Action Detection via an Image Diffusion Process (Supplementary Material)

Lin Geng Foo[1] Tianjiao Li[1] Hossein Rahmani[2] Jun Liu[1†]
[1]Singapore University of Technology and Design [2]Lancaster University
{lingeng_foo,tianjiao_li}@mymail.sutd.edu.sg,
h.rahmani@lancaster.ac.uk, jun_liu@sutd.edu.sg

1. More Experiments

Here, we report results of more experiments. These experiments are conducted on the THUMOS14 dataset, unless otherwise specified.

Closer comparison against state-of-the-art baseline. We also closely compare the performance between our method and the state-of-the-art baseline. Specifically, we evaluate the quality of our temporal boundary outputs (i.e., starting point and ending point AD images) as well as our action-class outputs (i.e., action-class AD image), by replacing the corresponding components of a current state-of-the-art baseline [10] with them. We compare against the following two baselines: 1) **Baseline w/ our temporal boundary outputs** where we obtain the temporal boundaries (i.e., starting and ending points) with our method, and obtain the action-class predictions with the baseline method. 2) **Baseline w/ our action-class outputs** where we obtain the temporal boundaries (i.e., starting and ending points) with the baseline method, and obtain the action-class predictions with our method. As shown by the reported results in Tab. 1, our method outperforms the baselines, showing that our method is able to effectively generate both high-quality temporal boundary and action-class AD images which demonstrate good performance when compared to the state-of-the-art baseline.

Table 1. Evaluation of temporal boundary and action-class outputs against a state-of-the-art baseline [10].

Setting	0.3	0.5	0.7	Avg
Baseline [10]	83.6	72.9	47.4	69.3
Baseline w/ our temporal boundary outputs	84.1	74.9	47.7	69.8
Baseline w/ our action-class outputs	84.0	74.6	47.6	69.9
Ours	84.9	76.5	48.0	70.8

Impact of number of samples M. We explore varying the hyperparameter M, with results shown in Tab. 2. We find that performance improves when we increase M, until M reaches 10, where the improvement tapers off. Thus, we set $M = 10$.

† Corresponding author

Table 2. Ablation study for number of samples M.

M	0.3	0.5	0.7	Avg
1	82.0	73.7	45.2	67.2
10	84.9	76.5	48.0	70.8
20	85.0	75.2	47.9	70.7

Impact of depth of diffusion network L. We also explore changing the depth (L) of the diffusion network with the Row-Column Transformer Architecture and report the results in Tab. 3. We find that performance improves when we increase L, until L reaches 3, where the improvements become minimal when we increase it further. Thus, we set $L = 3$.

Table 3. Ablation study for depth of diffusion model (L).

Layers	0.3	0.5	0.7	Avg
1	83.2	75.0	47.1	69.2
3	84.9	76.5	48.0	70.8
5	84.0	76.1	48.2	70.3

Impact of number of diffusion steps T. To further investigate the characteristics of our diffusion process, we experiment with different diffusion step numbers (T) and report the results in Tab. 4. We observe that performance improves significantly when we increase T, and tapers off when $T > 50$. Thus, we set $T = 50$ to achieve a good result while maintaining efficiency.

Table 4. Ablation study for number of sampling steps T.

T	0.3	0.5	0.7	Avg
10	81.9	74.3	46.5	68.1
30	83.7	75.1	47.0	69.7
50	84.9	76.5	48.0	70.8
70	84.5	76.3	48.0	70.6

Impact of number of trials K. We further investigate the impact of the "number of trials" hyperparameter K of the Multinomial distribution and report the results in Tab. 5. We observe that performance improves when we increase K, and tapers off when $K > 200$. Thus, we set $K = 200$.

Impact of diffusion process. We further evaluate the efficacy of employing a diffusion process by comparing against

Table 5. Ablation study for hyperparameter K.

K	0.3	0.5	0.7	Avg
100	82.5	75.2	45.0	67.8
200	84.9	76.5	48.0	70.8
400	84.5	76.4	47.9	70.2
600	84.6	76.3	47.8	70.4

two baseline models: (1) **Baseline A**: We use the same network as our diffusion model but we do not perform diffusion here, and instead obtain the predictions in a single step. (2) **Baseline B**: This baseline is similar to Baseline A, except that the network is stacked multiple times to approximate the computational complexity of our method. We report the results of the baselines and our method in Tab. 6. The performance of both baselines are much worse than ours, which demonstrates the efficacy of diffusion.

Table 6. Ablation study for diffusion process.

Method	0.3	0.5	0.7	Avg
Baseline A	79.2	66.4	43.1	64.9
Baseline B	78.7	65.1	43.8	65.1
Ours	84.9	76.5	48.0	70.8

Impact of diffusion's properties. We investigate the importance of the two properties mentioned in Sec. 3.2 of the main paper. Specifically, we compare against a baseline (**w/o Diffusion Properties**) that does not make use of Property 2 and Property 3 to facilitate training, and has to add noise in a step-by-step manner to generate every single noisy sample. Training times of each epoch under each setting are reported in Tab. 7. We find that the training efficiency drops by a lot without these properties, showing their importance.

Table 7. Ablation study of diffusion's properties.

