Condensing Action Segmentation Datasets via Generative Network Inversion

Guodong Ding, Rongyu Chen and Angela Yao
National University of Singapore
{dinggd, rchen, ayao}@comp.nus.edu.sg

Abstract

This work presents the first condensation approach for procedural video datasets used in temporal action segmentation. We propose a condensation framework that leverages generative prior learned from the dataset and network inversion to condense data into compact latent codes with significant storage reduced across temporal and channel aspects. Orthogonally, we propose sampling diverse and representative action sequences to minimize video-wise redundancy. Our evaluation on standard benchmarks demonstrates consistent effectiveness in condensing TAS datasets and achieving competitive performances. Specifically, on the Breakfast dataset, our approach reduces storage by over 500× while retaining 83% of the performance compared to training with the full dataset. Furthermore, when applied to a downstream incremental learning task, it yields superior performance compared to the state-of-the-art.

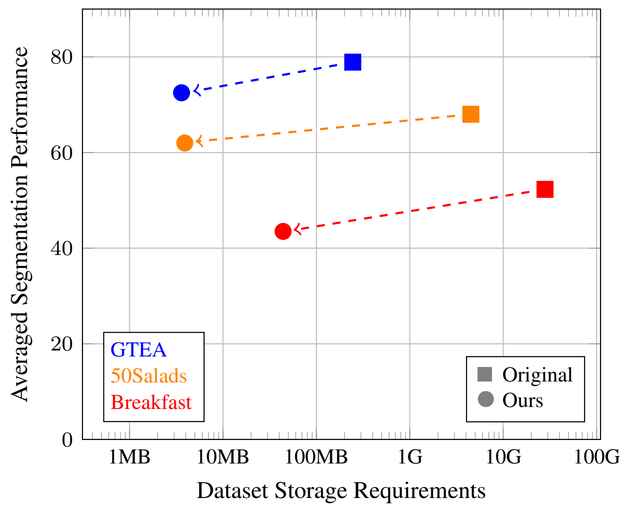


Figure 1. Comparison of action segmentation performance with dataset storage across common action segmentation benchmarks at different scales. Our method effectively reduces dataset storage while retaining competitive performance to the original setup.

1. Introduction

The effectiveness of data-driven deep neural network models hinges on training with large and diverse datasets. For instance, ImageNet [7] comprises over 14 million natural images for image classification, and Common Crawl [1] provides over 5 billion website pages to foster natural language processing tasks. However, the storage, processing, and training costs associated with massive datasets pose significant challenges, especially as data volumes continue to grow. This need for efficiency has driven interest in *dataset condensation* [36], a technique aimed at compressing large datasets into smaller, information-rich subsets. Such techniques have potential applications in downstream areas, including incremental learning and federated learning.

The core objective of dataset condensation is to learn a small synthetic dataset from an original large-scale dataset. Ideally, models trained on the synthetic data should perform comparably to those trained on the original. Significant efforts since then [4, 6, 22, 35, 43] have been dedicated to dataset condensation for image data. Video, however, remains under-explored. The recent work [37] condenses static and dynamic information from videos for action recognition in a two-stage framework. First, a synthetic "image" aggregates the static visual information, while a dynamic memory block is learned to capture and supplement the motion and dynamics.

This work aims to develop an effective dataset condensation approach for temporal action segmentation (TAS) [10]. TAS is a task that divides videos into segments by assigning labels on a per-frame basis to capture the sequence and duration of distinct actions. The videos are long, often spanning several minutes, creating significant challenges in raw data storage and processing. Dataset condensation for TAS presents unique challenges that are not typically encountered for action recognition datasets. First, TAS requires frame-level predictions rather than assigning a single label to an entire video or segment, as in action recognition. Consequently, dataset condensation for TAS must be capable of restoring the actual temporal resolutions of segments. Directly adapting the existing video condensation approach proposed by [37] for condensing TAS datasets is non-trivial. [37] uses a fixed-length frame processing, which cannot handle varying length segments while being able to restore the original temporal resolution. Additionally, the network in [37] is designed to work with RGB image sequences, while TAS typically uses pre-computed frame features [12, 21, 39]. Second, the action ordering in the video sequences is not rigidly fixed, but there is still a degree of structure or dependency that governs how actions unfold. Some actions can occur flexibly in their sequence, while others must follow a specific progression. As a result, redundant action sequences can emerge, where identical patterns in action order are presented multiple times. This sequence redundancy is not present in AR datasets.

This work presents the first study on condensing TAS datasets. We propose a framework that condenses the dataset with a generative model into a set of latent codes. The generative model is first trained on the entire dataset to learn the prior. Then, through network inversion, we optimize a set of latent codes as the condensed data, minimizing the error between the generated and original frame features.

Specifically, we use the TCA model proposed by [11] as our generative model. TCA is a conditional VAE reconstructing frame features while accounting for temporal dynamics. It is well-suited for our task because it compresses and restores the temporal resolution of action segments through it's coherence variable. TCA also condenses the feature dimension, as the latent code is more compact than the original feature. As a result, the storage requirement for a single action segment can be significantly reduced to the segment's latent code. This corresponds to the first challenge of TAS dataset condensation.

To address the second challenge of sequence redundancy, we propose a diversity-based sampling strategy grounded in the edit distance criterion. This approach iteratively selects the candidate sequence that maximizes the diversity within the chosen set. Empirically, we observed that TAS models achieve comparable performance with only half of the sequences. Our approach greatly reduces the storage while achieving comparable segmentation performance to the original dataset as shown in Fig. 1.

Contributions. Our contributions are summarized as follows: **(1)** This work is the first to investigate and propose an effective dataset condensation approach for the temporal action segmentation task. **(2)** We propose a generative condensation framework that first learns the generative prior on the dataset and then leverages the network inversion to condense data into compact latent codes. In addition, we propose a sampling strategy to further reduce the storage requirements based on sequence diversity. **(3)** Our approach effectively condenses TAS datasets of varying scales while consistently yielding comparable segmentation performances. Our approach outperforms the state-of-the-art by 10.7% on Breakfast under an incremental setup.

