Audio-Synchronized Visual Animation Supplementary Material

Lin Zhang[1], Shentong Mo[2], Yijing Zhang[1], and Pedro Morgado[1]

[1] University of Wisconsin-Madison
[2] Carnegie Mellon University

1 Synchronization classifier details 2
1.1 Classifier architecture 2
1.2 Training 2
1.3 Evaluation 4
2 Synchronization metrics 4
2.1 Metrics for real videos in data curation 4
2.2 Metrics for generated videos 5
3 Audio-Video Latent Diffusion (AVSyncD) details 6
3.1 Diffusion schedulers 6
3.2 Model details 6
3.3 Training 7
3.4 Evaluation 8
4 Baseline Details and Comparisons 9
4.1 VideoCrafter 9
4.2 CoDi 10
4.3 TempoToken 10
4.4 TPoS 10
4.5 AADiff 11
5 AVSync15 11
5.1 Comparison of Existing Audio-Visual datasets 11
5.2 AVSync15 Curation Details 11
6 More Ablation 14
6.1 Method Comparison 14
6.2 Category-wise Results 14
6.3 Effect of Data Curation 14
6.4 Effect of First-frame (FF) Lookups 14
6.5 Effect of Text / Audio conditioning 15
6.6 Effect of Classifier-free Audio Guidance Factor 16
6.7 Effect of pretrained audio encoders 16
7 More Applications 17
7.1 Animate Generated Images 17
7.2 Animate Contents from Internet 17
7.3 Control Animations with (Un)related Audios 17
7.4 Animate Target Objects with Audios 17

Videos are included in here and mentioned in green text in sections.

1 Synchronization classifier details

1.1 Classifier architecture

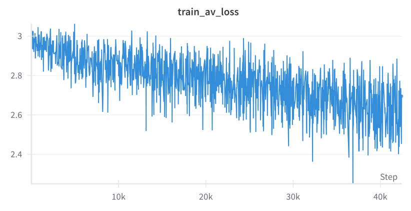
The synchronization classifier consists of an audio encoder ψ_a and a video encoder ψ_v, both following the architecture in [31] and are initialized with weights pretrained on AudioSet [15]. The input 2-second audio is converted into a mel-spectrogram of size $1\times128\times204$ with 128 and 204 frequency and time bins, respectively. The corresponding 2-second video is sampled into 12 frames of 224×224 images. After encoding and average pooling, an audio feature vector $\psi_a(\boldsymbol{a})\in\mathbb{R}^{512}$ and a video feature vector $\psi_v(\boldsymbol{v})\in\mathbb{R}^{512}$ are obtained. They are then concatenated and sent to a multi-layer perceptron of depth 3 to produce an unbounded value ϕ, which is the av-sync score defined in Sec. 3.1 of main text. Tab. 1 lists the detailed architecture. This architecture is similar to that of [8] and achieves similar performance.

Table 1: Synchronization classifier architecture.

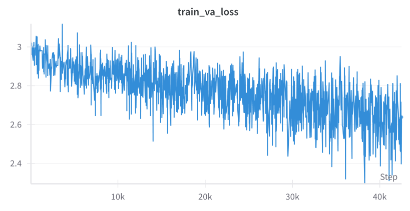
Layer Name	Audio Encoder		Video Encoder	
	Output Size	Module	Output Size	Module
Input	$1\times128\times204$	-	$3\times12\times224\times224$	-
Conv_1	$64\times64\times102$	7×7, 64, stride 2	$64\times12\times56\times56$	$\begin{bmatrix}3\times7\times7\text{,64,stride }1\times2\times2\\ \text{Max pool,}1\times3\times3\text{,stride }1\times2\times2\end{bmatrix}$
Block_1	$64\times32\times51$	$\begin{bmatrix}3\times3\text{,64,stride 2}\\3\times3\text{,64,stride 2}\end{bmatrix}\times1$	$64\times12\times56\times56$	$\begin{bmatrix}\text{R(2+1)D}(3\times3\times3\text{,64,stride }1\times1\times1)\end{bmatrix}\times2$
Block_2	$128\times16\times26$	$\begin{bmatrix}3\times3\text{,128,stride 2}\\3\times3\text{,128,stride 2}\end{bmatrix}\times1$	$128\times6\times28\times28$	$\begin{bmatrix}\text{R(2+1)D}(3\times3\times3\text{,128,stride }2\times2\times2)\\ \text{R(2+1)D}(3\times3\times3\text{,128,stride }1\times1\times1)\end{bmatrix}\times1$
Block_3	$256\times8\times13$	$\begin{bmatrix}3\times3\text{,256,stride 2}\\3\times3\text{,256,stride 2}\end{bmatrix}\times1$	$256\times3\times14\times14$	$\begin{bmatrix}\text{R(2+1)D}(3\times3\times3\text{,256,stride }2\times2\times2)\\ \text{R(2+1)D}(3\times3\times3\text{,256,stride }1\times1\times1)\end{bmatrix}\times1$
Block_4	$512\times8\times13$	$\begin{bmatrix}3\times3\text{,512,stride 1}\\3\times3\text{,512,stride 1}\end{bmatrix}\times1$	$512\times2\times7\times7$	$\begin{bmatrix}\text{R(2+1)D}(3\times3\times3\text{,512,stride }2\times2\times2)\\ \text{R(2+1)D}(3\times3\times3\text{,512,stride }1\times1\times1)\end{bmatrix}\times1$
Pool	512	average pool, 512 d vector	512	average pool, 512 d vector
Final Linear	$1024\rightarrow512\rightarrow256\rightarrow1$			

1.2 Training

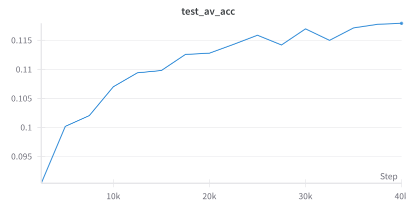
Previous works on learning audio-visual synchronization [2, 8, 25] found that applying contrastive loss between temporally shifted audio-video pairs (hard negatives) is better than using binary classification loss, and that including negative audio-video pairs from different instances (easy negatives) is detrimental to learning synchronization embeddings. They therefore conducted two stages of training. Since directly learning from hard negatives is difficult and may result in model divergence, they first apply contrastive loss from only easy negatives, serving as a warm-up. After than, they contrastively trained from only hard negatives applied to the second stage, serving as the major training stage to learn synchronization. In our experiments, we observed the same phenomenon, where training on easy and hard negatives simultaneously made training loss from hard negative examples diverges. We therefore initialized our model with weights from [31], which was pretrained as in stage-1. We then heavily trained the model using *only* temporally-shifted negatives from the same instances using contrastive loss, serving as stage-2 training.



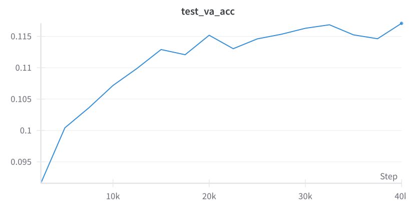
(a) a-to-v training loss.



(b) v-to-a training loss.