Method	Training time (hr)
w/o Diffusion Properties	2.3
w/ Diffusion Properties	0.3

Comparison between different backbones. In our experiments, we follow existing works [10, 14] to use an off-the-shelf R(2+1)D [12] models to extract video features f_{ST} for ActivityNet-1.3. To fairly compare with other works that use the I3D [2] backbone, here we run experiments on ActivityNet-1.3 using the I3D backbone as well. As reported in Tab. 8, we find that our performance is roughly the same and still achieves state-of-the-art results, outperforming other methods that use the I3D backbone (as reported in Tab. 1 of the main paper).

Table 8. Comparison between different backbones on ActivityNet-1.3.

Feature	0.5	0.75	0.95	Avg
I3D	56.1	38.2	8.7	37.7
R(2+1)D	56.9	38.9	9.1	38.3

Visualization of challenging examples. Here, we visualize the results of our method on some challenging video examples on THUMOS14 dataset (e.g., with cluttered background or complex motions) in Fig. 1. Specifically, we visualize the predicted probability of the ground truth action-class and the predicted probability of having a starting point and ending point across several selected frames for a few video clips. We observe that our model outputs high probability scores for the correct action-class during the action, while the probability scores for the starting and ending times also peak at the start and end of the actions respectively. This shows the efficacy of our model in producing high-accuracy predictions for action-class and temporal boundary predictions. Moreover, we find that our method performs well even under challenging conditions, such as complex motions and a cluttered background, as shown in the top and bottom examples in Fig. 1 respectively. This suggests that there is potential for future works to explore our method in fine-grained recognition settings [6] as well.

2. More Details

2.1. More Network Details

The main structure of our diffusion network d has been described in Sec. 3.3 of the main paper, where d mainly consists of L stacks of Row-Column Blocks. Below, we describe more details regarding the network architecture.

When extracting the features from the pre-trained encoder, we first extract features $f_v \in \mathbb{R}^{N \times F}$, where N is the number of frames and F is the feature size of the extracted features (e.g., $F = 2048$ for I3D). Then, two linear+relu layers are applied followed by a max pooling layer, such that we obtain the spatio-temporal features $f_{ST} \in \mathbb{R}^{N \times C_{ST}}$, where C_{ST} is set to 128.

Next, we also generate a diffusion step embedding $f_t \in \mathbb{R}^{N \times 1}$ via the sinusoidal function to represent the t-th diffusion step. More precisely, at each even $(2i)$ index of f_t, we set the element $f_t[2i]$ to $sin(t/10000^{2i/N})$, while at each odd $(2i+1)$ index, we set the element $f_t[2i+1]$ to $cos(t/10000^{2i/N})$.

For our MHSA layers, we set the number of self-attention heads to 8. Our Temporal Convolutions are set to have 4 channels. In both parts of the Row-Column Block, we have a 2-layer MLP. In both cases, the inputs and outputs of the 2-layer MLP have shape $N \times (C + C_{ST} + 1)$. Specifically, in the first part, the MLP encodes information across columns while in the second part, the MLP encodes information across rows. Also note that, when passing through the MLP layers in both parts, the shape of the hidden features remain unchanged.