2. Related Work

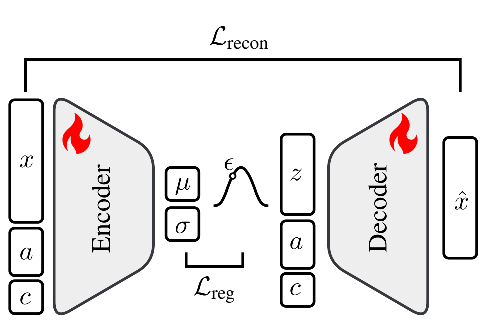
Temporal Action Segmentation. Various approaches have been proposed to tackle the temporal action segmentation task [10]. Fully supervised methods require dense annotations for each video frame [12, 39]. In contrast, semi-supervised methods [8, 32] only necessitate dense labels for a subset of the videos, leaving the rest unlabeled. There are also weaker forms of supervision, such as action transcripts [17], action sets [14, 19, 28], timestamps [20, 27], and activity labels [9]. Additionally, some approaches [18, 29, 30] operate in an unsupervised setting without relying on any action labels. Recent emerging directions include learning TAS incrementally [11] where procedural activities are learned sequentially and in an online [31, 45] fashion. However, the storage burden of these video datasets remains a pressing issue as these videos are long and with rich redundancy. Our work is the first to study the dataset condensation of temporal action segmentation datasets from the perspective of efficient data storage.

Dataset condensation. Dataset condensation (DC) is first formally introduced in [36]. The target is to learn a small set of synthetic data from an original large-scale dataset so that models trained on the synthetic dataset can perform comparably to those trained on the original. Based on the bilevel optimization formulation, varying techniques are leveraged to alleviate optimization difficulty including group optimization [46], the Neural Tangents Kernel (NTK) [24, 25], empirical Neural Network Gaussian Process (NNGP) [23]. Instead of matching model performance, [4, 6, 15, 42, 44] aim to indirectly achieve this by matching model parameters trained on original and condensed datasets. Another prominent line of work bypassing the bilevel optimization directly matches the distribution of original and condensed synthetic data [35, 43], known as distribution matching. Differently, [5, 41] introduced the generative model and synthesized in its latent space. Yet there is few work that extends this to the video domain. The work most related to ours is the dataset condensation for action recognition [37]. They disentangle the static and dynamic information in the video to minimize temporal redundancy. However, their method is not applicable to temporal action segmentation as they can not account for varying temporal resolution of actions and video redundancy. Thus, we are motivated to propose a generative framework to tackle the procedure video condensation problem for temporal action segmentation.

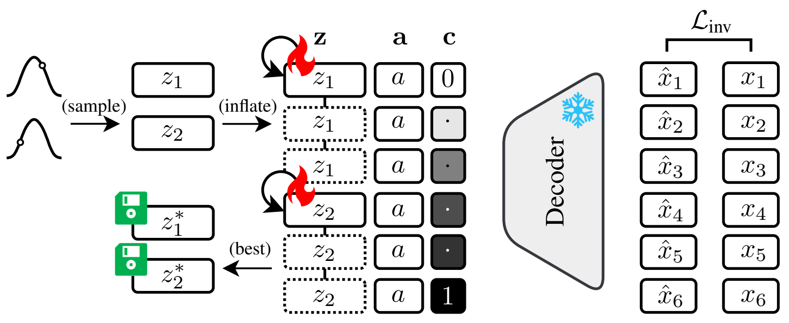
3. Dataset Condensation for TAS

3.1. Preliminaries

Temporal Action Segmentation (TAS). Temporal action segmentation is a task that segments untrimmed procedural videos into contiguous and non-overlapping actions. For example, given a video $\mathbf{X} = \{x^1, ..., x^T\}$ of T frames



(a) Generative Action Model



(sample)
z_1
z_2
(inflate)
$\mathbf{z}$
$\mathbf{a}$
$\mathbf{c}$
a
0
1
z_1^*
z_2^*
(best)
Decoder
$\mathcal{L}_{\text{inv}}$
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$
x_1 x_2 x_3 x_4 x_5 x_6

(b) Network Inversion

Figure 2. Generative Feature and Temporal Condensation Framework. (a) The generative action model is a conditional VAE that is trained to reconstruct the input frames conditioned on the action class label and a coherence variable. (b) The network inversion aims to optimize between decoded and original segments. Randomly sampled latent codes z_1 and z_2 are first inflated over time to the segment length, then concatenated with the action label and coherence variable for decoding. During the optimization, only the latent codes get updated while the decoder always stays fixed. These optimized latent codes z_1^* and z_2^* are stored as the condensed representation of the original segment. 🔥 indicates parameter updates during learning, while the ❄ indicates that the parameter is kept frozen.

long, a segmentation model outputs N continuous action segments consecutive in time:

$$\mathbf{s}_{1:N} = (s_1, s_2, ..., s_N), \quad \text{where } s_n = (a_n, t_n, \ell_n), \quad (1)$$
$$\text{s.t.} \qquad t_{n+1} = t_n + \ell_n.$$

s_n is a segment of length of ℓ_n, with action class label $a_n \in \mathcal{A}$ from A predefined categories. The t_n denotes the starting timestamp of segment s_n. Alternatively, most existing works [12, 21, 39] formulate it as a frame-wise classification task ($\mathcal{L}_{\text{cls}}$) and encourage the continuity of action segments with a smoothing term ($\mathcal{L}_{\text{sm}}$) with the learning objective written as:

$$\mathcal{L}_{\text{tas}} = \mathcal{L}_{\text{cls}}(x, y) + \lambda \cdot \mathcal{L}_{\text{sm}}(x, y), \quad (2)$$

where $y \in \mathcal{A}$ is the action label and λ a trade-off parameter.

Dataset Condensation (DC). Let $\mathcal{R} = \{\mathbf{X}_r, \mathbf{Y}_r\}$ represent a real image dataset, where $\mathbf{X}_r \in \mathbb{R}^{n_r \times d}$ denotes the set of training samples and $\mathbf{Y}_r \in \mathbb{R}^{n_r \times c}$ corresponds to their associated labels. Here, n_r denotes the number of original samples while d and c represent the dimensionality of the input features and output labels, respectively. The objective of dataset condensation is to construct a synthetic dataset of n_s samples, *i.e.* $\mathcal{S} = \{\mathbf{X}_s, \mathbf{Y}_s\}$, where $\mathbf{X}_s \in \mathbb{R}^{n_s \times d}$ and $\mathbf{Y}_s \in \mathbb{R}^{n_s \times c}$, with a considerably reduced size compared to the real dataset, such that $n_s \ll n_r$.