(c) a-to-v test accuracy.



(d) v-to-a test accuracy.

Fig. 1: Training loss and test accuracy curve of training the audio-visual synchronization classifier.

Specifically during training, for each audio-video instance, we randomly sampled 21 audio/video clips shifted by 0.2 seconds, i.e., clips at relative timestamp seconds [-2.0, -1.8, -1.6, ..., -0.4, -0.2, 0.0, 0.2, 0.4, ..., 1.6, 1.8, 2.0]. The 0.2 second gap is designed to follow [8]'s study that an audio-video pair shifted within 0.2s is typically indistinguishable by human. As such, we have 21 audios $\{\boldsymbol{a}_i\}_{i=1}^{21}$ and $\{\boldsymbol{v}_i\}_{i=1}^{21}$, where audio and videos with the same subscript are positive pairs. For each audio-video pair $(\boldsymbol{a}_i, \boldsymbol{v}_j)$, we have an av-sync score ϕ_{ij} produced by the synchronization classifier above. The models are trained to minimize the video-to-audio and audio-to-video contrastive losses defined as,

$$L_{a2v} + L_{v2a} = -\frac{1}{21}\sum_{k=1}^{21}\left(\log\frac{\exp(\phi_{kk}/\tau)}{\sum_{i=1}^{21}\exp(\phi_{ki}/\tau)} + \log\frac{\exp(\phi_{kk}/\tau)}{\sum_{i=1}^{21}\exp(\phi_{ki}/\tau)}\right) \tag{1}$$

where $\tau = 0.1$ is a temperature hyperparameter. To track performance, we also measure the audio-to-video and video-to-audio synchronization accuracy on the test set. For example, the audio-to-video test accuracy is computed from similar 21 clips spaced by 0.2s as follows

$$\text{Acc}_{a2v} = \frac{1}{21}\sum_{k=1}^{21}\mathbb{1}[k = \text{argmax}[\phi_{k1}, \phi_{k2}, ..., \phi_{kk}]] \tag{2}$$

Note that the test accuracy defined here is only used for reference during training. During testing, we adopted a different but more conventional evaluation protocol, which is described below.

We train the classifier on our VGGSS (149 classes from VGGSound [9]) for 40000 iterations until convergence, with batch size of 32, constant learning rate of 0.0002, and weight decay of 0.01. We do not use any audio augmentations and only include random horizontal flip as video augmentation. The training and testing curve is in Fig. 1.

1.3 Evaluation

During testing, we follow the evaluation pipeline of [8], which is similar to conventional synchronization classification evaluation protocol [12, 13]. Specifically, we sampled a dense set of 31 audio/video clips shifted by 0.04 seconds, i.e., clips at relative timestamp seconds [-0.6, -0.56, -0.52, ..., -0.04, 0.0, 0.04, ..., 0.52, 0.56, 0.6], with a largest gap of 1.2 seconds. We then use similar equation as Eq. (2) to compute test accuracy, except that the maximum score between two streams can be allowed to be within ± 5 frames (± 0.2s) from the ground truth, i.e., synchronisation error is not detectable by a human. In practice, our trained classifier can achieve 40.86 audio-to-video evaluation accuracy and 40.83 video-to-audio evaluation accuracy on VGGSS test set, comparable to 40.6 reported in [8] tested on 160 classes (including all 149 classes of VGGSS) on VGGSound. Moreover, we construct a test split from AVSync-AC by dropping the instances participating synchronization classifier training. Note that due to the noisy nature of original VGGSound, such a evaluation task is very challenging. As a reference, we found that the trained synchronization classifier can achieve 84.11 audio-to-video and 83.55 video-to-audio synchronization accuracy on the test split of AVSync-AC, as opposed to aforementioned 40.86 and 40.83 on VGGSS, further validating the necessity of automatic curation and the high accuracy of the trained classifier for evaluation purpose.

2 Synchronization metrics

In this section, we detail the metrics that we have used to quantify synchronization in experiments. We first describe the synchronization metrics directly derived from the trained synchronization classifier, which evaluate real videos for data curation. Unlike real videos where shifted audio-visual pairs can be easily obtained, generated videos do not have shifted pairs for reference and can semantically drift from first frame and provided audio drastically. Therefore, we further describe the synchronization metrics to evaluate generated videos.

2.1 Metrics for real videos in data curation

av-sync score (ϕ) As we have mentioned in Sec. 1.1, av-sync score ϕ is an unbounded value produced by the synchronization classifier for each pair of input (audio, video). A larger value of ϕ usually suggests higher likelihood that the audio-video is synchronized. In Fig. 2(b) in the submitted paper, we use av-sync score as a indicator, showing that our data curation pipeline gradually improves audio-video synchronization.

Synchronization probability ($\mathbf{P}_{Sync}$) While av-sync score is a straightforward indicator of synchronization, it does not use shifted audio-video pairs as a reference, which however is what our training objective in Eq. (1) does. As a result, a temporally shifted audio-video pair may receive similarly high ϕ as its synchronized (non-shifted) audio-video pair, a situation indicating strong synchronization but low synchronization cues. For example, in an ambient sounded video about raining, both audio and video do not have significant temporal changes, thus lacking synchronization cues, but the audio and video are usually determined to be highly synchronized due to semantics alignment. Therefore, in order to measure existence of clear synchronization cues, we refer to the principle of contrastive training loss to use synchronization probability P_{sync} as a metric. Specifically, Eq. (1) tries to maximize the likelihood

$$P = \frac{\exp(\phi_{kk}/\tau)}{\sum_{i=1}^{21}\exp(\phi_{ki}/\tau)} \tag{3}$$

Similarly, we approximate the synchronization probability by sampling 3 audio-video clips shifted by 0.25 seconds from middle of a video. We can then compute the audio-to-video synchronization probability ρ_{sync}^{a2v} as:

$$P_{\text{Sync}} = \frac{\exp(\phi_{2,2})}{\sum_{i=1}^{3}\exp(\phi_{2,i})} \tag{4}$$

which computes the probability to distinguish the correctly synchronized pair ($\phi_{2,2}$) from its shifted pairs. Videos with high P_{Sync} are more sensitive to the audio-video synchronization cues, thus providing better supervision once selected for training. We can then filter those videos with low P_{Sync}, e.g., the ambient sounded videos such as raining, where the shifted audio-video pair can have a similar ϕ to the non-shifted pair.

2.2 Metrics for generated videos

RelSync During testing, we have only the groundtruth synchronized audio-video pair ($\boldsymbol{a}$,$\boldsymbol{v}$), each with 2-second duration. $\boldsymbol{a}$ is input into our model as a condition to generate visual frames $\bar{\boldsymbol{v}}$. While P_{Sync} is a good metric to measure timestamp-wise synchronization cues, it cannot be computed without temporally-shifted videos. Fortunately, we still have the groundtruth video $\boldsymbol{v}$ as a reference.