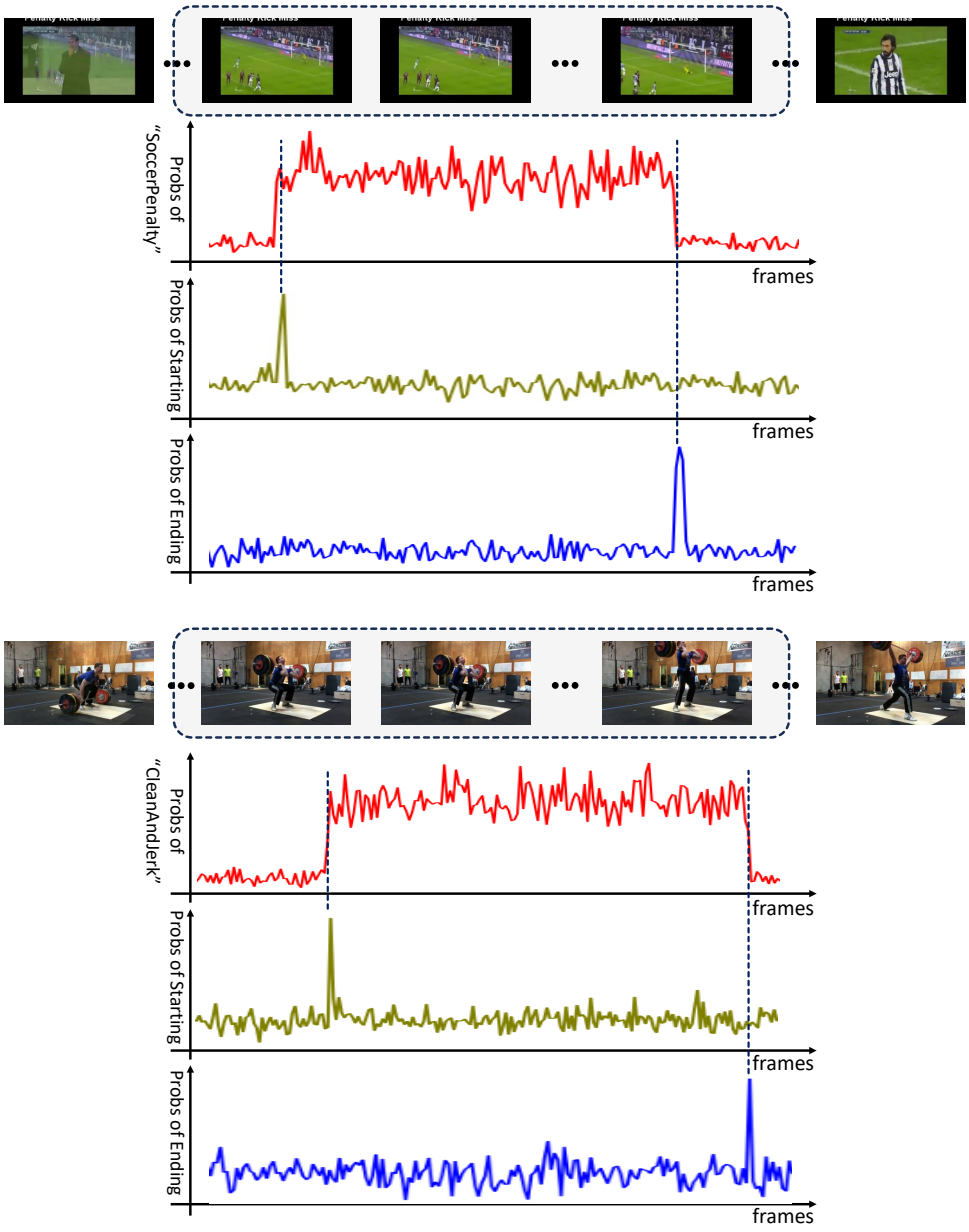

Figure 1. Visualization of action detection results. (Top) A video clip of a "Soccer Penalty" action, which is a complex action that consists of several different motions. Our model successfully assigns a high probability for the "Soccer Penalty" class during the action, and assigns high probabilities to the correct starting and ending points (as indicated by the vertical dotted lines). (Bottom) A video clip of a "Clean and Jerk" action, which is captured amidst a cluttered background, making it rather challenging for models to produce accurate predictions. We find that, in this challenging scenario, our model still produces rather accurate action-class predictions and starting/ending point predictions, showing the efficacy of our method.

2.2. Network Architecture for Combined Processing

We perform combined processing for generating the three AD images simultaneously, where we stitch the images together into a combined image and perform a combined processing for all three AD images. Specifically, we concatenate the three AD images $\{x_t^a, x_t^s, x_t^e\}$ horizontally to obtain the combined image as $x_t^{combined} \in \mathbb{R}^{N \times (C+4)}$, where each row consists of three discrete distributions. Then, to process the stitched image with the diffusion network, we modify the column dimensionality, i.e., by adding 4 columns to

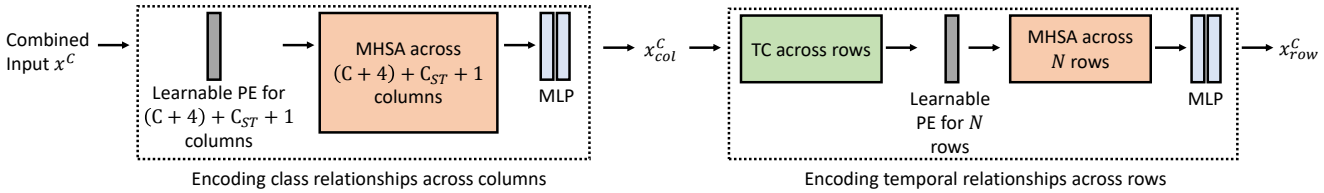

Figure 2. Illustration of the Row-Column Block with combined processing of all three AD images (x^a, x^s, x^e), where the combined input x^C is fed into the block. Although the components and processes of the Row-Column Block for combined processing are mostly similar to individual processing, there are some modifications in terms of the dimensions. For instance, in the first part of the Row-Column Block, the learnable positional embeddings (PE) are adjusted to handle $(C + 4) + C_{ST} + 1$ columns, and the MHSA is also performed across $(C + 4) + C_{ST} + 1$ columns. Furthermore, in the second part of the Row-Column Block, the learnable PE and MHSA are adjusted to handle tokens (rows) with a size of $(C + 4) + C_{ST} + 1$. The shapes of MLP weights in both parts are also adjusted accordingly.

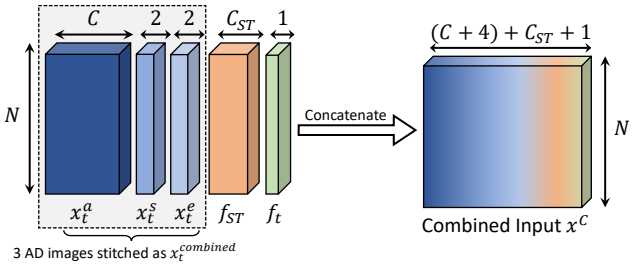

Figure 3. Illustration of how the three AD images (x^a, x^s, x^e) are concatenated with spatio-temporal features f_{ST} and diffusion step embedding f_t to form the combined input x^C to the diffusion network for combined processing. In this illustration, each AD image is represented by a rectangular block, where the number of columns in each AD image are indicated on top of the AD images. The resulting combined input to the diffusion network (x^C) has shape $N \times (C + 4) + C_{ST} + 1$.

the Row-Column Block design. Note that, even though the processing by the network is combined, the three discrete distributions in each row (from three different AD images) are still handled separately in our diffusion process, where the processing of each distribution follows Sec. 3.2 of the main paper. An overview of the combined processing with our Row-Column Block is shown in Fig. 2. We provide more details below.