3.2. Task Formulation

Given an original TAS dataset with n_r training videos, represented as $\mathcal{R} = \{(\mathbf{X}_i, \mathbf{Y}_i)\}_{i=1}^{n_r}$. Each video representation $\mathbf{X} \in \mathbb{R}^{T \times D}$ and its corresponding action label $\mathbf{Y} \in \mathbb{R}^{T \times A}$ have the same temporal length T. Here, D and A denote the dimensions of frame feature space ($x \in \mathbb{R}^D$) and the action space ($y \in \mathbb{R}^A$), respectively. The goal of dataset condensation is to create a compact subset $\mathcal{S} = \{\hat{\mathbf{X}}_i, \hat{\mathbf{Y}}_i\}_{i=1}^{n_s}$, where $\hat{\mathbf{X}} \in \mathbb{R}^{T' \times d}$ represents a condensed version of original video. The size of $\mathcal{S}$ is expected to be significantly smaller than that of the original dataset $\mathcal{R}$.

This objective can be achieved through two levels of dataset condensation: (1) Sample compression: reducing the dimensionality of each video such that $T' \times d \ll T \times D$ and (2) Sample Reduction: reducing the total number of samples, *i.e.*, $n_s \ll n_r$. In light of this, our condensation framework implements reductions at both levels: for sample compression, we propose a generative feature and temporal condensation technique using network inversion (Sec. 3.3). For sample reduction, we use a diversity-based sampling strategy (Sec. 3.4).

3.3. Generative Feature & Temporal Condensation

Generative models are compact yet flexible, able to produce outputs of various lengths, making them ideal for condensing TAS datasets. The condensation process involves two main stages. In the first stage, a generative model is trained to represent action segments. In the second stage, a network inversion process is applied to optimize the latent codes, ensuring the condensed dataset captures an optimal representation of the original action segments.

Generative Action Model. We choose the Temporally Coherent Action (TCA) model proposed in [11] as our generative action model. TCA is essentially a compact, two-layer MLP VAE trained to reconstruct frame features. Specifically, the encoder in the TCA model takes three inputs: frame feature x, action label a, and a coherence variable c. The variable c is mathematically defined as the relative position of the frame within its segment:

$$c_i = (i-1)/(\ell-1), \quad \text{and} \quad c_i \in [0,1]. \tag{3}$$

The VAE's encoder maps these inputs in a latent space while the decoder reconstructs the frame feature $\hat{x}$. We denote the encoder and decoder as $\mathbf{E}(x,a,c) = q_\phi(z|x,a,c)$ and $\mathbf{D}(z,a,c) = p_\theta(x|z,a,c)$, respectively. The TCA model is trained on the entire dataset's video frames with a reconstruction loss and a KL divergence regularizer:

$$\mathcal{L}_{\text{TCA}} = \underbrace{\mathbb{E}_z \log p_\theta(x|z,a,c)}_{\mathcal{L}_{\text{recon}}} - \underbrace{\mathrm{D}_{\text{KL}}(q_\phi(z|x,a,c)||p(z))}_{\mathcal{L}_{\text{reg}}}. \tag{4}$$

An overview of the generative action model is depicted in Fig. 2(a). In this way, a segment of $\mathbb{R}^{\ell \times D}$ can be efficiently compressed to a latent distribution characterized by the mean and standard deviation $(\mu, \sigma) \in \mathbb{R}^d$, d is the dimension of the latent space. An advantage of using a generative model for condensing TAS videos is the ability to restore the original resolution. Such a model can generate segments of any specified length ℓ, producing each frame $\hat{x}$ by decoding a randomly sampled latent code z as follows:

$$\hat{x}_i = p_\theta(x|z,a,c_i), \quad \text{and } i \in [1,...,\ell]. \tag{5}$$

To ensure temporal continuity of generated features, [11] suggests a fixed latent code z is applied across all frames within the same segment.

Network Inversion. The above model learns inherent action priors from the video dataset, enabling it to generate segments that reflect realistic actions. However, segments decoded from *randomly sampled* latent codes can still deviate significantly from real data. To limit the deviations, we propose using network inversion. Neural network inversion is the process of determining a neural network input when given the corresponding output. Formally, given a neural network $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that maps an input $x \in \mathbb{R}^n$ to an output $y \in \mathbb{R}^m$, where $y = f(x)$. Mathematically, given an output y, the objective of neural network inversion is to find an input x^* such that:

$$x^* = f^{-1}(y), \tag{6}$$

where f^{-1} represents an approximate or exact inverse of function f. Since neural networks are generally not invertible, the problem can be posed as an optimization problem:

$$x^* = \underset{x \in \mathbb{R}^d}{\arg\min}\, \mathcal{C}(f(x), y), \tag{7}$$

where $\mathcal{C}(\cdot,\cdot)$ is a cost function. Note that during the inversion, both $f(\cdot)$ and y remain fixed, while only x is updated.

As our generation depicted in Eq. (5) is at the segment level, the inversion objective from Eq. (7) becomes:

$$z^* = \underset{z \in \mathbb{R}^d}{\arg\min}\, \underbrace{||\mathbf{D}(\mathbf{z},\mathbf{a},\mathbf{c}) - \mathbf{x}||_2^2}_{\mathcal{L}_{\text{inv}}}, \tag{8}$$

where $\mathbf{z} = z \otimes \mathbf{1}_\ell$, $\mathbf{a} = a \otimes \mathbf{1}_\ell$, and $\mathbf{c} = [c_1, \ldots, c_\ell]$. $\mathbf{1}_\ell$ is a vector of ones of length ℓ. We choose the ℓ_2 norm as the cost function in the inversion loss $\mathcal{L}_{\text{inv}}$ to align with the reconstruction term $\mathcal{L}_{\text{recon}}$ in the generative model training (as shown in Eq. (4)). Upon performing the inversion, the optimized latent code $z^* \in \mathbb{R}^d$ is stored for each segment.

Instances per Segment. In TAS datasets, action segments can be particularly long, where a single global latent code may not suffice to restore the full complexity and temporal dynamics of the entire segment. This limitation highlights the need for finer-grained approximations. To address this, we introduce the concept of *instances per segment*, which divides each segment into smaller, finer-grained instances for more precise network inversion. This is akin to the instances per class commonly used in existing dataset condensation works [40]. We evenly split segments into smaller instances to enable inversions at local scales.