Specifically, Eq. (3) and Eq. (4) maximize a likelihood to discriminate the audio-video pair (numerator) from a set of (shifted) pairs (denominator). Lacking shifted pairs, we can instead contrast the generated audio-video pair ($\boldsymbol{a}$,$\bar{\boldsymbol{v}}$) with the groundtruth pair ($\boldsymbol{a}$,$\boldsymbol{v}$):

$$\text{RelSync}(\boldsymbol{a},\hat{\boldsymbol{v}},\boldsymbol{v}) = \frac{\exp(\phi_{\boldsymbol{a},\hat{\boldsymbol{v}}})}{\exp(\phi_{\boldsymbol{a},\boldsymbol{v}})+\exp(\phi_{\boldsymbol{a},\hat{\boldsymbol{v}}})} \tag{5}$$

RelSync evalutes the synchornization between the generated video $\hat{\boldsymbol{v}}$ and the input audio condition $\boldsymbol{a}$, by referring to the groundtruth audio-video pair ($\boldsymbol{a}$,$\boldsymbol{v}$). Note that, since RelSync uses the groundtruth audio-video pair as a reference, the reference pair is better to have highly synchronization cues.

It should also be noted that RelSync only measures temporal synchronization rather than visual quality and audio-visual semantics alignment of generated video. So we further use FID, FVD, and IA-Align as metrics.

AlignSync As mentioned above, RelSync only measures audio-video synchronization rather than audio-video semantics alignment. In fact, as mentioned in Sec. 1.2, the synchronization classifier is only trained on temporally-shifted audio-video pairs sampled from the same instance, which are mostly highly aligned in semantics. RelSync is thus implicitly conditioned on semantics alignment, approximating P(Sync|Align). To accommodate the need to measure both audio-video semantics alignment and synchronization, we further compute score to approximate P(Align), and multiply it with RelSync, ending up with AlignSync to measure both semantic alignment and temporal synchronization.

For a given audio-video pair ($\boldsymbol{a}$,$\boldsymbol{v}$), we first derive the score to measure its audio-video semantics alignment. While there exists some metrics such as IA computed using ImageBind [16] (see Sec. 3.4), they are too empirical to be used as a probability approximator.

However, since ImageBind is also trained contrastively like our synchronization classifier, we can approximate P(Align) in a similar way as approximating P(Sync|Align) using RelSync. Specifically, ImageBind imitates CLIP [32] to learn alignment between normalized image representations $\boldsymbol{q}$ and normalized audio representation $\boldsymbol{k}$ using contrastive loss:

$$L_{\text{ImageBind}} = -\log \frac{\exp(\boldsymbol{q}_i^T \boldsymbol{k}_i/\tau)}{\exp(\boldsymbol{q}_i^T \boldsymbol{k}_i/\tau) + \sum_{i \neq j} \exp(\boldsymbol{q}_i^T \boldsymbol{k}_j/\tau)} \tag{6}$$

where the subscript i,j indicates different (image, audio) instance pairs. Given an audio $\boldsymbol{a}$ and the first frame $\boldsymbol{v}_1$ for conditioning, we generate a video with b frames $\hat{\boldsymbol{v}} = \{\hat{\boldsymbol{v}}_1, ..., \hat{\boldsymbol{v}}_b\}$. Similar to Eq. (5), by treating the groundtruth frame $\boldsymbol{v}_1$ as a reference, we can approximate semantic alignment probability between a generated image $\hat{\boldsymbol{v}}_i$ and $\boldsymbol{a}$ as:

$$P_{\text{Align}}(\boldsymbol{a}, \hat{\boldsymbol{v}}_i, \boldsymbol{v}_1) = \frac{\exp(\boldsymbol{q}_{\boldsymbol{a}}^T \boldsymbol{k}_{\hat{\boldsymbol{v}}_i})}{\exp(\boldsymbol{q}_{\boldsymbol{a}}^T \boldsymbol{k}_{\hat{\boldsymbol{v}}_i}) + \exp(\boldsymbol{q}_{\boldsymbol{a}}^T \boldsymbol{k}_{\boldsymbol{v}_1})} \tag{7}$$

$$= \frac{\exp(\text{IA}_{\boldsymbol{a}, \hat{\boldsymbol{v}}_i})}{\exp(\text{IA}_{\boldsymbol{a}, \hat{\boldsymbol{v}}_i}) + \exp(\text{IA}_{\boldsymbol{a}, \boldsymbol{v}_1})} \tag{8}$$

where IA is the dot product between normalized audio and video representations, as seen in Sec. 3.4. We can then average the score among all generated frames, obtaining the alignment score:

$$P_{\text{Align}}(\boldsymbol{a}, \hat{\boldsymbol{v}}, \boldsymbol{v}) = \frac{1}{b-1} \sum_{i=2}^{b} P_{\text{Align}}(\boldsymbol{a}, \hat{\boldsymbol{v}}_i, \boldsymbol{v}_1) \tag{9}$$

where we only average among the generated frames $\hat{\boldsymbol{v}}_{2...b}$, excluding the generated first frame, which is usually a replicate of input groundtruth frame $\boldsymbol{v}_1$. As such, we can multiply the semantics alignment probability with RelSync, obtaining our AlignSync metric:

$$\text{AlignSync}(\boldsymbol{a}, \hat{\boldsymbol{v}}, \boldsymbol{v}) = P(\text{Sync}|\text{Align}) \cdot P(\text{Align}) \tag{10}$$

$$= \text{RelSync}(\boldsymbol{a}, \hat{\boldsymbol{v}}, \boldsymbol{v}) \cdot P_{\text{Align}}(\boldsymbol{a}, \hat{\boldsymbol{v}}, \boldsymbol{v}) \tag{11}$$

Based on RelSync, AlignSync also prefers a groundtruth audio-video reference pair with clear synchronization cues.

3 Audio-Video Latent Diffusion (AVSyncD) details

3.1 Diffusion schedulers

We use exactly the same diffusion samplers as in StableDiffusion-V1.5, i.e., DDPM [20] scheduler during training and PNDM scheduler [29] during inference.

3.2 Model details

Model parameters We include a table to record the number of parameters in every module of AVSyncD in Tab. 2. We use StableDiffusion-V1.5 as the pretrained image diffusion generation model and freeze all of its original parameters, including its associated text encoder. We also use the frozen pretrained ImageBind as the audio encoder.

Table 2: AVSyncD parameters.

Module Name	Trainable	Num Param(M)
Text Encoder	✗	340
ImageBind Audio Encoder	✗	86
Stable Diffusion UNet2D	✗	870
FF Temporal Convolution	✓	191
Temporal Attention	✓	75
Audio Cross Attention	✓	44

Attention formulation Formally, we use conventional multi-head attention that updates a feature $\boldsymbol{z}$ by attending to a series of key tokens $\{\boldsymbol{x}_k : k \in \mathcal{K}\}$

$$\boldsymbol{z} \leftarrow \boldsymbol{z} + \sum_{h=1}^{H} \sum_{k \in \mathcal{K}} SoftMax_k \left(\boldsymbol{z}^T \boldsymbol{Q}_h^T \boldsymbol{K}_h \boldsymbol{x}_k \right) \boldsymbol{V}_h \boldsymbol{x}_k \tag{12}$$

where $\boldsymbol{Q}_h$, $\boldsymbol{K}_h$ and $\boldsymbol{V}_h$ are the query, key and value projection matrices for the H attention heads. All attention layers differ only by how the set $\mathcal{K}$ is defined, as visualized in Fig. 2.

