performance when using only visual inputs. Table 1 shows the results obtained from our experiments. Three notable observations emerge. Firstly, both methods mostly demonstrate superior performance with pre-trained transition embeddings compared to randomly initialized embeddings. This underscores the effectiveness of leveraging pre-trained embeddings for better mapping of the

Table 1: Quantitative Results. We show the quantitative results of our encoder-decoder model compared to AutoTransition [23] (only visual component included), our baseline.

Transition Embedding	Method	Evaluation Metrics		
		Recall@1↑	Recall@5↑	Mean Rank↓
Random	AutoTransition [23]	15.24	42.00	9.1
	V-Trans4Style($\mathcal{E}+\mathcal{D}$)	18.43	47.39	8.19
Transition Embedding [23] (Pre-Trained)	AutoTransition [23]	16.93	44.58	8.82
	V-Trans4Style($\mathcal{E}+\mathcal{D}$)	23.24	56.06	6.77
$\mathcal{MLN}$ Transition Embedding (Pre-Trained)	AutoTransition [23]	12.12	36.66	10.48
	V-Trans4Style($\mathcal{E}+\mathcal{D}$)	22.59	61.03	5.90

visual space to the transition space. Secondly, our encoder-decoder model consistently outperforms AutoTransition across all metrics, regardless of the chosen type of embedding. This suggests that capturing temporal relations between transitions and visual content leads to more accurate recommendations for transitions. Thirdly, examining our model's performance with transition embeddings from [23]versus those from $\mathcal{MLN}$ reveals intriguing insights. Although the difference is subtle, it's worth noting that Recall@1 is higher when using [23]'s embeddings, whereas Recall@5 exhibits the opposite trend. This can be attributed to the distribution of transitions in the embedding space. Fig.5b(A) demonstrates better separation between transitions, explaining the higher precision of our encoder-decoder trained on [23]. On the other hand, Fig. 5b(B) illustrates closer proximity between transitions, elucidating the higher Recall@5 when trained using $\mathcal{MLN}$

Ablation Experiments on Encoder-Decoder Losses. Table 2 summarizes results from an ablation experiment on the two loss functions used. When using $L_{classification}$ alone, we observe the highest Recall@5 but the lowest Recall@1. Conversely, using only $L_{masked_triplet}$ shows the opposite trend. The combination of both losses provides the most balanced results. This behavior can be attributed to the specific strengths of each loss function, with $L_{classification}$ emphasizing broader classification accuracy and $L_{masked_triplet}$ focusing on intricate relationships within the data. The combination of both losses allows for a balanced optimization, leveraging the strengths of each to achieve a more comprehensive performance across both metrics.

Style conditioning Module ($\mathcal{SCM}$). To evaluate $\mathcal{SCM}$, we randomly selected 100 videos from the test dataset, each comprising more than 6 video clips. For each of the production styles defined in AutoTransition++ dataset, we computed the cosine-similarity between e_μ, i.e. the mean embedding of all the transitions predicted for a video and the desired video production style embedding. Table 3 shows the mean of the cosine similarity scores obtained across the 100 samples for each of the baseline methods. Higher similarity scores are better. Apart from

Table 2: Loss Ablation Experiments. We show ablation experiments on the loss functions used to train our encoder-decoder model. Bold denotes **best** while underline denotes second-best

Loss Function	Evaluation Metrics		
	Recall@1↑	Recall@5↑	Mean Rank↓
$L_{classification}$	11.63	**69.00**	11.41
$L_{masked_triplet}$	16.89	45.79	8.57
$L_{classification} + L_{masked_triplet}$	**22.59**	61.03	**5.90**

Table 3: Qualitative Results - SCM. We show the quantitative results of our algorithm to recommend transitions to enable the adaptation of video production styles with baselines. Bold denotes **best** while underline denotes second-best. The higher the similarity better it is.

Method	Video Production Styles				
	Vlog	Anime	Influencer	Photos	Nature
AutoTransition [23]	-0.007	**0.0308**	-0.089	-0.05	-0.025
V-Trans4Style($\mathcal{E}$ & $\mathcal{D}$)	-0.01	-0.035	-0.022	-0.018	0.013
V-Trans4Style($\mathcal{E}$ & $\mathcal{D}$, $\mathcal{SCM}$ w/o RRT)	0.018	0.0194	0.0102	0.0211	0.0496
V-Trans4Style($\mathcal{E}$ & $\mathcal{D}$, $\mathcal{SCM}$ w/ RRT)	**0.188**	-0.001	**0.227**	**0.7286**	**0.356**

Anime, we can observe that $\mathcal{SCM}$ improves the suitability of transitions for adapting to the desired production style.

7 Conclusion, Limitations and Future Work

We present V-Trans4Style, a bottom-up approach for recommending visual transitions that adapt videos to different production styles, such as documentaries, feature films, and dramas. Our 3-component algorithm inputs a video and desired production style class to output a sequence of recommended transitions. The encoder-decoder model maps the vision space to the transition space, ensuring visually and temporally consistent recommendations. The style conditioning module uses activation maximization to adjust the latent embedding, enabling the decoder to produce style-specific transitions. We demonstrate our results on AutoTransition++, a mini version of the AutoTransition dataset with 6k videos. While this work focuses on visual transitions, future research will explore other editing elements like camera motion, cuts, and audio. Nevertheless, our current work lays a crucial foundation for understanding video production styles and their connections to editing elements.

Acknowledgements

This work was done by Pooja Guhan as part of an internship with Dolby Laboratories.