Specifically, during the inversion step, for a given segment, we first initialize a set of K random codes $\{z_k\}_{k=1}^K$. These codes are evenly inflated over time to match the actual length of the segment, yielding the vector $\mathbf{z} = [z_1 \otimes \mathbf{1}_{\ell_1}, ..., z_K \otimes \mathbf{1}_{\ell_K}]$, where $\ell_k = \frac{\ell}{K}$. The vector $\mathbf{z}$ is concatenated with the action label $\mathbf{a}$ and the coherence variable $\mathbf{c}$, and the combined input is fed into the decoder for network inversion as defined in Eq. (8). After performing the inversion, we store the set of optimal latent codes $\{z_k^*\}_{k=1}^K$ as the condensed representation of the segment's features. An illustrative depiction of the inversion process for a segment of length $\ell = 6$ with $K = 2$ instances per segment is shown in Fig. 2(b). Initially, two latent codes z_1, z_2 are randomly sampled and expanded temporally to generate the segment $\{\hat{x}\}$ through the decoder. The decoder remains fixed while only the latent codes are optimized. Once optimized, the final z_1^*, z_2^* are stored as the condensed segment.

The proposed framework simultaneously condenses feature and temporal dimensions and reduces the storage requirement for each segment. Specifically, a segment $\mathbf{x} \in \mathbb{R}^{\ell \times D}$ can be efficiently condensed into latent codes $\mathbf{z} \in \mathbb{R}^{K \times d}$, with the condensation factor given by $\frac{\ell \cdot D}{K \cdot d}$. At the video level, our framework condenses the original $\mathbf{X} \in \mathbb{R}^{T \times D}$ into a reduced representation $\hat{\mathbf{X}}^* \in \mathbb{R}^{KN \times d}$, where N denotes the number of segments in the video, which is substantially smaller than the original video length T.

3.4. Diverse Sequence Sampling

The condensation process described above occurs at the sample level, reducing both feature and temporal dimensions. To account for sample redundancy, we introduce a diversity-based pruning strategy. Our intuition is that the selected sequences, taken collectively, should capture the maximum diversity of action ordering within the dataset. This ensures that the pruned set retains the broadest range of unique temporal patterns and action variations. Edit distance measures the minimum operations needed to transform one sequence into another, making it suitable for quantifying sequence diversity. Given two action sequences $\mathbf{s}_i$ and $\mathbf{s}_j$, we quantify the diversity with the normalized edit distance between them:

$$\text{Edit}(\mathbf{s}_i, \mathbf{s}_j) = \frac{e[|\mathbf{s}_i|, |\mathbf{s}_j|]}{\max(|\mathbf{s}_i|, |\mathbf{s}_j|)}, \quad \text{and} \tag{9}$$

$$e[m,n] = \begin{cases} \max(m,n), & \min(m,n)=0 \\ \min(e[m-1,n]+1, e[m,n-1]+1, & \\ \quad e[m-1,n-1]+\mathbb{1}(\mathbf{s}_i^m \neq \mathbf{s}_j^n)), & \text{otherwise.} \end{cases}$$

where m, n denote the action index within two comparing sequences, respectively. $\mathbb{1}(\cdot)$ is an indicator function.

We then apply a furthest point sampling strategy, commonly used in point clouds [26], to progressively select sequence $\mathbf{s}^*$ that maximizes the diversity until the desired set cardinality is reached. Specifically:

$$\mathbf{s}^* = \arg\max_{\mathbf{s}_i \in \mathcal{D} \setminus \mathcal{S}} \min_{\mathbf{s}_j \in \mathcal{S}} \text{Edit}(\mathbf{s}_i, \mathbf{s}_j), \tag{10}$$

where $\mathcal{D}$ is the original dataset and $\mathcal{S}$ the selected set, and $|\mathcal{S}| = \gamma|\mathcal{D}|$. We empirically set the size of the sampled to half of the original dataset, *i.e.*, $\gamma = 0.5$. This yields an extra $\sim$50% reduction in the storage of latent codes.

3.5. Decoding for TAS

Neural networks are sensitive to input resolution, and training a TAS model on low-resolution or condensed input can lead to suboptimal performance. Therefore, restoring the original resolution of input data is essential for the segmentation model to learn effectively. Different than the random generation in [11], we restore the action segments with their respective latent codes $\{z_k^*\}$, action labels a and length ℓ (coherence variable c), with the decoder $\mathbf{D}$ as follows:

$$\hat{\mathbf{x}}^* = \mathbf{D}(\mathbf{z}^*, \mathbf{a}, \mathbf{c}), \tag{11}$$

where $\mathbf{z}^* = [z_1^* \otimes \mathbf{1}_{\ell_1}, ..., z_K^* \otimes \mathbf{1}_{\ell_K}]$. These restored segments $\hat{\mathbf{x}}^*$ are then concatenated in time to form videos $\hat{\mathbf{X}}^*$, and their temporal order follows the symbolic sequence stored in the pruned set S. Hence, the training objective in Eq. (2) of the segmentation model becomes:

$$\mathcal{L}_{\text{tas}} = \mathcal{L}_{\text{cls}}(\hat{x}^*, y) + \lambda \cdot \mathcal{L}_{\text{sm}}(\hat{x}^*, y), \tag{12}$$

Details of the loss terms are given in the Supplementary.

4. Experiments

4.1. Datasets and Evaluation

Datasets. We evaluate our approach on three common TAS benchmarks that vary in storage scales. **GTEA** [13] contains 28 videos of seven kitchen activities composing 11 different actions. **50Salads** [33] has 50 videos with 19 action classes. **Breakfast** [16] dataset comprises 1,712 undirected breakfast preparation videos. There are 10 activities and a total of 48 action classes; each video features 5 to 14 actions. In terms of **storage**, the three datasets are at three scales: GTEA is the smallest at 245 MB, 50Salads is in the middle at 4.5 GB, and Breakfast is the largest at 28 GB. For all datasets, we use the I3D [3] feature representations and evaluate with the standard splits. Although I3D initially compresses frames by transforming RGB data into feature space, the original temporal resolution remains.