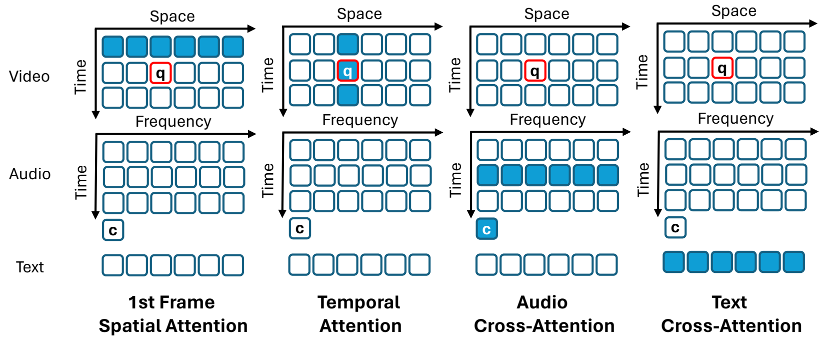


Fig. 2: For a query token q, we highlight the set $\mathcal{K}$ of key and value tokens used in different attentions. $\boldsymbol{c}$ refers to audio classification token.

3.3 Training

During training, we randomly sample 2-second audio-video clip in a video instance. Audio clip preprocess is the same as in training synchronization classifier, i.e., encoding an audio clip into mel-spectrogram of shape $1 \times 128 \times 204$ [16]. For video clip, we sample it in 6 FPS to obtain 12 (b) frames. The images are resized with their original aspect ratio so that the shortest side is 256, and then center cropped to 256×256 on AVSyncD and Landscapes, and 128×256 on The Greatest Hits. We use random horizontal flip on video frames as the only data augmentation. We randomly replace input audio with null audio representations as classifier-free audio guidance. We do not use classifier-free text guidance, i.e., always input audio category as text condition.

3.4 Evaluation

For each test instance, we uniformly sample 3 clips for evaluation, i.e., 2-s clips from the head, center, and the tail. Suppose there are n test clips in total, this then gives $3\times n$ groundtruth testing clips and $3\times n$ generated clips to evaluate. When evaluating generation as frames, we only evaluate future generated frames, excluding the first frame which is usually a replicate of input initial frame. When evaluating generation as a video, we however measure on all frames as a whole. We detail the metrics we have used in the following:

FID Fréchet Inception Distance (FID) [19] measures the distribution distance between two sets with equal number of images. It is a conventional image generation quality metric, highly aligned with human perception. We encode all future frames of the groundtruth clips ($\boldsymbol{v}_{2\ldots b}$) and the generated clips ($\hat{\boldsymbol{v}}_{2\ldots b}$) into image feature vectors using pretrained InceptionV3 model [11], and compute the frechet distance between the $n\times(b-1)$ groundtruth clips' feature vectors and the $n\times(b-1)$ generated clips' feature vectors, where n is the total number of testing clips.

FVD Fréchet Video Distance (FVD) [36] mimics FID but measures distribution distances between *video* features instead of *image* features. We encode *all* (including the first frame) groundtruth clips and generated clips into video feature vectors using pretrained I3D model [23], and compute the frechet distance between the n groundtruth clips' video feature vectors and the n generated clips' video feature vectors.

Image-Text CLIP Alignment (IT) We follow the convention to use Image-Text CLIP Alignment to measure the semantics alignment between image and text in the CLIP feature space. Specifically, given a text prompt $\boldsymbol{\tau}$ and a video clip with b frames $\boldsymbol{v}_{1\ldots b}$, the CLIP text encoder encodes the text prompt $\boldsymbol{\tau}$ into a normalized feature vector $f_{\boldsymbol{\tau}}$, and the CLIP image encoder independently encodes the frames into normalized feature vectors $\{f_{\boldsymbol{v}_k}\}_{k=1}^{b}$. It then computes the average cosine similarity between them to measure semantics alignment between the text and the video:

$$\mathrm{IT}(\boldsymbol{\tau},\boldsymbol{v})=\frac{1}{b}\sum_{k=1}^{b}\frac{||f_{\boldsymbol{\tau}}\cdot f_{\boldsymbol{v}_k}||}{||f_{\boldsymbol{\tau}}||\cdot||f_{\boldsymbol{v}_k}||} \tag{13}$$

As aforementioned, when using IT to measure generated videos, we exclude the first frame, i.e., average among future frames $2\ldots b$.

Image-Audio CLIP Alignment (IA) ImageBind [16] extends CLIP by incorporating the audio modality, thus can be used to measure Image-Audio CLIP Alignment (IA). Specifically, For an (audio, video) pair $(\boldsymbol{a},\boldsymbol{v})$, we first use its audio encoder encode the audio into a feature vector $f_{\boldsymbol{a}}$, then encode video frames into features $\{f_{\boldsymbol{v}_k}\}_{k=1}^{b}$ as we have done for IT. We then average cosine similarity between them:

$$\mathrm{IA}(\boldsymbol{a},\boldsymbol{v}) = \frac{1}{b}\sum_{k=1}^{b}\frac{\|f_{\boldsymbol{a}} \cdot f_{\boldsymbol{v}_k}\|}{\|f_{\boldsymbol{a}}\| \cdot \|f_{\boldsymbol{v}_k}\|} \tag{14}$$

Similarly as IT, whne evaluating generated videos, we exclude the first frame which is usually replicates the groundtruth input frame, and average among future frames $2...b$ instead.

User study We have described our user study in detail in Sec. 5.2 in submitted paper. We evaluate all 150 test examples in AVSync15. For each test video example, we retrieve the groundtruth audio-video pair from center of the video for generation.

Diff-Foley metric Diff-Foley [28] proposed a synchronization classifier specially trained for evaluation. However, due to lack of training details, we cannot replicate evaluation classifier. Instead, we tried our best to use their released checkpoints to evaluate on generated videos by repeatedly padding audios and videos to be 8.2-second long. Results shown in Tab. 3.

Table 3: Evaluation on AVSync15 with Diff-Foley metric. We repeatedly pad 2-second audio and video features to be 8.2-second.

GT	I+T+A								I+T		T+A	
	AVSyncD					**AADiff**	**TPoS**	**CoDi**	**I2VD**	**VideoCrafter**	**TempoToken**	**TPoS**
	$\eta=1$	$\eta=2$	$\eta=4$	$\eta=8$	$\eta=12$							
78.89	72.22	72.66	73.78	75.78	77.78	81.56	28.44	58.89	69.11	67.33	41.33	56.44

4 Baseline Details and Comparisons

4.1 VideoCrafter

VideoCrafter [7] is a text-to-video or image-to-video diffusion model. It accepts both image and text conditions, encoding them into CLIP space, and infused into a denoising UNet for conditioning. It can also generate videos at different FPS. The model is trained on high-quality large scale image and video datasets, namely LAION COCO [1] with 600M text-image pairs and WebVid-10M [4] with 10M text-video pairs. We use the VideoCrafter's pretrained image-to-video animation model with image size 320×512 on github. To solve the image size mismatch, we first rescale input images from 256x256 to 320x320, then pad the images horizontally to have width 512 (96 pixels on the left and 96 on the right), with its average pixel value. After generation, we remove the padded portion and rescale the center crop from 320x320 to 256x256. We also input FPS=6 to generate 12 frames in 6 fps. In practice, we found it has difficulty replicating the input groundtruth image. However, it can still generate promising video dynamics. Specifically, on the synchronization metrics, we validated that the input audio did bring significant advantages in our AVSyncD, as opposed to no audio input in VideoCrafter.