References

1. Adobe Creative Cloud: Types of visual transitions. `https://www.adobe.com/creativecloud/video/discover/types-of-film-transitions.html`, accessed: 2023-11-13
2. Alghamdi, E.A.: What makes video-based academic lectures difficult for language learners to comprehend? the role of multimodal complexity. Foreign Language Annals (2024)
3. Bain, M., Nagrani, A., Brown, A., Zisserman, A.: Condensed movies: Story based retrieval with contextual embeddings. In: Proceedings of the Asian Conference on Computer Vision (2020)
4. Chen, B., Ziai, A., Tucker, R.S., Xie, Y.: Match cutting: Finding cuts with smooth visual transitions. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2115–2125 (2023)
5. Cloud, A.C.: Adobe premiere pro cc. línea) Disponible en: http://www. adobe. com/mx/products/premiere. html (2014)
6. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. University of Montreal **1341**(3), 1 (2009)
7. Feichtenhofer, C., Fan, H., Malik, J., He, K.: Slowfast networks for video recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6202–6211 (2019)
8. Frühstück, A., Sarafianos, N., Xu, Y., Wonka, P., Tung, T.: Vive3d: Viewpoint-independent video editing using 3d-aware gans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4446–4455 (2023)
9. Gao, Y., Cao, Y., Kou, T., Sun, W., Dong, Y., Liu, X., Min, X., Zhai, G.: Vdpve: Vqa dataset for perceptual video enhancement. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1474–1483 (2023)
10. Hassanien, A., Elgharib, M., Selim, A., Bae, S.H., Hefeeda, M., Matusik, W.: Large-scale, fast and accurate shot boundary detection through spatio-temporal convolutional neural networks. arXiv preprint arXiv:1705.03281 (2017)
11. Imambi, S., Prakash, K.B., Kanagachidambaresan, G.: Pytorch. Programming with TensorFlow: Solution for Edge Computing Applications pp. 87–104 (2021)
12. Kalluri, T., Pathak, D., Chandraker, M., Tran, D.: Flavr: Flow-agnostic video representations for fast frame interpolation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2071–2082 (2023)
13. Karaman, S., Walter, M.R., Perez, A., Frazzoli, E., Teller, S.: Anytime motion planning using the rrt. In: 2011 IEEE international conference on robotics and automation. pp. 1478–1483. IEEE (2011)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Lee, S.H., Kim, S., Yoo, I., Yang, F., Cho, D., Kim, Y., Chang, H., Kim, J., Kim, S.: Soundini: Sound-guided diffusion for natural video editing. arXiv preprint arXiv:2304.06818 (2023)

16. Nam, J., Tewfik, A.H.: Detection of gradual transitions in video sequences using b-spline interpolation. IEEE Transactions on Multimedia **7**(4), 667–679 (2005)
17. Narasimhan, M., Rohrbach, A., Darrell, T.: Clip-it! language-guided video summarization. Advances in Neural Information Processing Systems **34**, 13988–14000 (2021)
18. Niklaus, S., Hu, P., Chen, J.: Splatting-based synthesis for video frame interpolation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 713–723 (2023)
19. Pardo, A., Heilbron, F.C., Alcázar, J.L., Thabet, A., Ghanem, B.: Moviecuts: A new dataset and benchmark for cut type recognition. In: European Conference on Computer Vision. pp. 668–685. Springer (2022)
20. Qi, C., Cun, X., Zhang, Y., Lei, C., Wang, X., Shan, Y., Chen, Q.: Fatezero: Fusing attentions for zero-shot text-based video editing. arXiv preprint arXiv:2303.09535 (2023)
21. Rao, A., Jiang, X., Wang, S., Guo, Y., Liu, Z., Dai, B., Pang, L., Wu, X., Lin, D., Jin, L.: Temporal and contextual transformer for multi-camera editing of tv shows. arXiv preprint arXiv:2210.08737 (2022)
22. Robbins, H., Monro, S.: A stochastic approximation method. The annals of mathematical statistics pp. 400–407 (1951)
23. Shen, Y., Zhang, L., Xu, K., Jin, X.: Autotransition: Learning to recommend video transition effects. In: European Conference on Computer Vision. pp. 285–300. Springer (2022)
24. Tang, S., Feng, L., Kuang, Z., Chen, Y., Zhang, W.: Fast video shot transition localization with deep structured models. In: Asian Conference on Computer Vision. pp. 577–592. Springer (2018)
25. Viewinder: Film school: How editing can transform genre. `https://viewinder.com/editing-transforms-genre/`
26. Wang, W., Xie, K., Liu, Z., Chen, H., Cao, Y., Wang, X., Shen, C.: Zero-shot video editing using off-the-shelf image diffusion models. arXiv preprint arXiv:2303.17599 (2023)
27. Yasmin, G., Chowdhury, S., Nayak, J., Das, P., Das, A.K.: Key moment extraction for designing an agglomerative clustering algorithm-based video summarization framework. Neural Computing and Applications **35**(7), 4881–4902 (2023)
28. Zhang, T., Tian, F., Hou, X., Xie, Q., Yi, F.: Evaluating the effect of transitions on the viewing experience for vr video. In: 2018 International Conference on Audio, Language and Image Processing (ICALIP). pp. 273–277. IEEE (2018)
29. Zhang, Y., Kampffmeyer, M., Liang, X., Tan, M., Xing, E.P.: Query-conditioned three-player adversarial network for video summarization. arXiv preprint arXiv:1807.06677 (2018)
30. Zhou, J.: The analysis of short film shooting and editing techniques. In: 2nd International Conference on Education, Language and Art (ICELA 2022). pp. 322–329. Atlantis Press (2023)