Firstly, similar to how individual AD images are processed, the combined image $x_t^{combined} \in \mathbb{R}^{N\times(C+4)}$ are concatenated with spatio-temporal features $f_{ST} \in \mathbb{R}^{N\times C_{ST}}$ and diffusion step embedding $f_t \in \mathbb{R}^{N\times C_{ST}}$, to form the input $x^C \in \mathbb{R}^{N\times(C+4)+C_{ST}+1}$. We remark that the f_{ST} and f_t in this combined image remain unchanged as compared to the individual processing. We visualize this in Fig. 3.

Then, in the first part of the Row-Column Block, we treat each column of the input $x^C \in \mathbb{R}^{N\times(C+4)+C_{ST}+1}$ as a token, thus obtaining $(C + 4) + C_{ST} + 1$ tokens of length N. A learnable positional embedding is added to each token to encode the positional information, and then MHSA is performed among all tokens, where the number of self-attention heads is kept at 8. We also reshape the weights of the last 2 MLP layers accordingly, to have inputs and outputs of shape $N \times (C + 4) + C_{ST} + 1$.

In the second part of the Row-Column Block, we first apply a 1×3 Temporal Convolution as usual. Then, we treat each row as a feature, obtaining N tokens of length $(C + 4) + C_{ST} + 1$. Afterwards, we add the learnable positional embedding to each token and perform MHSA between them, where the weights are adjusted to handle tokens of length $(C + 4) + C_{ST} + 1$. We also reshape the weights of the last 2 MLP layers accordingly, to have inputs and outputs of shape $N \times (C + 4) + C_{ST} + 1$.

We remark that some parts of the network are shared between all three types of AD images, while some parts are not shared. Specifically, all three AD images share the set of network parameters for the weights in MHSA and TC layers. However, the positional embeddings of the tokens are all learned separately, so the tokens of each AD image can learn unique positional embeddings for themselves.

2.3. More Implementation Details

For the diffusion process, we follow DDIM [11] to define the sequence $\{\beta_1, ..., \beta_T\}$ by linearly interpolating from $1e-4$ to $2e-2$, i.e., $\beta_1 = 1e-4$ and $\beta_T = 2e-2$. Then, as mentioned in the main paper, we define α_t and $\bar{\alpha}_t$ based on β_t, according to the following equations: $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{k=1}^{t} \alpha_k$. At inference time, we also adopt the acceleration technique from DDIM [11], which allows us to skip a proportion of diffusion steps.

To facilitate training, following previous works [7, 8], we consider a small window of frames around the labelled starting/ending points as positives. Specifically, we compute this small window as $\frac{1}{10}$ of the length of the action instance. Furthermore, to balance between the positives and negatives for the starting/ending point prediction, we follow existing work [7] by setting threshold IoP as 0.5, where IoP is defined as the overlap ratio with ground truth proportional to the duration of the action instances. To be more specific, when $IoP > 0.5$, the action instances are considered positive; otherwise, they will be considered negative.

3. More Proofs and Analysis

3.1. Details of Multinomial Distribution

In Sec. 3.2 of the main paper, we describe our forward diffusion process, where the random noise v_t added at each step is sampled from a Multinomial distribution. Here, we formally define the Multinomial distribution and provide more details.

We denote the Multinomial distribution as $Multinomial(K, \boldsymbol{p})$, where K is a parameter that represents the number of total trials and $\boldsymbol{p}$ represents a vector of probabilities of length C which add up to 1. Intuitively, the Multinomial distribution describes the distribution of counts over the C classes, when classes are picked K times with replacement where the probability of picking each class is represented by an element in $\boldsymbol{p}$. The likelihood function in terms of $\boldsymbol{x}$, where $\boldsymbol{x}$ is a vector that represents non-negative integer counts for all classes, is defined as follows:

$$Multinomial(\boldsymbol{x}; K, \boldsymbol{p}) = \frac{\Gamma(K+1)}{\prod_{c=1}^{C}\Gamma(x_c+1)}\prod_{c=1}^{C} p_c^{x_c} \tag{1}$$

Note that, the elements of the vector $\boldsymbol{x}$ sum to K. In the case where we compute the Multinomial likelihood of v_t instead, where the elements sum to 1, we can easily perform substitution in the probability mass function (as denoted by the subscript Kv_t) as follows:

$$Multinomial_{Kv_t}(v_t; K, \boldsymbol{p}) = \frac{\Gamma(K+1)}{\prod_{c=1}^{C}\Gamma(Kv_{t,c}+1)}\prod_{c=1}^{C} p_c^{Kv_{t,c}} \tag{2}$$

For the sake of simplicity, we use the formulation of $\boldsymbol{x}$ in Eq. 1 for the rest of this section (where sum of elements in $\boldsymbol{x}$ is K), but the statements and results apply to v_ts as well. At the same time, we use $Multinomial_K(K, \boldsymbol{p})$ to denote sampling v_t from the Multinomial in Eq. 2, i.e., sampling from $Multinomial(K, \boldsymbol{p})$ and dividing the output counts by K, in order to let all elements of v_t sum to 1.