4.2 CoDi

CoDi [34] is an any-to-any generation model. It trains multiple modality encoders to produce aligned features in the CLIP space. When receiving inputs from multiple modalities, it first encodes them into features, then performs weighted sum among these features. The final feature vector is then fused into a video diffusion model through cross-attention layers. The model is trained on many large-scale video datasets including WebVid-10M [4], HD-Villa-100M [38], and many large-scale audio datasets including AudioSet [15], AudioCaps [24], FreeSound 500K [34], and SoundNet [3]. To suit the need of its input setting, we zero pad the input 2-second audios on the right to 10-second, and resize our input images from 256×256 to 224×224. We fuse the input image, text, and audio features with equal weights, generating videos with 8 frames for evaluation. CoDi performed even worse than VideoCrafter in faithfully reconstructing the input image, and always generating videos with objects vague in semantics. Due to its inferior encoding method which loses the temporal cues of audio, it also performs poorly on audio-video synchronization.

4.3 TempoToken

TempoToken [39] encodes 2-second audio into time-aware segments with BEATS [10], an audio encoder trained on full set of AudioSet-2M in self-supervised learning manner. It then learns a carefully designed mapper to project the encoded audio temporal segments onto another space for condition. The encoded segment features are then sent to a pretrained text-to-video diffusion model ModelScope [37], which is trained on 10M text-video pairs and 2B text-image pairs, to generate videos. With both 2-second audio and text as inputs, we use TempoToken's officially released checkpoints trained on VGGSound and Landscapes, producing 6 fps (12 frames) videos in size 384×384 and then resized to 256×256, for evaluation. While TempoToken by design cannot animate an image, we found on our targeted synchronization performance, it still showed limited improvement upon other baselines, and often produced video contents misaligned with input text and audio in semantics.

4.4 TPoS

TPoS [22] learns an audio encoder to produce segmented time-aware audio features, with each audio segmented feature aligned in CLIP space. The audio encoder is composed of a pretrained ResNet [17], LSTM [21] blocks, and temporal attention blocks. Each segmented audio feature is then supervised to be aligned with text features and the other audio segmented features within the same audio instance, through carefully-designed audio-audio and audio-image contrastive learning. The learned segmented audio features, which can be interpolated along the time dimension to produce more frames following their proposed technique, are then sent to pretrained text-to-image generator StableDiffusion [33] to produce videos under the SEGA [5] framework. Besides receiving text and audio to achieve audio-to-video generation, TPoS also supports receiving additional image to animate images guided by text and audio. Original TPoS receives 2-second audio input for each frame, producing 5 frames. To produce 12 frames conditioned on 2-second audio, we first split the audio uniformly into 5-segments, with each audio segment repeatedly padded to 2-second, giving 5 encoded audio segmented features. The segmented features are then interpolated into 12 frames using their proposed interpolation method to produce 12 frames. Images are always encoded and generated with size 512×512, and then resized to 256×256 after generation. In practice, we use TPoS for both audio-to-video generation and audio-guided image animation. In both cases, we found the frames produced by TPoS often lacking meaningful variations, which follows their produced results on official webpage `https://ku-vai.github.io/TPoS/`.

4.5 AADiff

AADiff [27] is a training-free framework for audio-to-image generation and audio-guided image animation. With a text and an audio, it first encodes both into features using CLAP, a text-audio aligned encoder trained on text-audio pairs contrastively. It then locate the word with highest feature similarity wit the audio feature, then reweights the text-image cross attention map corresponding to this word using Prompt2Prompt technique [18], where the weights comes from the audio amplitude in local window. By using null-text-inversion, AADiff can also achieve audio-guided image animation. In practice, we directly feed the audio category as the conditioning prompt. The audio amplitude of each frame is the average of the corresponding audio amplitude within its $\frac{1}{6}$-seconds local window. This gives us 12 audio amplitudes $\{s_i\}_{i=1}^{12}$ corresponding to 12 frames. With the first frame corresponding to audio segments with compute amplitude s_1 and attention weights 1.0, the following frames have attention weights $r_i = \frac{s_i}{s_1} \times 1.0$. Note that due to AADiff's inherent disadvantages of (1) relying on audio amplitude instead of more fine-grained audio segmented features for conditioning and (2) relying on image style editing method Prompt2Prompt, it naturally cannot generate meaningful and reliable video dynamics, as seen in our qualitative comparisons in the submitted paper.

5 AVSync15

5.1 Comparison of Existing Audio-Visual datasets

We include a comparison of existing audio-visual datasets in Tab. 4, comparing some attributes particularly related to learning synchronized audio-video events generation, such as if the audios are ambient, the camera viewpoint (thus the frame contents) are stable, or the dynamics demonstrated in both audio and video are sychronized. As can be seen, most of existing audio-visual datasets are not appropriate for audio-synchronized video dynamics generation tasks, falling short in these 3 attributes or class diversity. The proposed AVSync15 is the first benchmark to get rid of all these defects to be used for audio-synchronized visual dynamics generation task. We provide video samples for comparison in folder suppl-videos/5.1 - Dataset Comparison.

Besides, we also measured synchronization cues of groundtruth videos in each dataset using metric in Eq. (4). Instead of producing 3 av pairs shifted by 0.25 seconds from each video, we uniformly sample 3 av pairs from the beginning, center, and end of each video. A fully ambient video without any changes in audio and video thus have a P_{Sync} score of $\frac{1}{3}(random)$. Higher score indicates higher synchronization clue in the dataset. Results in Tab. 5 showed our curated AVSync15 has more synchronization cues than any others.

5.2 AVSync15 Curation Details

We have described our data curation pipeline in submitted paper Sec. 3.1. We further detail the procedures in the following, including categories, metric threshold, etc.

Table 4: Comparison of existing audio-visual datasets

Dataset	Num Classes	Num Video	Ambient Sound	Stable camera viewpoint	Dynamics Synchronized
AudioSet [15]	632	2M	Partial	Partial	Partial
VGGSound [9]	309	200K	Partial	Partial	Partial
AIST++ [35]	60 songs	1K	No	Yes	No
Landscapes [26]	9	1K	Most	Partial	Partial
The Greatest Hits [30]	1	977	No	Yes	Yes
AVSync15	15	1.5K	No	Yes	Yes

Table 5: Evaluation of P_{Sync} on datasets.