Evaluation Measures. TAS is evaluated using three metrics: frame-wise accuracy (Acc), segment-wise edit score (Edit), and F1 score with varying overlap thresholds of 10%, 25%, and 50%. In addition to these conventional TAS metrics, we also report the storage size to highlight the level of dataset condensation.

4.2. Implementation

Generative Network Inversion. We use the TCA [11] as our generative model, and follow their implementation as a two-layer MLP for both encoder and decoder with the latent size $d = 256$. On each dataset, we train the model for $7.5K$ epochs with a learning rate of $1e^{-3}$. For the network inversion, we optimize Eq. (8) for $10K$ iterations to obtain the optimal latent codes z^*. In all our experiments, unless otherwise specified, we set the number of instances per segment $K = 8$ and the sequence sampling ratio $\gamma = 0.5$.

Segmentation Backbones. We evaluate the effectiveness of our dataset condensation framework with two popular TAS backbones, *i.e.*, MSTCN [12] and ASFormer [39]. The former is a convolution-based segmentation model, while the latter is based on transformer architectures. We train MSTCN with a learning rate of $5e^{-4}$ for 50 epochs and $1e^{-4}$ for 30 epochs with ASFormer.

Baselines. As the first work to address dataset condensation for TAS, we establish the following baselines for comparison. Recognizing that storage size is a key evaluation aspect of dataset condensation approaches, we vigorously implement the following with aligned storage sizes to ensure fair comparisons:

– "**Original**" uses features of standard TAS datasets and no dataset condensation techniques are applied.

– "**Mean**" is a straightforward method that stores the average frame features of action segments as representatives. During TAS training, each average feature is repeated to match the segment length, creating a static *boring* video [44]. This method effectively reduces video length to the number of segments, *i.e.*, $\mathbb{R}^{T\times D} \rightarrow \mathbb{R}^{N\times D}$.

– "**Coreset**" utilizes the Herding [38] to identify the frame feature closest to the mean feature of the segment. The selected frames are then upsampled similarly to "Mean" to restore the original temporal resolution. Therefore, they have the same condensation ratio.

	GTEA [13]				50Salads [33]				Breakfast [16]			
	Acc	Edit	F1@{10, 25, 50}	Storage	Acc	Edit	F1@{10, 25, 50}	Storage	Acc	Edit	F1@{10, 25, 50}	Storage
						MS-TCN [12]						
Original	79.0	76.3	85.8 / 83.4 / 69.8	245 MB	80.6	63.1	69.9 / 67.4 / 59.0	4.5 GB	67.2	60.6	50.5 / 46.3 / 36.8	28 GB
Mean	71.2	**73.3**	77.1 / 73.7 / 59.4	7.2 MB	69.0	42.7	50.0 / 46.1 / 37.4	7.8 MB	47.6	31.8	27.8 / 23.3 / 15.6	96 MB
Coreset [38]	66.7	66.1	72.4 / 68.9 / 53.2	7.2 MB	61.7	43.3	49.9 / 46.3 / 35.4	7.8 MB	49.7	36.8	32.3 / 27.5 / 19.3	96 MB
TCA [11]	60.9	54.1	59.2 / 55.3 / 39.3	-	56.4	33.6	39.8 / 35.8 / 25.9	-	34.2	20.7	17.9 / 13.8 / 8.4	-
Encoded	70.4	65.5	72.2 / 68.8 / 52.1	3.6 MB	69.0	43.6	50.6 / 46.0 / 37.4	3.9 MB	37.9	49.8	40.0 / 32.8 / 19.4	44 MB
Ours	**75.2**	71.9	**78.3 / 74.6 / 62.7**	3.6 MB	**74.4**	**59.5**	**65.1 / 61.0 / 50.2**	3.9 MB	**55.5**	**45.6**	**46.7 / 41.1 / 28.7**	44 MB
Encoded†	70.5	72.7	77.1 / 73.7 / 59.8	30.5 MB	72.1	58.2	63.2 / 60.0 / 49.3	564 MB	43.4	53.2	45.8 / 37.4 / 22.8	3.4 GB
Ours†	**73.3**	**73.8**	**79.2 / 75.4 / 65.5**	30.5 MB	**72.8**	**59.8**	**65.2 / 61.3 / 51.3**	564 MB	**54.1**	**53.3**	**49.8 / 44.3 / 33.1**	3.4 GB
						ASFormer [39]						
Original	79.7	84.6	90.1 / 88.8 / 79.2	245 MB	85.6	79.6	85.1 / 83.4 / 76.0	4.5 GB	73.5	75.0	76.0 / 70.6 / 57.4	28 GB
Mean	72.2	76.9	82.1 / 79.7 / 65.1	7.2 MB	71.6	49.8	56.6 / 52.5 / 43.4	7.8 MB	52.2	43.2	43.5 / 38.3 / 26.7	96 MB
Coreset [38]	71.0	75.4	81.0 / 78.1 / 62.9	7.2 MB	69.4	46.8	56.6 / 52.9 / 39.6	7.8 MB	52.0	48.1	48.3 / 42.4 / 29.7	96 MB
TCA [11]	62.2	57.8	63.0 / 57.4 / 39.9	-	66.8	44.0	52.2 / 47.3 / 36.6	-	36.6	28.2	26.3 / 22.1 / 14.3	-
Encoded	69.2	70.2	73.3 / 67.3 / 49.8	3.6 MB	71.2	45.4	55.0 / 50.4 / 40.2	3.9 MB	37.6	53.6	50.7 / 41.3 / 24.0	44 MB
Ours	**77.9**	**82.7**	**86.4 / 84.5 / 70.4**	3.6 MB	**81.2**	**68.9**	**77.0 / 73.8 / 64.7**	3.9 MB	**59.8**	**48.8**	**54.1 / 47.7 / 34.1**	44 MB
Encoded†	74.0	78.1	83.1 / **79.6** / 67.3	30.5 MB	75.6	60.1	67.7 / 64.2 / 53.5	564 MB	45.7	54.8	52.6 / 43.3 / 25.2	3.4 GB
Ours†	**75.0**	**79.0**	**83.6** / 79.5 / **67.7**	30.5 MB	**76.2**	**65.0**	**73.1 / 68.8 / 58.5**	564 MB	**61.1**	**61.4**	**62.4 / 56.0 / 42.1**	3.4 GB

Table 1. Performance comparison on dataset condensation for TAS on three common benchmarks with different backbones. Storage sizes are highlighted in colors (high, medium, low). Our method remarkably reduces storage while retaining competitive performances across different datasets and model architectures. More details of the settings (d, K, and γ) for each method are provided in the Supplementary.