3.2. Analysis of t forward steps

Next, we show how we can obtain Eq. 4 of the main paper from the t-step forward process as shown in Eq. 3 of the main paper.

For convenience, we repeat Eq. 3 of the main paper here:

$$z_t = \bar{\alpha}_t z_0 + (\prod_{\tau=2}^{t}\alpha_\tau)\beta_1 v_1 + (\prod_{\tau=3}^{t}\alpha_\tau)\beta_2 v_2 + ... + \beta_t v_t \tag{3}$$

We show how we can approximately obtain the simplified likelihood of the t-step forward process, ie., Eq. 4 of the main paper, as follows:

$$q(z_t|z_0) = Multinomial_{\frac{B_t K(z_t - \bar{\alpha}_t z_0)}{1-\bar{\alpha}_t}}(z_t; B_t K, \frac{1}{C}\mathbf{1}, z_0) \tag{4}$$

where the Multinomial's subscript is the substitution formula ($\frac{B_t K(z_t - \bar{\alpha}_t z_0)}{1-\bar{\alpha}_t}$) in terms of z_t which is used for formulating the likelihood function (i.e., similar to the substitution in Eq. 2), where $\frac{B_t K(z_t - \bar{\alpha}_t z_0)}{1-\bar{\alpha}_t}$ in terms of z_t follows the Multinomial distribution.

Next, in order to simplify the sum from Eq. 3, we first show that the sum of samples from Multinomials with the same $\boldsymbol{p}$ will produce a sample from another Multinomial. Specifically, if $m = x_1 + x_2$, and $x_1 \sim Multinomial(K_1, \boldsymbol{p})$, $x_2 \sim Multinomial(K_2, \boldsymbol{p})$, then $m \sim Multinomial(K_1 + K_2, \boldsymbol{p})$. To show this, we use the characteristic functions of the Multinomial distributions. We note that the characteristic function (CF) of the Multinomial distribution is:

$$CF_{Multinomial(\boldsymbol{x}; K, \boldsymbol{p})}(\boldsymbol{t}) = \mathbb{E}[e^{i\boldsymbol{t}\boldsymbol{x}}] = \left(\sum_{j=1}^{C} p_j e^{it_j}\right)^K \tag{5}$$

where $i^2 = -1$, and $\boldsymbol{t}$ is an input argument which is a vector of length C.

Since the sum of two random variables is equals to the product of their characteristic functions, we get:

$$\begin{aligned} CF_{x_1+x_2}(\boldsymbol{t}) &= CF_{x_1}(\boldsymbol{t}) CF_{x_2}(\boldsymbol{t}) && (6)\\ &= \left(\sum_{j=1}^{C} p_j e^{it_j}\right)^{K_1}\left(\sum_{j=1}^{C} p_j e^{it_j}\right)^{K_2} && (7)\\ &= \left(\sum_{j=1}^{C} p_j e^{it_j}\right)^{K_1+K_2} && (8)\\ &= CF_{Multinomial(\boldsymbol{x}; K_1+K_2, \boldsymbol{p})}(\boldsymbol{t}), && (9) \end{aligned}$$

which shows that the resulting sample is from the $Multinomial(K_1 + K_2, \boldsymbol{p})$ distribution.

In our case specifically, we also need to add scaling factors (β_t) that re-weigh the samples from the Multinomial distribution at each t-th step. Using the proof above, we can observe that, when samples from Multinomials are added, they remain a Multinomial. However, when we add a linear combination of samples with different β_t, it becomes more difficult to exactly quantify the resulting likelihood. To derive a good approximation for the likelihood of h_t (and therefore $q(z_t|z_0)$), we introduce Lemma 1, which is described as follows:

Lemma 1. *If* $h_t = (\prod_{\tau=2}^{t}\alpha_\tau)\beta_1 x_1 + (\prod_{\tau=3}^{t}\alpha_\tau)\beta_2 x_2 + ... + \beta_t x_t$, *where* $x_\tau \sim Multinomial(K, \frac{1}{C}\mathbf{1})$ *for*

$\tau \in [1, ..., t]$ and x_τ are all independently sampled, then the following approximately holds: $\frac{B_t}{1-\bar{\alpha}_t} h_t \sim Multinomial(B_t K, \frac{1}{C}\mathbf{1})$ *where* $\bar{\alpha}_t = \prod_{\tau=1}^{t} \alpha_\tau$ *and* $B_t = \frac{(1-\bar{\alpha}_t)^2}{\left((\prod_{\tau=2}^{t} \alpha_\tau)^2 \beta_1^2 + (\prod_{\tau=3}^{t} \alpha_\tau)^2 \beta_2^2 + ... + \beta_t^2\right)}$

Proof. To derive the approximation, we make use of the following property of multinomial distributions: as K becomes large, the likelihood over each dimension can be approximated with a Gaussian. In our case, with a $Multinomial(K, \frac{1}{C}\mathbf{1})$, the approximating Gaussian has mean and variance parameters of $\frac{K}{C}$ and $\frac{K}{C}(1 - \frac{1}{C})$. Thus, when we set our K to be large, the likelihood over each dimension of x_τ is approximated as: $\mathcal{N}(\frac{K}{C}, \frac{K}{C}(1 - \frac{1}{C}))$.