AVSync15	**AVSync-AC**	**VGGSS**	**TheGreatestHits**	**Landscapes**	**Random**
39.50	38.67	36.01	35.21	33.40	33.33

Preliminary Curation In preliminary curation, we narrow down from 176,620 videos in 309 classes in original VGGSound to 86,242 video in 149 classes with potentially clear audio-video synchronization cues. The dataset is named as VGGSS with the following classes: alligators crocodiles hissing, baby babbling, baby crying, baltimore oriole calling, bee wasp etc. buzzing, bird chirping tweeting, bird squawking, bird wings flapping, bowling impact, bull bellowing, cap gun shooting, car passing by, cat caterwauling, cat growling, cat hissing, cat meowing, cat purring, cattle bovinae cowbell, chainsawing trees, cheetah chirrup, chicken clucking, chicken crowing, child singing, child speech kid speaking, children shouting, chipmunk chirping, chopping food, crow cawing, cutting hair with electric trimmers, dinosaurs bellowing, dog barking, dog bow-wow, dog growling, dog whimpering, door slamming, driving motorcycle, duck quacking, eagle screaming, electric grinder grinding, elephant trumpeting, eletric blender running, female singing, female speech woman speaking, firing cannon, firing muskets, forging swords, fox barking, francolin calling, frog croaking, goat bleating, golf driving, goose honking, hammering nails, helicopter, horse clip-clop, horse neighing, lions growling, lions roaring, lip smacking, machine gun shooting, male singing, male speech man speaking, missile launch, mouse clicking, mouse pattering, mouse squeaking, opening or closing car doors, opening or closing car electric windows, otter growling, owl hooting, parrot talking, penguins braying, people babbling, people belly laughing, people burping, people cheering, people clapping, people coughing, people eating apple, people eating crisps, people finger snapping, people gargling, people giggling, people nose blowing, people running, people screaming, people shuffling, people slapping, people slurping, people sneezing, people sniggering, people sobbing, people whispering, people whistling, pigeon dove cooing, plastic bottle crushing, playing accordion, playing acoustic guitar, playing banjo, playing bass drum, playing bass guitar, playing cello, playing drum kit, playing electric guitar, playing electronic organ, playing flute, playing french horn, playing hammond organ, playing harmonica, playing harp, playing harpsichord, playing oboe, playing piano, playing snare drum, playing steel guitar slide guitar, playing table tennis, playing tennis, playing trombone, playing trumpet, playing violin fiddle, playing volleyball, popping popcorn, race car auto racing, rapping, ripping paper, rowboat canoe kayak rowing, running electric fan, sea lion barking, sharpen knife, shot football, skateboarding, skidding, skiing, sliding door, snake hissing, snake rattling, striking bowling, swimming, tapping guitar, toilet flushing, tractor digging, train horning, train whistling, turkey gobbling, typing on computer keyboard, typing on typewriter, using sewing machines, wind chime, woodpecker pecking tree.

Automatic Curation Many videos in VGGSS contain shot changes, with frames converting from one to the other sharply. We thus use PySceneDetect [6] to detect these shot changes and split them further, resulting in 105,204 videos. The resulted videos, however, are still not free from shot changes and contain lots of noises such as unstable camera viewpoints and ambient sound, etc., in need of further cleaning. If some moments in a video are found to be problematic, discarding the whole video would result in waste of the other valid moments. Therefore, to maximize the video utilization, we split each video into 3-second clips with 0.5-second strides, resulting in 1,102,158 3-second overlapping short clips, we then choose whether to discard each short clip or not based on the following metrics.

- *Raw Pixel Differences*. For each clip, we uniformly sample 10 frames $\{f_i\}_1^{10}$. We measure distance between a sampled frame f_i and its previous sampled frame f_{i-1} via their average absolute differences in RGB space, i.e., mean($\|f_i - f_{i-1}\|$), resulting in 9 average difference values in the RGB space within scale range [0.0,1.0]. We then further average these 9 values, producing an average raw pixel difference value for each clip. We discard clips with whose raw pixel difference value less than 0.01 or larger than 0.15, corresponding to videos potentially demonstrating static contents or too sharp motions.
- *Image Semantics Differences*. The previous *Raw Pixel Differences* measures distance only in the RGB raw pixel space. To complement it in the semantic space and further complement PySceneDetect to detect shot changes, we encode each frame into CLIP image feature vectors, and measures distance as L2 distances between CLIP image feature vectors instead. After obtaining 9 distance values, we choose the maximum value among them instead of average as above, attempting to find out the shot changes. We discard clips with a value less than 0.17 or larger than 0.55.
- *Audio Waveform Amplitude*. To remove clips with imperceivable audios, we read the audios as waveforms, and remove those clips whose maximum waveform amplitude is smaller than 0.03.
- *Semantic Alignment*. To ensure semantics alignment between video frames, audios, and audio categories, we use ImageBind [16] to compute IA and IT (see Sec. 3.4 for definition) for filtering. Specifically, we uniformly sample 3 frames in each clip, along with corresponding 3 2-second audios centered around the frames. We obtain text features, image features, and audio features by using ImageBind to encode audio catetory, the 3 images, and the 3 audios. Then, we can obtain 3 IA scores and 3 IT scores for each clip, where we choose the minimum score among the 3 to represent each clip's IA and IT. We then discard those clips whose IA is less than 0.15 or IT is less than 0.22.
- *Audio-Video Synchronization*. As mentioned in submitted paper, we also propose to use synchronization probability P_{Sync} as detailed in Sec. 2.1 to filter out clips with low synchronization cues. For each clip, we sample a 2-second audio and corresponding 12 frames from the middle, and create temporally shifted pairs by shifting the audio by 0.25 seconds or shifting the video by 0.25 seconds, resulting in two synchronization probabilities, i.e., audio-to-video sync or video-to-audio sync. We then discard those clips with either value less than 0.341.

After the above automatic curation, 100,812 3-second short clips still remain, i.e., 9.15% of original 1,102,158 short clips. We then merge the some clips that are split from the save video and overlapped. Finally, we obtain 42,882 videos with duration longer than 3-second.

Manual Curation To ensure the quality of collected dataset as a benchmark to evaluate synchronization, we further apply manual curation based on AVSync-AC. We first check the quality of each class generally and select 15 classes with potentially clear and human perceivable audio-video synchronization cues. Then for each video, we manually check its quality to ensure that (1) it is free from unstable camera motions (2) every significant motion/frame content changes demonstrated in the video well corresponds to the audio, and vise versa (3) the audio and video contents are well aligned in semantics. When necessary, we also manually extract the valid short clips from the video. After all of these, we obtain our class-balanced set of videos, AVSync15, with 100 videos in each class and each video longer than 2-second. We split each class into 90 traininig videos and 10 testing videos. We have manually checked each video to avoid existence of offensive content. Note that AVSync15 only constitutes a small portion of AVSync-AC, since we want to make sure that the final dataset can serve as a high-quality benchmark for audio-video synchronized generation and that the video distribution is balanced among diverse classes. We provide some video samples of AVSync15 in folder suppl-videos/5.2 - AVSync15 Samples.

6 More Ablation

6.1 Method Comparison

We provide more videos to compare our AVSyncD with baselines on AVSync15 and Landscapes, as well as comparison with I2VD on TheGreatestHits, in the included .zip folder suppl-videos/6.1 - Method Comparison.