– “**TCA**” [11] is a baseline that follows its original implementation in which action segments are generated directly from random latent codes. This method does not require storage for latent codes, as they can be sampled on the fly during decoding.
– “**Encoded**” is the closest to our setup, with the key difference being that, instead of using network inversion to obtain latent codes, it stores the mean of encoded segment frames. Specifically, $z_k = \text{mean}(\mu_1, ..., \mu_{\ell_k})$. This approach results in the same storage requirement as ours.
– “**Encoded**†” refers to a setup similar to “Encoded” except for removing the sequence sampling and setting the number of instances per segment to the actual segment length, *i.e.*, $K = \ell$, which creates a latent code for each individual frame. The approach condenses along the feature dimension rather than the temporal dimension.

4.3. Effectiveness

Table 1 compares our approach to the baselines on three widely adopted TAS benchmarks. As observed, approaches like “Mean” and “Coreset”, which primarily condense from the temporal aspect, achieve similar performance across all datasets while maintaining an identical storage size. Note that in the best scenario, *boring* videos generated by these approaches can account for up to 80% performance of training with the “Original”. This highlights the temporal redundancy present in videos. TCA [11] does not incur additional storage requirements for the latent code, yet it produces the lowest overall performance across all evaluation metrics on three datasets. Although the generated segments inherit the action priors learned from the dataset, it is still likely the decoded segments from randomly sampled latent codes may not align well with the original data. A segmentation model trained on these misaligned features may not generalize well to the real testing data.

By storing encoded mean features of segments from the encoder as latent codes and diverse sequence sampling, “Encoded” can manage to achieve segmentation performance comparable to the “Mean” baseline, while requiring only half the storage cost. The best performance is achieved by our approach, which adds a network inversion process on top of “Encoded”. By imposing network inversion, a significant performance gain in segmental metrics is observed. For instance, on the 50Salads dataset, the average F1 score is boosted by a substantial 14.1% (from 44.7% to 58.8%). This underscores the effectiveness of network inversion, as it adapts the latent codes to better reflect the actual data.

Comparing across storage sizes, our approach also significantly outperforms its counterparts, “Mean” and “Coreset”, while only requiring roughly half the storage burden – 44 MB compared to 96 MB on the Breakfast dataset.

Our proposed condensation framework is independent of

	Sampling	Acc	Edit	F1@{10, 25, 50}	Storage
GTEA	✗	76.3	74.8	80.0 / 78.0 / 61.7	7.2 MB
	Random	74.0	69.3	76.2 / 72.8 / 60.4	3.6 MB
	Ours	75.2	71.9	78.3 / 74.6 / 62.7	3.6 MB
50Salads	✗	75.3	60.0	66.2 / 62.6 / 49.9	7.8 MB
	Random	71.9	58.9	62.3 / 58.4 / 49.5	3.9 MB
	Ours	74.4	59.5	65.1 / 61.0 / 50.2	3.9 MB
Breakfast	✗	55.6	52.3	47.3 / 42.1 / 31.4	91 MB
	Random	52.3	39.9	41.2 / 36.5 / 24.1	44 MB
	Ours	55.5	45.6	46.7 / 41.1 / 28.7	44 MB

Table 2. Effectiveness of the sequence sampling strategies on three TAS benchmarks. Our proposed sampling outperforms random while retaining comparable performances to the case where no sequence sub-sampling is performed.

γ	Acc	Edit	F1@{10, 25, 50}	Storage	Ratio(%)
0.1	45.0	44.9	40.5 / 35.6 / 23.2	1.3 MB	0.53
0.2	53.1	49.1	50.4 / 44.6 / 30.6	1.9 MB	0.78
0.3	56.9	52.3	56.8 / 52.6 / 36.7	2.4 MB	0.98
0.4	73.6	**72.9**	77.1 / 74.2 / **63.1**	2.9 MB	1.18
0.5	**75.2**	71.9	**78.3 / 74.6** / 62.7	3.6 MB	1.47
1	76.3	74.8	80.0 / 78.0 / 61.7	7.2 MB	2.94

Table 3. Sequence sampling ratio (γ) effects on GTEA. With only 0.5, we can achieve comparable performances to the full $\gamma = 1$.

the segmentation model, making it compatible with different backbones. TAS performances in Tab. 1 with two segmentation backbones [12, 39] demonstrates consistent performance improvements over the baselines.

4.4. Ablation and Hyper-parameter Study

Sequence Sampling Strategy. To evaluate the effectiveness of our proposed diversity-based sequence sampling technique, we compare it against random sampling and report the results in Tab. 2. For all datasets, the default sampling ratio γ is set to 0.5. We first observe that, with a sampling ratio of 0.5, effectively reducing the number of samples by half, the segmentation performance is not significantly affected, highlighting sample redundancy in the video datasets. On the other hand, our strategy consistently outperforms the random sampling across all metrics. Specifically, on the 50Salads dataset, there is a 2.5% gap in the frame-wise accuracy (74.4% vs. 71.9%). The consistent performance gain over the counterpart underscores that, when constrained by a sequence budget, prioritizing the incorporation of diverse action sequences enhances the model's generalization capability more effectively.

Sequence Sampling Ratio γ. We further evaluate segmentation performances on the GTEA dataset using various sequence sampling ratios γ, as shown in Tab. 3. As γ increases, a greater number of sequences are used to train the segmentation model, leading to a clear trend of improvement on all segmentation metrics. Notably, there is a substantial performance boost when γ increases from 0.3 to 0.4, with a 16.7% improvement in Acc and 20.6% in the Edit score. Given the small scale of the GTEA dataset, a sampling ratio of 0.5 provides sufficient diversity in sampled sequences to effectively represent the dataset.