Furthermore, the convolution of two independent Gaussians is a Gaussian with the expectation and variance being the sum of the constituent Gaussians, i.e., $\mathcal{N}(\mu_1, \mathbb{V}_1) * \mathcal{N}(\mu_2, \mathbb{V}_2) = \mathcal{N}(\mu_1 + \mu_2, \mathbb{V}_1 + \mathbb{V}_2)$. Thus, by approximating the likelihood of each multinomial with a Gaussian, we can find the parameters of h_t through computing the mean and variance of each constituent x_τ and summing them up. Therefore, the sum of $(\prod_{\tau=2}^{t} \alpha_\tau)\beta_1 x_1 + (\prod_{\tau=3}^{t} \alpha_\tau)\beta_2 x_2 + ... + \beta_t x_t$ at the t-th step for any given dimension can be approximated to be a Gaussian with mean and variance as:

$$\mathbb{E}[h_t] = (\prod_{\tau=2}^{t} \alpha_\tau)\beta_1 \mu_{v_1} + (\prod_{\tau=3}^{t} \alpha_\tau)\beta_2 \mu_{v_2} + ... + \beta_t \mu_{v_t} \tag{10}$$

$$= ((\prod_{\tau=2}^{t} \alpha_\tau)\beta_1 + (\prod_{\tau=3}^{t} \alpha_\tau)\beta_2 + ... + \beta_t)\frac{K}{C} \tag{11}$$

$$= (1 - \bar{\alpha}_t)\frac{K}{C} \tag{12}$$

$$\mathbb{V}[h_t] = (\prod_{\tau=2}^{t} \alpha_\tau)\beta_1 \mathbb{V}_{v_1} + (\prod_{\tau=3}^{t} \alpha_\tau)\beta_2 \mathbb{V}_{v_2} + ... + \beta_t \mathbb{V}_{v_t} \tag{13}$$

$$= ((\prod_{\tau=2}^{t} \alpha_\tau)^2 \beta_1^2 + (\prod_{\tau=3}^{t} \alpha_\tau)^2 \beta_2^2 + ... + \beta_t^2)\frac{K}{C}(1 - \frac{1}{C}) \tag{14}$$

$$= \gamma_t \frac{K}{C}(1 - \frac{1}{C}) \tag{15}$$

where $\gamma_t = \left((\prod_{\tau=2}^{t} \alpha_\tau)^2 \beta_1^2 + (\prod_{\tau=3}^{t} \alpha_\tau)^2 \beta_2^2 + ... + \beta_t^2\right)$.

Next we aim to find the parameters of a Multinomial, of which the likelihood can be approximated by a Gaussian with the expectation and variance according to Eq. 12 and Eq. 15 above. Specifically, if we draw a sample y_t from the Multinomial, the expectation and variance of y_t should follow Eq. 12 and Eq. 15, i.e., such that it is approximately drawn from a Gaussian with those parameters.

Firstly, it is clear that the probability parameters should be kept constant at $\frac{1}{C}\mathbf{1}$ for the resulting Multinomial as well. Furthermore, it is straightforward to design the expectation to be equal to Eq. 12, where we can scale the original trial parameter K by a constant B_t and scale the sample y_t further by $\frac{1}{B_t} \times (1 - \bar{\alpha}_t)$, which will ensure that the expectations are equal.

Then, it is simple to check that expectation of $\frac{(1-\bar{\alpha}_t)}{B_t} y_t$, where $y_t \sim Multinomial(B_t K, \frac{1}{C}\mathbf{1})$, is as follows:

$$\mathbb{E}[\frac{(1 - \bar{\alpha}_t)}{B_t} y_t] = \frac{(1 - \bar{\alpha}_t)}{B_t}\mathbb{E}[y_t] \tag{16}$$

$$= \frac{(1 - \bar{\alpha}_t)}{B_t}\frac{B_t K}{C}\mathbf{1} \tag{17}$$

$$= (1 - \bar{\alpha}_t)\frac{K}{C}\mathbf{1} \tag{18}$$

which is equal to Eq. 12 as expected.

At the same time, we can solve for the value of B_t by expanding the variance of $\frac{(1-\bar{\alpha}_t)}{B_t} y_t$ as follows:

$$\mathbb{V}[\frac{(1 - \bar{\alpha}_t)}{B_t} y_t] = \frac{(1 - \bar{\alpha}_t)^2}{B_t^2}\mathbb{V}[y_t] \tag{19}$$

$$= \frac{(1 - \bar{\alpha}_t)^2}{B_t^2}\frac{B_t K}{C}\mathbf{1}(\mathbf{1} - \frac{1}{C}\mathbf{1}) \tag{20}$$

$$= \frac{(1 - \bar{\alpha}_t)^2}{B_t}\frac{K}{C}\mathbf{1}(\mathbf{1} - \frac{1}{C}\mathbf{1}) \tag{21}$$

Therefore, by equating Eq. 15 and Eq. 21, we find that we can approximate the distribution of h_t with $\frac{(1-\bar{\alpha}_t)}{B_t} y_t$ by setting $B_t = \frac{(1-\bar{\alpha}_t)^2}{\gamma_t}$, where we check that the variance of each element of $\frac{(1-\bar{\alpha}_t)}{B_t} y_t$ is as follows:

$$\frac{(1 - \bar{\alpha}_t)^2}{B_t}\frac{K}{C}(1 - \frac{1}{C}) = (1 - \bar{\alpha}_t)^2 \frac{\gamma_t}{(1 - \bar{\alpha}_t)^2}\frac{K}{C}(1 - \frac{1}{C}) \tag{22}$$

$$= \gamma_t \frac{K}{C}(1 - \frac{1}{C}) \tag{23}$$

which is equal to what we obtain in Eq. 15.