6.2 Category-wise Results

We provide category-wise evaluation results in Tab. 6. We provide sounded videos in each category in the included .zip folder suppl-videos/6.1 - Method Comparison.

6.3 Effect of Data Curation

We include the full quantitative comparison in Tab. 2(b) of submitted paper in Tab. 7. We also include qualitative comparison of models trained on subsets of VGGSS, AVSync-AC, and our AVSync15, in folder suppl-videos/6.3 - Effect Data Curation.

6.4 Effect of First-frame (FF) Lookups

We include the full quantitative comparison in Tab. 2(a) of submitted paper in Tab. 8. We also include qualitative comparison of models trained with different modules, in folder suppl-videos/6.4 - Effect FF Lookups.

Table 6: Category-wise evaluation results for AVSyncD with classifier-free audio guidance factor 4.0 on AVSync15 test set.

Category	**FID↓**	**FVD↓**	**IA↑**	**IT↑**	**AlignSync↑**	**RelSync↑**
baby babbling crying	82.7	1143.9	39.14	29.53	24.76	49.27
cap gun shooting	84.2	1043.7	31.49	29.17	22.55	45.22
chicken crowing	82.6	1185.2	44.04	30.33	23.35	46.54
dog barking	84.7	1342.4	38.45	27.05	21.70	43.50
frog croaking	88.8	1252.9	43.11	31.09	24.04	48.15
hammering	93.7	1353.2	31.94	27.48	20.97	42.32
lions roaring	84.7	1030.8	47.08	29.05	23.59	47.02
machine gun shooting	75.5	1367.3	44.61	32.32	21.21	42.30
playing cello	77.9	1380.1	42.18	31.24	22.17	45.10
playing trombone	83.7	956.1	39.36	33.18	20.94	43.01
playing trumpet	76.2	1044.0	35.15	31.08	21.75	43.96
playing violin fiddle	78.1	1313.4	39.18	30.88	22.67	45.74
sharpen knife	95.4	1465.6	35.19	29.56	22.37	45.10
striking bowling	91.0	1587.3	25.47	32.48	23.66	47.77
toilet flushing	99.0	1484.5	41.61	32.35	23.62	47.87

Table 7: Full results on effect of data curation in Tab. 2(b) in submitted paper.

Dataset	**AC**	**MC**	**FID↓**	**IA↑**	**IT↑**	**FVD↓**	**AlignSync↑**	**RelSync↑**
VGGSS	✗	✗	12.9	29.17	25.96	1307.9	21.50	45.37
AVSync-AC	✓	✗	12.0	38.13	30.18	428.8	22.09	44.53
AVSync15	✓	✓	11.8	38.52	30.52	325.6	22.33	44.94

Table 8: Full results on effect of FF lookups in Tab. 2(a) in submitted paper.

FF-Conv	**FF-Attn**	**FID↓**	**IA↑**	**IT↑**	**FVD↓**	**AlignSync↑**	**RelSync↑**
✗	✗	11.8	38.35	30.35	383.3	22.19	44.69
✓	✗	11.6	38.34	30.28	347.4	22.24	44.79
✓	✓	11.8	38.52	30.52	325.6	22.33	44.94

Table 9: Results on effect of text / audio conditioning at test time.

Dataset	**Text**	**Audio**	**FID↓**	**IA↑**	**IT↑**	**FVD↓**	**AlignSync↑**	**RelSync↑**
AVSync15	✓	✗	12.4	38.08	30.27	463.6	21.91	44.19
	✗	✓	11.7	38.40	30.26	369.4	22.55	45.41
	✓	✓	11.7	38.53	30.45	349.1	22.62	45.52
Landscapes	✓	✗	16.7	21.99	22.56	520.4	24.80	50.01
	✗	✓	16.2	22.25	22.66	419.5	24.77	49.89
	✓	✓	16.2	22.49	22.79	415.2	24.82	49.93
TheGreatestHits	✓	✗	9.0	11.93	13.20	343.0	22.39	45.10
	✗	✓	8.7	12.07	13.30	249.6	22.83	45.95
	✓	✓	8.7	12.07	13.31	249.3	22.83	45.95

6.5 Effect of Text / Audio conditioning

As mentioned in L296 in the submitted paper, we also discuss the importance of text / audio conditioning. Based on the same trained AVSyncD, we evaluate the results without text or audio condition at test time, results in Tab. 9. As can be seen, the input text only brings insignificant improvement, while the audio plays a much more important role.

6.6 Effect of Classifier-free Audio Guidance Factor

We also include more results on the effect of classifier-free audio guidance factor η on the three datasets in Tab. 10. As can be seen, higher η leads to higher AlignSync and RelSync, but too low or too high η can both hurt generated visual quality. The rows with η marked as invalid show the results of AVSyncD trained without classifier-free audio guidance, i.e., pure audio-conditional generation without controllability. We provide videos for examples in Fig. 5(a) in submitted paper, in folder suppl-videos/6.6 - Effect Audio Guidance

Table 10: Results on effect of classifier-free audio guidance factor η.

Dataset	η	FID↓	IA↑	IT↑	FVD↓	AlignSync↑	RelSync↑
AVSync15	-	11.8	38.52	30.52	325.6	22.33	44.94
	1	12.1	38.36	30.34	382.7	22.25	44.81
	2	11.9	38.51	30.41	343.3	22.42	45.12
	4	11.7	38.53	30.45	349.1	22.62	45.52
	8	11.7	37.99	30.27	420.7	22.74	45.88
	12	11.9	37.08	29.86	569.2	22.75	46.11
Landscapes	1	16.5	22.29	22.81	463.1	24.81	49.96
	2	16.4	22.42	22.87	430.6	24.79	49.89
	4	16.2	22.49	22.79	415.2	24.82	49.93
	8	16.3	22.37	22.78	489.7	24.84	50.00
TheGreatestHits	1	9.0	11.85	13.18	313.5	22.59	45.52
	2	8.9	11.96	13.18	282.7	22.69	45.70
	4	8.7	12.07	13.31	249.3	22.83	45.95
	8	8.6	12.04	13.63	254.2	23.04	46.37

6.7 Effect of pretrained audio encoders

We also tried replacing ImageBind with pretrained CLAP [14] and BEATs [10] in Tab. 11a. While BEATs performs comparable to ImageBind on our task, CLAP shows worse performance. This might be due to CLAP's low audio encoding density along the temporal axis, e.g, the output encoded audio tokens' temporal dimension is 32 for 10-second input audio(repeatedly padded from 2-second), which can only encode audio information at 3.2 FPS, much lower than our 6 FPS output video frames.

Table 11: Results by using different backbones.

backbone	FID↓	FVD↓	IA↑	IT↑	AignSync↑	RelSync↑
CLAP	12.1	444.4	38.33	30.32	21.88	44.06
BEATs	12.0	384.4	38.39	30.33	22.22	44.74
ImageBind	12.1	382.7	38.36	30.34	22.25	44.81

(a) Effect of encoders on AVSync15 w/ $\eta = 1$.