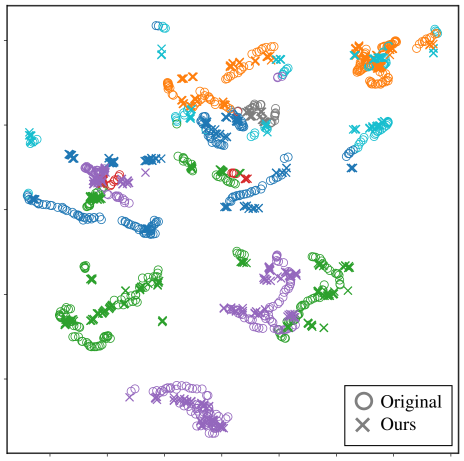


Figure 3. T-SNE visualization of original and decoded video features. Different colors indicate different action classes. The visualization shows that our generated features are well-aligned with original features. Best viewed when zoomed in.

IPS (K)	Acc	Edit	F1@{10, 25, 50}	Storage	Ratio(%)
Mean	47.6	31.8	27.8 / 23.3 / 15.6	96 MB	0.34
1	52.1	32.2	28.1 / 23.7 / 16.0	11 MB	0.04
2	52.7	38.4	34.9 / 30.1 / 21.1	22 MB	0.08
4	52.4	45.9	40.7 / 35.8 / 26.0	45 MB	0.15
8	**55.6**	**52.3**	47.3 / **42.1** / **31.4**	91 MB	0.31
16	54.2	51.2	**47.4** / 41.8 / 31.0	182 MB	0.62
†	54.1	53.3	49.8 / 44.3 / 33.1	3.4 GB	12.0

Table 4. Effect of the number of instance per segment (K) on Breakfast dataset without sequence sampling imposed. The ratio denotes the relative storage size of each setup compared to the original full dataset size. † indicates the setup in which latent codes are optimized on a per-frame basis.

Instances per Segment (K). We next examine how the number of instances per segment (K) impacts the segmentation performance and storage. Table 4 presents the segmentation performance without imposing the sequence sampling. Across various K values, our approach consistently outperforms the "Mean" baseline. Notably, even with $K = 1$, requiring only $1/8$ of the storage (11 MB vs. 96

	MSTCN [12]				ASFormer [39]			
	Acc	Edit	F1@{10, 25, 50}	Avg	Acc	Edit	F1@{10, 25, 50}	Avg
Mean [2]	18.4	14.1	14.4 / 13.0 / 9.8	13.9	12.1	10.4	10.7 / 9.7 / 7.9	10.2
TCA [11]	31.4	25.0	25.5 / 22.9 / 17.4	28.3	36.0	31.9	32.4 / 29.3 / 22.8	30.5
Ours	**38.2**	**31.6**	**32.8 / 29.7 / 22.8**	**31.0**	**46.7**	**41.1**	**41.7 / 38.3 / 30.8**	**39.7**
Original	44.4	40.2	40.8 / 36.9 / 28.8	38.2	51.7	46.2	47.2 / 43.4 / 34.7	44.6

Table 5. Performance comparison on the Breakfast dataset under the 10-task incremental setup following [11]. "Avg" indicates the averaged performance on all metrics. Our approach consistently surpasses the counterparts, with both MSTCN and ASFormer backbones.

MB), a performance margin is achieved over "Mean". In both cases, each segment is condensed into a single representation, but with different dimensions. The "mean" approach retains the original feature dimension $D = 2048$ while $K = 1$ maps the segment into one latent code with dimension $d = 256$. This improvement demonstrates our method's effectiveness under extreme storage constraints.

Frame-wise Acc shows minimal sensitivity to K, with performance remaining relatively stable and a maximum variation of 3.5% across different K values. However, segmental metrics, such as Edit and F1 scores, demonstrate a clear upward trend as K increases. For example, with $K = 1$, where each segment is condensed into a single latent code, the Edit score is 32.2%. Increasing K to 8 boosts the score to 52.3%, highlighting that a finer-grained condensation improves representation. This is because higher K values allow each segment to be represented by multiple latent codes rather than a single, highly compressed one, creating a more detailed representation that better approximates the original data. However, performance plateaus at $K = 16$, with no further improvement when condensation is conducted on a per-frame basis (denoted by †). This suggests that the performance may be constrained by the generative model's expressiveness. The storage size, as expected, increases linearly with K. With $K = 8$, the compression ratio for Breakfast is 0.31%, providing a good balance between storage efficiency and performance.

Visualization. We plot both the original and decoded frame features using T-SNE [34] for a sample video sequence from the GTEA dataset in Fig. 3. As shown, our network inversion approach effectively restores features that closely approximate the original, using the optimized latent codes. More visualizations are available in the Supplementary.

5. Incremental Action Segmentation

One promising application of dataset condensation is in continual learning as it effectively alleviates the storage burden associated with reply data.

We integrate our approach into the incremental temporal action segmentation (iTAS) framework recently proposed by [11]. In this setup, iTAS trains the segmentation model incrementally on different activity videos, with each stage focused on training with videos from a single activity. Each activity is treated as consisting of disjoint action classes, distinct from those of other activities. In their training, they assume that a small reservoir of samples from previous activities is considered available for the model to revisit, a process known as data replay. Our framework is applied to condense the replay data, the process is identical but on a per-activity basis.

Following [11], we conduct the experiment on the Breakfast dataset using the 10-task incremental setup, where each task corresponds to a single activity. Specifically, we train the TCA model for 2.5K epochs same as [11] and optimize the latent codes for the 10K iterations. To ensure the most efficient storage, we select the number of instances per segment as $K = 1$. This choice aligns with [11], where they also sample a single random latent code for each segment decoding. The results are summarized in Tab. 5. With our approach applied, we achieve a 6.8% increase in the final frame-wise accuracy with MSTCN [12] and a 10.7% increase with ASFormer [39]. Furthermore, our approach also significantly improves all segmental metrics by a margin larger than 10% with ASFormer.

6. Conclusion

This work introduces the first study on dataset condensation of temporal action segmentation. We propose a novel condensation framework to tackle the unique challenges of handling long procedural videos. It first condenses video segments into compact latent codes through generative network inversion from both temporal and channel perspectives. A diverse sequence sampling is further proposed to reduce the video-wise redundancy. Results on common benchmarks and with different backbones show our framework significantly reduces storage requirements, while preserving performance comparable to the original. This framework offers a practical solution for effectively condensing TAS datasets.

Acknowledgment This research / project is supported by the Ministry of Education, Singapore, under the Academic Research Fund Tier 1 (FY2022).