We further remark that the same calculations for the variance can be extended for the covariance as well, to show that the covariance between elements of h_t and $\frac{(1-\bar{\alpha}_t)}{B_t} y_t$ are approximately equal when $B_t = \frac{(1-\bar{\alpha}_t)^2}{\gamma_t}$. In short, for all $i \neq j$, $Cov(h_{t,i}, h_{t,j}) = -\gamma_t K \frac{1}{C^2}$, and similarly, $Cov(\frac{(1-\bar{\alpha}_t)}{B_t} y_{t,i}, \frac{(1-\bar{\alpha}_t)}{B_t} y_{t,j}) = -\frac{(1-\bar{\alpha}_t)^2}{B_t} K \frac{1}{C^2}$, and the values are equal when $B_t = \frac{(1-\bar{\alpha}_t)^2}{\gamma_t}$.

Note that, if $B_t K$ is not an integer, we can round it off to the nearest integer. Furthermore, $B_t K > K$ since $B_t > 1$ for all t.

□

In summary, using Lemma 1, the following approximately holds:

$$\frac{B_t}{(1-\bar{\alpha}_t)} h_t \sim Multinomial(B_t K, \frac{1}{C}\mathbf{1}) \tag{24}$$

where $h_t = (\prod_{k=2}^{t} \alpha_k)\beta_1 x_1 + (\prod_{k=3}^{t} \alpha_k)\beta_2 x_2 + ... + \beta_t x_t$.

Overall, this allows us to formulate Eq. 4 as follows:

$$q(z_t|z_0) = Multinomial_{\frac{B_t K(z_t - \bar{\alpha}_t z_0)}{1-\bar{\alpha}_t}}(z_t; B_t K, \frac{1}{C}\mathbf{1}, z_0) \tag{25}$$

3.3. Modeling the forward diffusion steps as a Markov Chain

In this subsection, we show that the transitions using Eq. 1 of the main paper can be modelled as a discrete-time Markov chain. Specifically, each forward step can be treated as a transition in a Markov chain. In other words, we can take Eq. 1 of the main paper, and formulate it as a transition matrix P_t of a Markov chain, representing the transition matrix at the t-th step. P_t is a matrix of size $C \times C$, where the element at the i-th row and j-th column indicates the probability of the i-th state transitioning to the j-th state. Directly using Eq. 1 of the main paper, we can set the elements of transition matrix P_t to be:

$$1 - \beta_t + \beta_t v_{t,i} \quad \text{for each } i\text{-th diagonal entry}, \tag{26}$$

$$\beta_t v_{t,j} \quad \text{for each } i, j\text{-th non-diagonal entry}. \tag{27}$$

where P_t is a matrix of size $C \times C$, and $v_{t,i}$ refers to the i-th entry of the vector v_t, which is a randomized result from the Multinomial distribution.

We can quickly verify this transition probability matrix by computing $z_{t-1}P_t$, which can give us Eq. 1 of the main paper. Using the transition probabilities in Eq. 26 and Eq. 27, the probability value of j-th class at the t-th step is as follows:

$$z_{t,j} = z_{t-1,j}(1 - \beta_t + \beta_t v_{t,j}) + \sum_{c \in \{1,....,C\} \setminus j} z_{t-1,c}\beta_t v_{t,j} \tag{28}$$

$$= (1-\beta_t) z_{t-1,j} + \beta_t \sum_{c \in \{1,....,C\}} z_{t-1,c} v_{t,j} \tag{29}$$

$$= (1-\beta_t) z_{t-1,j} + \beta_t v_{t,j} \tag{30}$$

Since the above derivations hold for all classes, this verifies the Markov chain is equivalent to Eq. 1 of the main paper.

Using the Markov chain, it is also easy to derive Eq. 3 of the main paper via the recurrence relation.

3.4. More analysis of the forward process posterior

Here, we derive the formulation in Eq. 5 of the main paper.

Firstly, from Bayes' Theorem, we have:

$$q(z_{t-1}|z_t, z_0) = \frac{q(z_t|z_{t-1}, z_0) \cdot q(z_{t-1}|z_0)}{q(z_t|z_0)} \tag{31}$$

$$= \frac{q(z_t|z_{t-1}) \cdot q(z_{t-1}|z_0)}{q(z_t|z_0)} \tag{32}$$

$$= \frac{q(z_t|z_{t-1}) \cdot q(z_{t-1}|z_0)}{\sum_{z_{t-1}} q(z_t|z_{t-1}) \cdot q(z_{t-1}|z_0)} \tag{33}$$

where the second step is due to the Markov property, i.e., $q(z_t|z_{t-1}, z_0) = q(z_t|z_{t-1})$. Note that we verify that the forward process is a Markov chain in Sec. 3.3.