7 More Applications

Following Sec. 5.4 in the submitted paper, we provide videos for more intuitive visualizations in the .zip folder, as described below.

7.1 Animate Generated Images

When lacking an image as input, we can use existing image generators to generate the image, which AVSyncD can also animate. Videos included folder suppl-videos/7.1 - Animate Generated Images.

7.2 Animate Contents from Internet

AVSyncD can also generalize to unseen images and audios from internet. Videos included in folder suppl-videos/7.2 - Animate with Real Images Audios.

7.3 Control Animations with (Un)related Audios

It is possible to control the same object using different audios, when the audio is related. Videos included in folder suppl-videos/7.3 - Related Control.

7.4 Animate Target Objects with Audios

When multiple objects are available, it is possible to control the object of interest via related audio. Videos included in folder suppl-videos/7.4 - Animate Target Objects.

References

1. Laion-coco. `https://laion.ai/blog/laion-coco/`
2. Afouras, T., Owens, A., Chung, J.S., Zisserman, A.: Self-supervised learning of audio-visual objects from video. In: ECCV (2020)
3. Aytar, Y., Vondrick, C., Torralba, A.: Soundnet: Learning sound representations from unlabeled video. In: Advances in Neural Information Processing Systems (2016)
4. Bain, M., Nagrani, A., Varol, G., Zisserman, A.: Frozen in time: A joint video and image encoder for end-to-end retrieval. In: IEEE International Conference on Computer Vision (2021)
5. Brack, M., Schramowski, P., Friedrich, F., Hintersdorf, D., Kersting, K.: The stable artist: Steering semantics in diffusion latent space (2023)
6. Castellano, B.: Pyscenedetect. `https://www.scenedetect.com/`
7. Chen, H., Xia, M., He, Y., Zhang, Y., Cun, X., Yang, S., Xing, J., Liu, Y., Chen, Q., Wang, X., Weng, C., Shan, Y.: Videocrafter1: Open diffusion models for high-quality video generation (2023)
8. Chen, H., Xie, W., Afouras, T., Nagrani, A., Vedaldi, A., Zisserman, A.: Audio-visual synchronization in the wild. In: Proceedings of the British Machine Vision Conference (BMVC) (2021)
9. Chen, H., Xie, W., Vedaldi, A., Zisserman, A.: Vggsound: A large-scale audio-visual dataset. In: ICASSP (2020)
10. Chen, S., Wu, Y., Wang, C., Liu, S., Tompkins, D., Chen, Z., Wei, F.: Beats: Audio pre-training with acoustic tokenizers. In: ICML (2023)

11. Christian, S., Wei, L., Yangqing, J., Pierre, S., Scott, R., Dragomir, A., Dumitru, E., Vincent, V., Andrew, R.: Going deeper with convolutions. In: CVPR (2015)
12. Chung, J.S., Zisserman, A.: Out of time: automated lip sync in the wild. In: ACCV Workshop (2016)
13. Chung, S.W., Chung, J.S., Kang, H.G.: Perfect match: Improved cross-modal embeddings for audio-visual synchronisation. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2019)
14. Elizalde, B., Deshmukh, S., Al Ismail, M., Wang, H.: Clap learning audio concepts from natural language supervision. In: ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2023)
15. Gemmeke, J.F., Ellis, D.P.W., Freedman, D., Jansen, A., Lawrence, W., Moore, R.C., Plakal, M., Ritter, M.: Audio set: An ontology and human-labeled dataset for audio events. In: Proc. IEEE ICASSP 2017 (2017)
16. Girdhar, R., El-Nouby, A., Liu, Z., Singh, M., Alwala, K.V., Joulin, A., Misra, I.: Imagebind: One embedding space to bind them all. In: CVPR (2023)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
18. Hertz, A., Mokady, R., Tenenbaum, J., Aberman, K., Pritch, Y., Cohen-Or, D.: Prompt-to-prompt image editing with cross attention control. In: ICLR (2023)
19. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. In: Advances in Neural Information Processing Systems (2017)
20. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: NeurIPS (2020)
21. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation (1997)
22. Jeong, Y., Ryoo, W., Lee, S., Seo, D., Byeon, W., Kim, S., Kim, J.: The power of sound (tpos): Audio reactive video generation with stable diffusion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7822–7832 (2023)
23. Joao, C., Andrew, Z.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
24. Kim, C.D., Kim, B., Lee, H., Kim, G.: Audiocaps: Generating captions for audios in the wild. In: NAACL-HLT (2019)
25. Korbar, B., Tran, D., Torresani, L.: Cooperative learning of audio and video models from self-supervised synchronization. In: Advances in Neural Information Processing Systems (2018)
26. Lee, S.H., Oh, G., Byeon, W., Bae, J., Kim, C., Ryoo, W.J., Yoon, S.H., Kim, J., Kim, S.: Sound-guided semantic video generation. In: ECCV (2022)
27. Lee, S., Kong, C., Jeon, D., Kwak, N.: Aadiff: Audio-aligned video synthesis with text-to-image diffusion. In: CVPR Workshop on Content Generation (2023)
28. Luo, S., Yan, C., Hu, C., Zhao, H.: Diff-foley: Synchronized video-to-audio synthesis with latent diffusion models. In: NeurIPS (2023)
29. Luping, L., Yi, R., Zhijie, L., Zhou, Z.: Pseudo numerical methods for diffusion models on manifolds. In: International Conference on Learning Representations (2022)
30. Owens, A., Isola, P., McDermott, J., Torralba, A., Adelson, E.H., Freeman, W.T.: Visually indicated sounds. In: CVPR (2016)
31. Pedro Morgado, Ishan Misra, N.V.: Robust audio-visual instance discrimination. In: Computer Vision and Pattern Recognition (CVPR), IEEE/CVF Conf. on (2021)
32. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: ICML (2021)
33. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: CVPR (2022)
34. Tang, Z., Yang, Z., Zhu, C., Zeng, M., Bansal, M.: Any-to-any generation via composable diffusion. In: Thirty-seventh Conference on Neural Information Processing Systems (2023), https://openreview.net/forum?id=2EDqbSCnmF

35. Tsuchida, S., Fukayama, S., Hamasaki, M., Goto, M.: Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In: Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019. Delft, Netherlands (Nov 2019)
36. Unterthiner, T., van Steenkiste, S., Kurach, K., Marinier, R., Michalski, M., Gelly, S.: Towards accurate generative models of video: A new metric & challenges. In: arXiv (2019)
37. Wang, J., Yuan, H., Chen, D., Zhang, Y., Wang, X., Zhang, S.: Modelscope text-to-video technical report (2023)
38. Xue, H., Hang, T., Zeng, Y., Sun, Y., Liu, B., Yang, H., Fu, J., Guo, B.: Advancing high-resolution video-language representation with large-scale video transcriptions. In: International Conference on Computer Vision and Pattern Recognition (CVPR) (2022)
39. Yariv, G., Gat, I., Benaim, S., Wolf, L., Schwartz, I., Adi, Y.: Diverse and aligned audio-to-video generation via text-to-video model adaptation (2023)