References

[1] Common crawl. https://commoncrawl.org/about/. 1

[2] Lama Alssum, Juan León Alcázar, Merey Ramazanova, Chen Zhao, and Bernard Ghanem. Just a glimpse: Rethinking temporal information for video continual learning. In *CVPRW*, 2023. 8

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 5

[4] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Dataset distillation by matching training trajectories. In *CVPR*, 2022. 1, 2

[5] George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A Efros, and Jun-Yan Zhu. Generalizing dataset distillation via deep generative prior. In *CVPR*, 2023. 2

[6] Justin Cui, Ruochen Wang, Si Si, and Cho-Jui Hsieh. Scaling up dataset distillation to imagenet-1k with constant memory. In *ICML*, 2023. 1, 2

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[8] Guodong Ding and Angela Yao. Leveraging action affinity and continuity for semi-supervised temporal action segmentation. In *ECCV*, 2022. 2

[9] Guodong Ding and Angela Yao. Temporal action segmentation with high-level complex activity labels. *IEEE TMM*, 2022. 2

[10] Guodong Ding, Fadime Sener, and Angela Yao. Temporal action segmentation: An analysis of modern techniques. *IEEE TPAMI*, 2023. 1, 2

[11] Guodong Ding, Hans Golong, and Angela Yao. Coherent temporal synthesis for incremental action segmentation. In *CVPR*, 2024. 2, 3, 4, 5, 6, 8

[12] Yazan Abu Farha and Jurgen Gall. Ms-tcn: Multi-stage temporal convolutional network for action segmentation. In *CVPR*, 2019. 2, 3, 5, 6, 7, 8

[13] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, 2011. 5, 6

[14] Mohsen Fayyaz and Jurgen Gall. Sct: Set constrained temporal transformer for set supervised action segmentation. In *CVPR*, 2020. 2

[15] Jang-Hyun Kim, Jinuk Kim, Seong Joon Oh, Sangdoo Yun, Hwanjun Song, Joonhyun Jeong, Jung-Woo Ha, and Hyun Oh Song. Dataset condensation via efficient synthetic-data parameterization. In *ICML*, 2022. 2

[16] Hilde Kuehne, Ali Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *CVPR*, 2014. 5, 6

[17] Hilde Kuehne, Alexander Richard, and Juergen Gall. Weakly supervised learning of actions from transcripts. *Computer Vision and Image Understanding*, 163:78–89, 2017. 2

[18] Anna Kukleva, Hilde Kuehne, Fadime Sener, and Jurgen Gall. Unsupervised learning of action classes with continuous temporal embedding. In *CVPR*, 2019. 2

[19] Jun Li and Sinisa Todorovic. Set-constrained viterbi for set-supervised action segmentation. In *CVPR*, 2020. 2

[20] Zhe Li, Yazan Abu Farha, and Jurgen Gall. Temporal action segmentation from timestamp supervision. In *CVPR*, 2021. 2

[21] Daochang Liu, Qiyue Li, Anh-Dung Dinh, Tingting Jiang, Mubarak Shah, and Chang Xu. Diffusion action segmentation. In *ICCV*, 2023. 2, 3

[22] Songhua Liu, Kai Wang, Xingyi Yang, Jingwen Ye, and Xinchao Wang. Dataset distillation via factorization. *NeurIPS*, 35, 2022. 1

[23] Noel Loo, Ramin Hasani, Alexander Amini, and Daniela Rus. Efficient dataset distillation using random feature approximation. *NeurIPS*, 2022. 2

[24] Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. Dataset meta-learning from kernel ridge-regression. *arXiv preprint arXiv:2011.00050*, 2020. 2

[25] Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. Dataset distillation with infinitely wide convolutional networks. *NeurIPS*, 2021. 2

[26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017. 5

[27] Rahul Rahaman, Dipika Singhania, Alexandre Thiery, and Angela Yao. A generalized and robust framework for timestamp supervision in temporal action segmentation. In *ECCV*, 2022. 2

[28] Alexander Richard, Hilde Kuehne, and Juergen Gall. Action sets: Weakly supervised action segmentation without ordering constraints. In *CVPR*, 2018. 2

[29] Saquib Sarfraz, Naila Murray, Vivek Sharma, Ali Diba, Luc Van Gool, and Rainer Stiefelhagen. Temporally-weighted hierarchical clustering for unsupervised action segmentation. In *CVPR*, 2021. 2

[30] Fadime Sener and Angela Yao. Unsupervised learning and segmentation of complex activities from video. In *CVPR*, 2018. 2

[31] Yuhan Shen and Ehsan Elhamifar. Progress-aware online action segmentation for egocentric procedural task videos. In *CVPR*, 2024. 2

[32] Dipika Singhania, Rahul Rahaman, and Angela Yao. Iterative contrast-classify for semi-supervised temporal action segmentation. In *AAAI*, 2022. 2

[33] Sebastian Stein and Stephen J McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *UbiComp*, 2013. 5, 6

[34] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9 (11), 2008. 8

[35] Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. Cafe: Learning to condense dataset by aligning features. In *CVPR*, 2022. 1, 2

[36] Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018. 1, 2

[37] Ziyu Wang, Yue Xu, Cewu Lu, and Yong-Lu Li. Dancing with still images: Video distillation via static-dynamic disentanglement. In *CVPR*, 2024. 1, 2

[38] Max Welling. Herding dynamical weights to learn. In *ICML*, 2009. 5, 6

[39] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. Asformer: Transformer for action segmentation. In *BMVC*, 2021. 2, 3, 5, 6, 7, 8

[40] Ruonan Yu, Songhua Liu, and Xinchao Wang. Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 4

[41] David Junhao Zhang, Heng Wang, Chuhui Xue, Rui Yan, Wenqing Zhang, Song Bai, and Mike Zheng Shou. Dataset condensation via generative model. *arXiv preprint arXiv:2309.07698*, 2023. 2

[42] Bo Zhao and Hakan Bilen. Dataset condensation with differentiable siamese augmentation. In *ICML*, 2021. 2

[43] Bo Zhao and Hakan Bilen. Dataset condensation with distribution matching. In *WACV*, 2023. 1, 2

[44] Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020. 2, 5

[45] Qing Zhong, Guodong Ding, and Angela Yao. Onlinetas: An online baseline for temporal action segmentation. *NeurIPS*, 2024. 2

[46] Yongchao Zhou, Ehsan Nezhadarya, and Jimmy Ba. Dataset distillation using neural feature regression. *NeurIPS*, 2022. 2