From Eq. 4 of the main paper, we can get:

$$q(z_{t-1}|z_0) = Multinomial_{\frac{B_{t-1} K(z_{t-1} - \bar{\alpha}_{t-1} z_0)}{1-\bar{\alpha}_{t-1}}}(z_{t-1}; B_{t-1} K, \frac{1}{C}\mathbf{1}, z_0) \tag{34}$$

We can also get the following:

$$q(z_t|z_{t-1}) = q(z_{t-1}|z_t) = Multinomial_{\frac{K(z_t - (1-\beta_t) z_{t-1})}{\beta_t}}(z_{t-1}; K, \frac{1}{C}\mathbf{1}, z_t), \tag{35}$$

Substituting in the values, we have:

$$\begin{aligned} q(z_{t-1}|z_t, z_0) &= (Multinomial_{\frac{K(z_t - (1-\beta_t) z_{t-1})}{\beta_t}}(z_{t-1}; K, \frac{1}{C}\mathbf{1}, z_t)) \\ &\cdot (Multinomial_{\frac{B_{t-1} K(z_{t-1} - \bar{\alpha}_{t-1} z_0)}{1-\bar{\alpha}_{t-1}}}(z_{t-1}; B_{t-1} K, \frac{1}{C}\mathbf{1}, z_0)) \\ &\cdot \Big(\sum_{z_{t-1}} [(Multinomial_{\frac{K(z_t - (1-\beta_t) z_{t-1})}{\beta_t}}(z_{t-1}; K, \frac{1}{C}\mathbf{1}, z_t)) \\ &\cdot (Multinomial_{\frac{B_{t-1} K(z_{t-1} - \bar{\alpha}_{t-1} z_0)}{1-\bar{\alpha}_{t-1}}}(z_{t-1}; B_{t-1} K, \frac{1}{C}\mathbf{1}, z_0))] \Big)^{-1} \end{aligned} \tag{36}$$

$$= \frac{1}{\sigma_t} (Multinomial_{\frac{K(z_t - (1-\beta_t) z_{t-1})}{\beta_t}}(z_{t-1}; K, \frac{1}{C}\mathbf{1}, z_t)) \cdot (Multinomial_{\frac{B_{t-1} K(z_{t-1} - \bar{\alpha}_{t-1} z_0)}{1-\bar{\alpha}_{t-1}}}(z_{t-1}; B_{t-1} K, \frac{1}{C}\mathbf{1}, z_0)) \tag{37}$$

where $\sigma_t = \sum_{z_{t-1}} [(Multinomial_{\frac{K(z_t - (1-\beta_t) z_{t-1})}{\beta_t}}(z_{t-1}; K, \frac{1}{C}\mathbf{1}, z_t)) \cdot (Multinomial_{\frac{B_{t-1} K(z_{t-1} - \bar{\alpha}_{t-1} z_0)}{1-\bar{\alpha}_{t-1}}}(z_{t-1}; B_{t-1} K, \frac{1}{C}\mathbf{1}, z_0))]$. We note that, the exact value of σ_t does not matter in practice since it is constant for all observed z_{t-1}, hence we fix it as a hyperparameter for better efficiency.

3.5. More discussion on diffusion models

Diffusion models [1, 3, 5, 11] have undergone rapid developments and many variants have been previously explored. For instance, in the continuous setting, there have been variants which add different types of noise in the forward diffusion process, e.g., GMM-based noise [4, 9] or noise from a Mixture-of-Cauchy distribution [13]. Some previous works also investigate discrete diffusion models [1, 11], but they explore the addition of noise sampled from a Uniform distribution. These works [1, 11] discretize the continuous pixel space (where each pixel now takes discrete values) and find that it is effective to perform diffusion on the discrete space

of pixel values. Conversely, we instead formulate a row of our image as representing probabilities for discrete states (where each pixel represents a state), and perform discrete diffusion over these discrete states (with continuous probability values) by adding Multinomial noise.

References

[1] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021. 7

[2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 2

[3] Lin Geng Foo, Hossein Rahmani, and Jun Liu. Ai-generated content (aigc) for various data modalities: A survey. *arXiv preprint arXiv:2308.14177*, 2, 2023. 7

[4] Jia Gong, Lin Geng Foo, Zhipeng Fan, Qiuhong Ke, Hossein Rahmani, and Jun Liu. Diffpose: Toward more reliable 3d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 7

[5] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 7

[6] Tianjiao Li, Lin Geng Foo, Qiuhong Ke, Hossein Rahmani, Anran Wang, Jinghua Wang, and Jun Liu. Dynamic spatio-temporal specialization learning for fine-grained action recognition. In *European Conference on Computer Vision*, pages 386–403. Springer, 2022. 2

[7] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 4

[8] Tianwei Lin, Xiao Liu, Xin Li, Errui Ding, and Shilei Wen. Bmn: Boundary-matching network for temporal action proposal generation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3889–3898, 2019. 4

[9] Eliya Nachmani, Robin San Roman, and Lior Wolf. Non gaussian denoising diffusion models. *arXiv preprint arXiv:2106.07582*, 2021. 7

[10] Dingfeng Shi, Yujie Zhong, Qiong Cao, Lin Ma, Jia Li, and Dacheng Tao. Tridet: Temporal action detection with relative boundary modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18857–18866, 2023. 1, 2

[11] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. 4, 7

[12] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018. 2

[13] Li Xu, Haoxuan Qu, Yujun Cai, and Jun Liu. 6d-diff: A keypoint diffusion framework for 6d object pose estimation. *arXiv preprint arXiv:2401.00029*, 2023. 7

[14] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In *European Conference on Computer Vision*, pages 492–510. Springer, 2022. 2